Wintellect.PowerCollections Namespace

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#
namespace Wintellect.PowerCollections

Visual Basic (Declaration)
Namespace Wintellect.PowerCollections

Visual C++
namespace Wintellect.PowerCollections
## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithms</strong></td>
<td>Algorithms contains a number of static methods that implement algorithms that work on collections. Most of the methods deal with the standard generic collection interfaces such as IEnumerable&lt;T&gt;, ICollection&lt;T&gt; and IList&lt;T&gt;.</td>
</tr>
<tr>
<td><strong>Bag(Of T)&gt;</strong></td>
<td>Bag&lt;T&gt; is a collection that contains items of type T. Unlike a Set, duplicate items (items that compare equal to each other) are allowed in a Bag.</td>
</tr>
<tr>
<td><strong>BigList(Of T)&gt;</strong></td>
<td>BigList&lt;T&gt; provides a list of items, in order with indices of the items ranging from 0 to one less than the count of items in the collection. BigList&lt;T&gt; is optimized for efficient operations on large (&gt;100 items) lists, especially for insertions, deletions, copies, and concatenations.</td>
</tr>
<tr>
<td><strong>CollectionBase(Of T)&gt;</strong></td>
<td>CollectionBase is a base class that can be used to more easily implement the generic ICollection&lt;T&gt; and non-generic ICollection interfaces.</td>
</tr>
<tr>
<td><strong>Deque(Of T)&gt;</strong></td>
<td>The Deque class implements a type of list known as a Double Ended Queue. A Deque is quite similar to a List, in that items have indices (starting at 0), and the item at any index can be efficiently retrieved. The difference between a List and a Deque lies in the efficiency of inserting elements at the beginning. In a List, items can be efficiently added to the end, but inserting an item at the beginning of the List is slow, taking time</td>
</tr>
</tbody>
</table>
proportional to the size of the List. In a Deque, items can be added to the beginning or end equally efficiently, regardless of the number of items in the Deque. As a trade-off for this increased flexibility, Deque is somewhat slower than List (but still constant time) when being indexed to get or retrieve elements.

DictionaryBase is a base class that can be used to more easily implement the generic IDictionary<T> and non-generic IDictionary interfaces.

ListBase is an abstract class that can be used as a base class for a read-write collection that needs to implement the generic IList<T>, non-generic IList collections. The derived class needs to override the following methods: Count, Clear, Insert, RemoveAt, and the indexer. The implementation of all the other methods in IList<T> and IList are handled by ListBase.

The MultiDictionary class that associates values with a key. Unlike an Dictionary, each key can have multiple values associated with it. When indexing an MultiDictionary, instead of a single value associated with a key, you retrieve an enumeration of values.

When constructed, you can chose to allow the same value to be associated with a key multiple times, or only one time.

MultiDictionaryBase is a base class that can be used to more easily implement a class that associates multiple values to a single key. The class implements the generic
**OrderedBag<Of<T>>**

OrderedBag<Of<T>> is a collection that contains items of type T. The items are maintained in sorted order. Unlike an OrderedSet, duplicate items (items that compare equal to each other) are allowed in an OrderedBag.

The OrderedBag<T>.View class is used to look at a subset of the items inside an ordered bag. It is returned from the Range, RangeTo, RangeFrom, and Reversed methods.

**OrderedDictionary<Of<TKey, TValue>>**

OrderedDictionary<Of<TKey, TValue>> is a collection that maps keys of type TKey to values of type TValue. The keys are maintained in a sorted order, and at most one value is permitted for each key.

The OrderedDictionary<TKey,TValue>.View class is used to look at a subset of the keys and values inside an ordered dictionary. It is returned from the Range, RangeTo, RangeFrom, and Reversed methods.

**OrderedMultiDictionary<Of<TKey, TValue>>**

The OrderedMultiDictionary class that associates values with a key. Unlike an OrderedDictionary, each key can have multiple values associated with it. When indexing an OrderedMultiDictionary, instead of a single value associated with a key, you retrieve an enumeration of values.

All of the keys are stored in sorted order. All the values associated with a given key are kept in sorted order as well.

When constructed, you can choose to allow the same value to be associated with a key multiple times, or only one time.
The `OrderedMultiDictionary<TKey,TValue>.View` class is used to look at a subset of the key and values inside an ordered multi-dictionary. It is returned from the `Range`, `RangeTo`, `RangeFrom`, and `Reversed` methods.

`OrderedSet<T>` is a collection that contains items of type T. The items are maintained in sorted order, and duplicate items are not allowed. Each item has an index in the set: the smallest item has index 0, the next smallest item has index 1, and so forth.

The `OrderedSet<T>.View` class is used to look at a subset of the items inside an ordered set. It is returned from the `Range`, `RangeTo`, `RangeFrom`, and `Reversed` methods.

`ReadOnlyCollectionBase<(Of <T>)>`

`ReadOnlyDictionaryBase<(Of TKey, TValue)>`

`ReadOnlyListBase<(Of <T>)>`

`ReadOnlyMultiDictionaryBase<(Of MultiDictionaryBase is a base class that can be used to more easily implement a class that associates multiple values to a single key. The class implements the generic

`OrderedMultiDictionary<(Of TKey, TValue)>..::View`

`OrderedSet<(Of <T>)>`

`OrderedSet<(Of <T>)>..::View`

`ReadOnlyCollectionBase<(Of <T>)>`

`ReadOnlyDictionaryBase<(Of TKey, TValue)>`
IDictionary<TKey, ICollection<TValue>> interface. The resulting collection is read-only -- items cannot be added or removed.

Set<T> is a collection that contains items type T. The item are maintained in a haphazard, unpredictable order, and duplicate items are not allowed.
### Structures

<table>
<thead>
<tr>
<th>Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair&lt;TFirst, TSecond&gt;</td>
<td>Stores a pair of objects within a single struct. This struct is useful to use as the T of a collection, or as the TKey or TValue of a dictionary.</td>
</tr>
<tr>
<td>Triple&lt;TFirst, TSecond, TThird&gt;</td>
<td>Stores a triple of objects within a single struct. This struct is useful to use as the T of a collection, or as the TKey or TValue of a dictionary.</td>
</tr>
</tbody>
</table>
## Delegates

<table>
<thead>
<tr>
<th>Delegate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinaryPredicate(Of T)</td>
<td>The BinaryPredicate delegate type encapsulates a method that takes two items of the same type, and returns a boolean value representing some relationship between them. For example, checking whether two items are equal or equivalent is one kind of binary predicate.</td>
</tr>
</tbody>
</table>

Send [comments](mailto:) about this topic to Microsoft.
Algorithms Class

See Also  Members

- Visual Basic (Declaration)  Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Algorithms contains a number of static methods that implement algorithms that work on collections. Most of the methods deal with the standard generic collection interfaces such as IEnumerable<T>, ICollection<T> and IList<T>.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static class Algorithms

Visual Basic (Declaration)

Public NotInheritable Class Algorithms

Visual C++

public ref class Algorithms abstract sealed
Inheritance Hierarchy

- System::Object
  Wintellect.PowerCollections::Algorithms
See Also

Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms Members

See Also  Methods

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
The type exposes the following members.
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BinarySearch</strong></td>
<td>Overloaded. Computes the cartesian product of two collections: all possible pairs of items, with the first item taken from the first collection and the second item taken from the second collection. If the first collection has N items, and the second collection has M items, the cartesian product will have N * M pairs.</td>
</tr>
<tr>
<td><strong>CartesianProduct(Of TFirst, TSecond) &gt;</strong></td>
<td>Concatenates all the items from several collections. The collections need not be of the same type, but must have the same item type.</td>
</tr>
<tr>
<td><strong>Concatenate(Of T) &gt;</strong></td>
<td>Convert a collection of items by applying a delegate to each item in the collection. The resulting collection contains the result of applying converter to each item in sourceCollection, in order.</td>
</tr>
<tr>
<td><strong>Convert(Of&lt;TSource, TDest) &gt;</strong></td>
<td>Overloaded. Count the number of items in an IEnumerable&lt;T&gt; collection. If a more specific collection type is being used, it is more efficient to use the Count property, if one is provided.</td>
</tr>
<tr>
<td><strong>Copy</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>Count(Of T) &gt;</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td><strong>CountEqual</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>CountWhere(Of T) &gt;</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>DisjointSets</strong></td>
<td></td>
</tr>
</tbody>
</table>
- **S EqualCollections**
  Overloaded.

- **S EqualSets**
  Overloaded.

- **S Exists(Of <T>)**
  Determines if a collection contains any item that satisfies the condition defined by predicate.

- **S Fill**
  Overloaded.

- **S FillRange**
  Overloaded.

- **S FindFirstIndexWhere(Of <T>)**
  Finds the index of the first item in a list that satisfies the condition defined by predicate.

- **S FindFirstWhere(Of <T>)**
  Finds the first item in a collection that satisfies the condition defined by predicate.

- **S FindIndicesWhere(Of <T>)**
  Enumerates the indices of all the items in list that satisfy the condition defined by predicate.

- **S FindLastIndexWhere(Of <T>)**
  Finds the index of the last item in a list that satisfies the condition defined by predicate.

- **S FindLastWhere(Of <T>)**
  Finds the last item in a collection that satisfies the condition defined by predicate.

- **S FindWhere(Of <T>)**
  Enumerates all the items in collection that satisfy the condition defined by predicate.

- **S FirstConsecutiveEqual**
  Overloaded.

- **S FirstConsecutiveWhere(Of <T>)**
  Finds the first occurence of count consecutive items in the list for which a given predicate returns true.

- **S FirstIndexOf**
  Overloaded.

- **S FirstIndexOfMany**
  Overloaded.

- **S ForEach(Of <T>)**
  Performs the specified action on each item in a collection.

Generates all the possible permutations of the items in collection.
If collection has N items, then N
factorial permutations will be generated. This method does not compare the items to determine if any of them are equal. If some items are equal, the same permutation may be generated more than once. For example, if the collection contains the three items A, A, and B, then this method will generate the six permutations, AAB, AAB, ABA, ABA, BAA, BAA (not necessarily in that order). To take equal items into account, use the GenerateSortedPermutations method.

GenerateSortedPermutations

GetCollectionEqualityComparer

GetComparerFromComparison<(Of <T>)>

GetComparisonFromComparer<(Of <T>)>

GetDictionaryConverter

GetIdentityComparer<(Of <T>)>

GetLexicographicalComparer

GetReverseComparer<(Of <T>)>

Equality and hash codes are
GetReverseComparison\(\langle\text{Of}\ <\text{T}\rangle\rangle\)

Reverses the order of comparison of an Comparison\(\langle\text{T}\rangle\). The resulting comparison can be used, for example, to sort a collection in descending order.

GetSetEqualityComparer

Overloaded.

IndexOfMaximum

Overloaded.

IndexOfMinimum

Overloaded.

IndicesOf

Overloaded.

IndicesOfMany

Overloaded.

IsProperSubsetOf

Overloaded.

IsSubsetOf

Overloaded.

LastIndexOf

Overloaded.

LastIndexOfMany

Overloaded.

LexicographicalCompare

Overloaded.

Maximum

Overloaded.

MergeSorted

Overloaded.

Minimum

Overloaded.

NCopiesOf\(\langle\text{Of}\ <\text{T}\rangle\rangle\)

Creates an IEnumerator that enumerates a given item \(n\) times.

Partition\(\langle\text{Of}\ <\text{T}\rangle\rangle\)

Partition a list or array based on a predicate. After partitioning, all items for which the predicate returned true precede all items for which the predicate returned false.

RandomShuffle

Overloaded.

RandomShuffleInPlace

Overloaded.

RandomSubset

Overloaded.

Range

Overloaded.

ReadOnly

Overloaded.

Creates a read-write IList\(\langle\text{T}\rangle\) wrapper around an array. When an array is
ReadWriteList<Of <T> >

Implicitly converted to an IList<T>, changes to the items in the array cannot be made through the interface. This method creates a read-write IList<T> wrapper on an array that can be used to make changes to the array.

Use this method when you need to pass an array to an algorithms that takes an IList<T> and that tries to modify items in the list. Algorithms in this class generally do not need this method, since they have been design to operate on arrays even when they are passed as an IList<T>.

RemoveDuplicates
RemoveDuplicatesInPlace

RemoveWhere<Of <T> >

Replace
ReplaceInPlace

Reverse<Of <T> >
ReverseInPlace<Of <T> >

Rotate<Of <T> >
RotateInPlace<Of <T> >

SearchForSubsequence
SetDifference
SetIntersection
SetSymmetricDifference
SetUnion
Sort
SortInPlace

Overloaded.
Overloaded.
Removes all the items in the collection that satisfy the condition defined by predicate.
Overloaded.
Overloaded.
Reverses a list and returns the reversed list, without changing the source list.
Reverses a list or array in place.
Rotates a list and returns the rotated list, without changing the source list.
Rotates a list or array in place.
Overloaded.
Overloaded.
Overloaded.
Overloaded.
Overloaded.
StablePartition<Of <T> >

Partition a list or array based on a predicate. After partitioning, all items for which the predicate returned true precede all items for which the predicate returned false. The partition is stable, which means that if items X and Y have the same result from the predicate, and X precedes Y in the original list, X will precede Y in the partitioned list.

StableSort

Overloaded.

StableSortInPlace

Overloaded.

ToArray<Of <T> >

Create an array with the items in a collection.

ToString

Overloaded.

TrueForAll<Of <T> >

Determines if all of the items in the collection satisfy the condition defined by predicate.

TryFindFirstWhere<Of <T> >

Finds the first item in a collection that satisfies the condition defined by predicate.

TryFindLastWhere<Of <T> >

Finds the last item in a collection that satisfies the condition defined by predicate.

TypedAs

Overloaded.

Untyped

Overloaded.
See Also

Algorithms
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

public static class Algorithms

Visual Basic (Declaration)

Public NotInheritable Class Algorithms

Visual C++

public ref class Algorithms abstract sealed

The type exposes the following methods.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BinarySearch</code></td>
<td>Overloaded. Computes the cartesian product of two collections: all possible pairs of items, with the first item taken from the first collection and the second item taken from the second collection. If the first collection has N items, and the second collection has M items, the cartesian product will have N * M pairs.</td>
</tr>
<tr>
<td><code>CartesianProduct&lt;Of &lt;TFirst, TSecond&gt;&gt;</code></td>
<td>Concatenates all the items from several collections. The collections need not be of the same type, but must have the same item type. Convert a collection of items by applying a delegate to each item in the collection. The resulting collection contains the result of applying converter to each item in sourceCollection, in order.</td>
</tr>
<tr>
<td><code>Concatenate&lt;Of &lt;T&gt;&gt;</code></td>
<td>Overloaded. Count the number of items in an IEnumerable&lt;T&gt; collection. If a more specific collection type is being used, it is more efficient to use the Count property, if one is provided.</td>
</tr>
<tr>
<td><code>Convert&lt;Of &lt;TSource, TDest&gt;&gt;</code></td>
<td>Overloaded. Counts the number of items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td><code>Copy</code></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><code>Count&lt;Of &lt;T&gt;&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>CountEqual</code></td>
<td></td>
</tr>
<tr>
<td><code>CountWhere&lt;Of &lt;T&gt;&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>DisjointSets</code></td>
<td></td>
</tr>
</tbody>
</table>
- **EqualCollections**  
  Overloaded.

- **EqualSets**  
  Overloaded.

- **Exists(Of <T>)**
  Determines if a collection contains any item that satisfies the condition defined by predicate.

- **Fill**  
  Overloaded.

- **FillRange**  
  Overloaded.

- **FindFirstIndexWhere(Of <T>)**
  Finds the index of the first item in a list that satisfies the condition defined by predicate.

- **FindFirstWhere(Of <T>)**
  Finds the first item in a collection that satisfies the condition defined by predicate.

- **FindIndicesWhere(Of <T>)**
  Enumerates the indices of all the items in list that satisfy the condition defined by predicate.

- **FindLastIndexWhere(Of <T>)**
  Finds the index of the last item in a list that satisfies the condition defined by predicate.

- **FindLastWhere(Of <T>)**
  Finds the last item in a collection that satisfies the condition defined by predicate.

- **FindWhere(Of <T>)**
  Enumerates all the items in collection that satisfy the condition defined by predicate.

- **FirstConsecutiveEqual**  
  Overloaded.

- **FirstConsecutiveWhere(Of <T>)**
  Finds the first occurrence of count consecutive items in the list for which a given predicate returns true.

- **FirstIndexOf**  
  Overloaded.

- **FirstIndexOfMany**  
  Overloaded.

- **ForEach(Of <T>)**
  Performs the specified action on each item in a collection.

Generates all the possible permutations of the items in collection. If collection has N items, then N
factorial permutations will be generated. This method does not compare the items to determine if any of them are equal. If some items are equal, the same permutation may be generated more than once. For example, if the collections contains the three items A, A, and B, then this method will generate the six permutations, AAB, AAB, ABA, ABA, BAA, BAA (not necessarily in that order). To take equal items into account, use the GenerateSortedPermutations method.

**GeneratePermutations(Of `T`)**

**GenerateSortedPermutations** Overloaded.

**GetCollectionEqualityComparer** Overloaded.

**GetComparerFromComparison(Of `T`)**

Given a comparison delegate that compares two items of type `T`, gets an IComparer<T> instance that performs the same comparison.

**GetComparisonFromComparer(Of `T`)**

Given in IComparer<T> instance that comparers two items from type `T`, gets a Comparison delegate that performs the same comparison.

**GetDictionaryConverter** Overloaded.

**GetIdentityComparer(Of `T`)**

Gets an IEqualityComparer<T> instance that can be used to compare objects of type `T` for object identity only. Two objects compare equal only if they are references to the same object.

**GetLexicographicalComparer**

Reverses the order of comparison of an IComparer<T>. The resulting comparer can be used, for example, to sort a collection in descending order. Equality and hash codes are
Reverses the order of comparison of an `Comparison<T>`. The resulting comparison can be used, for example, to sort a collection in descending order.

Overloaded.

Partitions a list or array based on a predicate. After partitioning, all items for which the predicate returned `true` precede all items for which the predicate returned `false`.

Overloaded.

Creates an `IEnumerator` that enumerates a given item `n` times.

Overloaded.

Creates a read-write `IList<T>` wrapper around an array. When an array is
ReadWriteList(Of T)

Implicitly converted to an IList<T>, changes to the items in the array cannot be made through the interface. This method creates a read-write IList<T> wrapper on an array that can be used to make changes to the array.

Use this method when you need to pass an array to an algorithms that takes an IList<T> and that tries to modify items in the list. Algorithms in this class generally do not need this method, since they have been design to operate on arrays even when they are passed as an IList<T>.

RemoveDuplicates
RemoveDuplicatesInPlace

RemoveWhere(Of T)

Replace
ReplaceInPlace

Reverse(Of T)
ReverseInPlace(Of T)

Rotate(Of T)
RotateInPlace(Of T)

SearchForSubsequence
SetDifference
SetIntersection
SetSymmetricDifference
SetUnion
Sort
SortInPlace

Overloaded.
Overloaded.
Removes all the items in the collection that satisfy the condition defined by predicate.
Overloaded.
Overloaded.
Reverses a list and returns the reversed list, without changing the source list.
Reverses a list or array in place.
Rotates a list and returns the rotated list, without changing the source list.
Rotates a list or array in place.
Overloaded.
Overloaded.
Overloaded.
Overloaded.
Overloaded.
Overloaded.
**StablePartition<(Of <T>)>**

Partition a list or array based on a predicate. After partitioning, all items for which the predicate returned true precede all items for which the predicate returned false. The partition is stable, which means that if items X and Y have the same result from the predicate, and X precedes Y in the original list, X will precede Y in the partitioned list.

**StableSort**
Overloaded.

**StableSortInPlace**
Overloaded.

**ToArray<(Of <T>)>**
Create an array with the items in a collection.

**ToString**
Overloaded.

**TrueForAll<(Of <T>)>**
Determines if all of the items in the collection satisfy the condition defined by predicate.

**TryFindFirstWhere<(Of <T>)>**
Finds the first item in a collection that satisfies the condition defined by predicate.

**TryFindLastWhere<(Of <T>)>**
Finds the last item in a collection that satisfies the condition defined by predicate.

**TypedAs**
Overloaded.

**Untyped**
Overloaded.
See Also

Algorithms
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::BinarySearch Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinarySearch(Of T)&gt;(IList(Of T), T, Int32%)</td>
<td>Searches a sorted list for an item via binary search. The list must be sorted by the natural ordering of the type (it's implementation of IComparable&lt;T&gt;).</td>
</tr>
<tr>
<td>BinarySearch(Of T)&gt;(IList(Of T), T, IComparer(Of T), Int32%)</td>
<td>Searches a sorted list for an item via binary search. The list must be sorted by the ordering in the passed instance of IComparer&lt;T&gt;.</td>
</tr>
<tr>
<td>BinarySearch(Of T)&gt;(IList(Of T), T, Comparison(Of T), Int32%)</td>
<td>Searches a sorted list for an item via binary search. The list must be sorted by the ordering in the passed Comparison&lt;T&gt; delegate.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::BinarySearch(Of <T>-peer) Method (IList(Of <T>-peer), T, Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Searches a sorted list for an item via binary search. The list must be sorted by the natural ordering of the type (it's implementation of IComparable<T>).

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public static int BinarySearch<T>(
    IList<T> list,
    T item,
    out int index
) where T : IComparable<T>
```

### Visual Basic (Declaration)

```vbnet
Public Shared Function BinarySearch(Of T As IComparable(Of T)) ( _
    list As IList(Of T), _
    item As T, _
    <OutAttribute> ByRef index As Integer _
) As Integer
```

### Visual C++

```csharp
public:
    generic<typename T>
    where T : IComparable<T>
    static int BinarySearch ( 
        IList<T>^ list,
        T item,
        [OutAttribute] int% index
    )
```

### Parameters

**list**

`IList<Of <T>>`

The sorted list to search.

**item**

`T`

The item to search for.

**index**
**Int32%**

Returns the first index at which the item can be found. If the return value is zero, indicating that item was not present in the list, then this returns the index at which item could be inserted to maintain the sorted order of the list.

**Return Value**

The number of items equal to item that appear in the list.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::BinarySearch(Of T) Method (IList(Of T), T, IComparer(Of T), Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Searches a sorted list for an item via binary search. The list must be sorted by the ordering in the passed instance of IComparer<T>.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static int BinarySearch<T>(
    IList<T> list,
    T item,
    IComparer<T> comparer,
    out int index
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function BinarySearch(Of T) ( _
    list As IList(Of T), _
    item As T, _
    comparer As IComparer(Of T), _
    [OutAttribute] ByRef index As Integer _
) As Integer
```

Visual C++

```cpp
public:
    template<typename T>
    static int BinarySearch ( _
        IList<T>* list,
        T item,
        IComparer<T>* comparer,
        [OutAttribute] int& index
    )
```

Parameters

list
    IList<(Of <T>)> The sorted list to search.

item
    T The item to search for.
comparer
    `IComparer(Of <T>)`
    The comparer instance used to sort the list. Only the Compare method is used.

index
    `Int32`
    Returns the first index at which the item can be found. If the return value is zero, indicating that item was not present in the list, then this returns the index at which item could be inserted to maintain the sorted order of the list.

**Return Value**

The number of items equal to item that appear in the list.
Type Parameters

T
See Also

* **Algorithms** Class
  *Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
Searches a sorted list for an item via binary search. The list must be sorted by the ordering in the passed `Comparison<T>` delegate.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static int BinarySearch<T>(
    IList<T> list,
    T item,
    Comparison<T> comparison,
    out int index
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function BinarySearch(Of T) ( _
    list As IList(Of T), _
    item As T, _
    comparison As Comparison(Of T), _
    <OutAttribute> ByRef index As Integer _
) As Integer
```

Visual C++

```cpp
public:
    generic<typename T>
    static int BinarySearch ( 
        IList<T>^ list,
        T item,
        Comparison<T>^ comparison,
        [OutAttribute] int% index
    )
```

Parameters

list
    IList<(Of <T>)>
    The sorted list to search.

item
    T
    The item to search for.
comparison
   \texttt{Comparison<}(\texttt{Of }<\texttt{T}>)\texttt{>}
   The comparison delegate used to sort the list.

index
   \texttt{Int32}\%
   Returns the first index at which the item can be found. If the return value is zero, indicating that item was not present in the list, then this returns the index at which item could be inserted to maintain the sorted order of the list.

\textbf{Return Value}

The number of items equal to item that appear in the list.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::CartesianProduct(Of <TFirst, TSecond>)> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the cartesian product of two collections: all possible pairs of items, with the first item taken from the first collection and the second item taken from the second collection. If the first collection has N items, and the second collection has M items, the cartesian product will have N * M pairs.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
C#

public static IEnumerable<Pair<TFirst, TSecond>> CartesianProduct<TFirst, TSecond>(IEnumerable<TFirst> first, IEnumerable<TSecond> second)

Visual Basic (Declaration)

Public Shared Function CartesianProduct(Of TFirst, TSecond) ( _
  first As IEnumerable(Of TFirst), _
  second As IEnumerable(Of TSecond) _
) As IEnumerable(Of Pair(Of TFirst, TSecond))

Visual C++

public:
  generic<typename TFirst, typename TSecond>
  static IEnumerable<Pair<TFirst, TSecond>>^ CartesianProduct ( _
    IEnumerable<TFirst>^ first,
    IEnumerable<TSecond>^ second
  )

Parameters

first
  IEnumerable<(Of <TFirst>)>
  The first collection.

second
  IEnumerable<(Of <TSecond>)>
  The second collection.

Return Value

An IEnumerable<Pair<TFirst, TSecond>> that enumerates the cartesian product
of the two collections.
Type Parameters

TFirst
   The type of items in the first collection.
TSecond
   The type of items in the second collection.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Concatenates all the items from several collections. The collections need not be of the same type, but must have the same item type.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static IEnumerable<T> Concatenate<T>(
   params IEnumerable<T>[] collections
)

Visual Basic (Declaration)

Public Shared Function Concatenate(Of T) ( _
   ParamArray collections As IEnumerable(Of T)() _
) As IEnumerable(Of T)

Visual C++

public:
   generic<typename T>
   static IEnumerable<T>^ Concatenate (  
      ... array<IEnumerable<T>^>^ collections
   )

Parameters

collections
   array<IEnumerable<(Of <T>)>>[]()
   The set of collections to concatenate. In many languages, this parameter can be specified as several individual parameters.

Return Value

An IEnumerable that enumerates all the items in each of the collections, in order.
Type Parameters

T
See Also

* Algorithms Class
  * Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Convert a collection of items by applying a delegate to each item in the collection. The resulting collection contains the result of applying converter to each item in sourceCollection, in order.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public static IEnumerable<TDest> Convert<TSource, TDest>(
    IEnumerable<TSource> sourceCollection,
    Converter<TSource, TDest> converter
)
```

### Visual Basic (Declaration)

```vbnet
Public Shared Function Convert(Of TSource, TDest) (_
    sourceCollection As IEnumerable(Of TSource), _
    converter As Converter(Of TSource, TDest) _) As IEnumerable(Of TDest)
```

### Visual C++

```cpp
public:

generic<typename TSource, typename TDest>
static IDictionary^ Convert (
    IDictionary^ sourceCollection, 
    Converter^ converter
)
```

## Parameters

- **sourceCollection**
  ```csharp
  IEnumerable<TSource>
  ```
  The collection of item being converted.

- **converter**
  ```csharp
  Converter<TSource, TDest>
  ```
  A delegate to the method to call, passing each item in sourceCollection.

## Return Value

The resulting collection from applying converter to each item in
sourceCollection, in order.
**Type Parameters**

**TSource**
- The type of items in the collection to convert.

**TDest**
- The type each item is being converted to.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...::ArgumentNullException</td>
<td>sourceCollection or converter is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...::Copy Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Copy(Of T)</code>&lt;br&gt;(IList(Of T), Int32)</td>
<td>Copies all of the items from the collection source to the list dest, starting at the index destIndex. If necessary, the size of the destination list is expanded.</td>
</tr>
<tr>
<td><code>Copy(Of T)</code>&lt;br&gt;(IEnumerable(Of T), array&lt;T&gt;<a href=""></a>, Int32)</td>
<td>Copies all of the items from the collection source to the array dest, starting at the index destIndex.</td>
</tr>
<tr>
<td><code>Copy(Of T)</code>&lt;br&gt;(IList(Of T), Int32, Int32)</td>
<td>Copies at most count items from the collection source to the list dest, starting at the index destIndex. If necessary, the size of the destination list is expanded. The source collection must not be the destination list or part thereof.</td>
</tr>
<tr>
<td><code>Copy(Of T)</code>&lt;br&gt;(IList(Of T), Int32, array&lt;T&gt;<a href=""></a>, Int32)</td>
<td>Copies at most count items from the collection source to the array dest, starting at the index destIndex. The source collection must not be the destination array or part thereof.</td>
</tr>
<tr>
<td><code>Copy(Of T)</code>&lt;br&gt;(IList(Of T), Int32, array&lt;T&gt;<a href=""></a>, Int32)</td>
<td>Copies count items from the list source, starting at the index sourceIndex, to the list dest, starting at the index destIndex. If necessary, the size of the destination list is expanded. The source and destination lists may be the same.</td>
</tr>
<tr>
<td><code>Copy(Of T)</code>&lt;br&gt;(IList(Of T), Int32, array&lt;T&gt;<a href=""></a>, Int32)</td>
<td>Copies count items from the list or array source, starting at the index sourceIndex, to the array dest, starting at the index destIndex. The source may be the same as the destination array.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...::.Copy(Of T) Method (IEnumerable(Of T), IList(Of T), Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies all of the items from the collection source to the list dest, starting at the index destIndex. If necessary, the size of the destination list is expanded.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static void Copy<T>(
    IEnumerable<T> source,
    IList<T> dest,
    int destIndex
)

Visual Basic (Declaration)

Public Shared Sub Copy(Of T) ( _
    source As IEnumerable(Of T), _
    dest As IList(Of T), _
    destIndex As Integer _
)

Visual C++

public:
    generic<typename T>
    static void Copy ( 
        IEnumerable<T>^ source,
        IList<T>^ dest,
        int destIndex
    )

Parameters

source
    IEnumerable<(Of <T)>>
    The collection that provide the source items.

dest
    IList<(Of <T)>>
    The list to store the items into.

destIndex
**Int32**
The index to begin copying items to.
Type Parameters

T
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::ArgumentOutOfRangeException</td>
<td>destIndex is negative or greater than dest.Count.</td>
</tr>
<tr>
<td>System..::ArgumentNullException</td>
<td>source or dest is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::Copy(Of <T>), Method (IEnumerable(Of <T>), array<T>[], , Int32)

**See Also**

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies all of the items from the collection source to the array dest, starting at the index destIndex.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static void Copy<T>(
    IEnumerable<T> source,
    T[] dest,
    int destIndex
)

Visual Basic (Declaration)

Public Shared Sub Copy(Of T) ( _
    source As IEnumerable(Of T), _
    dest As T(), _
    destIndex As Integer _
)

Visual C++

public:
    generic<typename T>
    static void Copy ( 
        IEnumerable<T>^ source, 
        array<T>^ dest, 
        int destIndex
    )

Parameters

source
    IEnumerable<(Of <T>)>
    The collection that provide the source items.

dest
    array<T>[]()
    The array to store the items into.

destIndex
Int32
The index to begin copying items to.
Type Parameters

T
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentOutOfRangeException</code></td>
<td>destIndex is negative or greater than dest.Length.</td>
</tr>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>source or dest is null. The collection has more items than will fit into the array. In this case, the array has been filled with as many items as fit before the exception is thrown.</td>
</tr>
<tr>
<td><code>System..::ArgumentException</code></td>
<td>destIndex is negative or greater than dest.Length.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::Copy<(Of <T>)> Method (IEnumerable<(Of <T>>), IList<(Of <T>)), Int32, Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies at most count items from the collection source to the list dest, starting at the index destIndex. If necessary, the size of the destination list is expanded. The source collection must not be the destination list or part thereof.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public static void Copy<T>(
    IEnumerable<T> source,
    IList<T> dest,
    int destIndex,
    int count)
```

#### Visual Basic (Declaration)

```vbnet
Public Shared Sub Copy(Of T) (    
    source As IEnumerable(Of T), _
    dest As IList(Of T), _
    destIndex As Integer, _
    count As Integer _
)
```

#### Visual C++

```cpp
public:
    generic<typename T>
    static void Copy (    
        IEnumerable<T>^ source,
        IList<T>^ dest,
        int destIndex,
        int count
    )
```

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| source    | IEnumerable<Of T>  
            | The collection that provide the source items. |
| dest      | IList<Of T>      
            | The list to store the items into. |
destIndex
   Int32
   The index to begin copying items to.

count
   Int32
   The maximum number of items to copy.
Type Parameters

T
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentOutOfRangeException</code></td>
<td>destIndex is negative or greater than dest.Count</td>
</tr>
<tr>
<td><code>System..::ArgumentOutOfRangeException</code></td>
<td>count is negative.</td>
</tr>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>source or dest is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::Copy(Of T) Method (IEnumerable(Of T), array[T][](), Int32, Int32)

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies at most count items from the collection source to the array dest, starting at the index destIndex. The source collection must not be the destination array or part thereof.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static void Copy<T>(
    IEnumerable<T> source,
    T[] dest,
    int destIndex,
    int count
)

Visual Basic (Declaration)

Public Shared Sub Copy(Of T)
    source As IEnumerable(Of T),
    dest As T(),
    destIndex As Integer,
    count As Integer
)

Visual C++

public:
    generic<typename T>
    static void Copy (
        IEnumerable<T>^ source,
        array<T>^ dest,
        int destIndex,
        int count
    )

Parameters

source
    IEnumerable<(Of <T>>)
    The collection that provide the source items.

dest
    array<T>[]()
    The array to store the items into.
destIndex
  Int32
  The index to begin copying items to.

count
  Int32
  The maximum number of items to copy. The array must be large enough to fit this number of items.
Type Parameters

T
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::ArgumentOutOfRangeException</td>
<td>destIndex is negative or greater than dest.Length.</td>
</tr>
<tr>
<td>System..::ArgumentOutOfRangeException</td>
<td>count is negative or destIndex + count is greater than dest.Length.</td>
</tr>
<tr>
<td>System..::ArgumentNullException</td>
<td>source or dest is null.</td>
</tr>
</tbody>
</table>
See Also

* Algorithms Class
* Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::Copy(Of <T>)(Of <T>), Int32, IList(Of <T>), Int32, Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies count items from the list source, starting at the index sourceIndex, to the list dest, starting at the index destIndex. If necessary, the size of the destination list is expanded. The source and destination lists may be the same.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static void Copy<T>(
    IList<T> source,
    int sourceIndex,
    IList<T> dest,
    int destIndex,
    int count
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Sub Copy(Of T) ( _
    source As IList(Of T), _
    sourceIndex As Integer, _
    dest As IList(Of T), _
    destIndex As Integer, _
    count As Integer _
)
```

Visual C++

```cpp
public:
    template<typename T>
    static void Copy ( 
        IList<T>^ source,
        int sourceIndex,
        IList<T>^ dest,
        int destIndex,
        int count
    )
```

Parameters

source
   ```csharp
   IList<(Of <T>)>
   ```
   The collection that provide the source items.

sourceIndex
Int32
The index within source to begin copying items from.

dest
ILList<(Of <T>)>
The list to store the items into.

destIndex
Int32
The index within dest to begin copying items to.

count
Int32
The maximum number of items to copy.
Type Parameters

T
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>sourceIndex is negative or greater than source.Count</td>
</tr>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>destIndex is negative or greater than dest.Count</td>
</tr>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>count is negative or too large.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>source or dest is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::Copy(Of <T>) Method (IList(Of <T>), Int32, array<T>[](), Int32, Int32)

See Also

☐ Visual Basic (Declaration) ☐ Visual Basic (Usage)
☐ C#
☐ Visual C++
☐ J#
☐ JScript
☐ XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies count items from the list or array source, starting at the index sourceIndex, to the array dest, starting at the index destIndex. The source may be the same as the destination array.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public static void Copy<T>(
    IList<T> source,
    int sourceIndex,
    T[] dest,
    int destIndex,
    int count
)
```

### Visual Basic (Declaration)

```vbnet
Public Shared Sub Copy(Of T) ( _
    source As IList(Of T), _
    sourceIndex As Integer, _
    dest As T(), _
    destIndex As Integer, _
    count As Integer _
)
```

### Visual C++

```cpp
public:
    generic<typename T>
    static void Copy ( 
        IList<T>^ source,
        int sourceIndex,
        array<T>^ dest,
        int destIndex,
        int count
    )
```

## Parameters

### source

`IList<Of <T>>`

The list or array that provide the source items.

### sourceIndex
**Int32**
The index within source to begin copying items from.

dest
array<T>[][]
The array to store the items into.

destIndex
**Int32**
The index within dest to begin copying items to.

count
**Int32**
The maximum number of items to copy. The destination array must be large enough to hold this many items.
Type Parameters

T
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::ArgumentOutOfRangeException</td>
<td>sourceIndex is negative or greater than</td>
</tr>
<tr>
<td></td>
<td>source.Count</td>
</tr>
<tr>
<td>System..::ArgumentOutOfRangeException</td>
<td>destIndex is negative or greater than</td>
</tr>
<tr>
<td></td>
<td>dest.Length</td>
</tr>
<tr>
<td>System..::ArgumentOutOfRangeException</td>
<td>count is negative or too large.</td>
</tr>
<tr>
<td>System..::ArgumentNullException</td>
<td>source or dest is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Count the number of items in an IEnumerable<T> collection. If a more specific collection type is being used, it is more efficient to use the Count property, if one is provided.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static int Count<T>(
    IEnumerable<T> collection
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function Count(Of T) ( _
    collection As IEnumerable(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static int Count ( 
        IEnumerable<T>* collection
    )
```

### Parameters

`collection`:
- `IEnumerable<T>`
- The collection to count items in.

### Return Value

The number of items in the collection.
Type Parameters

T
Remarks

If the collection implements ICollection<T>, this method simply returns ICollection<T>.Count. Otherwise, it enumerates all items and counts them.
<table>
<thead>
<tr>
<th><strong>Exception</strong></th>
<th><strong>Condition</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::ArgumentNullException</code></td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
See Also

Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| CountEqual<T>()
(IEnumerable<T>, T) | Counts the number of items in the collection that are equal to find.|
| CountEqual<T>()
(IEnumerable<T>, T, IEqualityComparer<T>) | Counts the number of items in the collection that are equal to find. |
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CountEqual<\(\text{Of } T\)\rangle Method (IEnumerable<\(\text{Of } T\)\rangle, T)

Counts the number of items in the collection that are equal to find.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static int CountEqual<T>(
    IEnumerable<T> collection,
    T find
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function CountEqual(Of T) ( _
    collection As IEnumerable(Of T), _
    find As T _
) As Integer
```

Visual C++

```csharp
public:
    generic<typename T>
    static int CountEqual ( _
        IEnumerable<T>^ collection,
        T find
    )
```

Parameters

collection
    IEnumerable<(Of <T>)>  
        The collection to count items in.

find
    T  
        The item to compare to.

Return Value

The number of items in the collection that are equal to find.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CountEqual<Of T> Method (IEnumerable<Of T>, T, IEqualityComparer<Of T>)

See Also

- Visual Basic (Declaration) - Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Counts the number of items in the collection that are equal to find.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static int CountEqual<T>(
    IEnumerable<T> collection,
    T find,
    IEqualityComparer<T> equalityComparer
)

Visual Basic (Declaration)

Public Shared Function CountEqual(Of T) ( _
    collection As IEnumerable(Of T), _
    find As T, _
    equalityComparer As IEqualityComparer(Of T) _
) As Integer

Visual C++

public:
    generic<typename T>
    static int CountEqual(
        IEnumerable<T>^ collection,
        T find,
        IEqualityComparer<T>^ equalityComparer
    )

Parameters

collection
    IEnumerable<(Of <T>)>
    The collection to count items in.

find
    T
    The item to compare to.

equalityComparer
**IEqualityComparer**<\(\text{Of }\langle T\rangle\)>

The comparer to use to determine if two items are equal. Only the Equals member function will be called.

**Return Value**

The number of items in the collection that are equal to find.
Type Parameters

T
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentException</code></td>
<td>collection or equalityComparer is null.</td>
</tr>
</tbody>
</table>
See Also

* Algorithms Class
  * Wintellect.PowerCollections Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
Algorithms:::CountWhere<Of <T>> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Counts the number of items in the collection that satisfy the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static int CountWhere<T>(
    IEnumerable<T> collection,
    Predicate<T> predicate
)

Visual Basic (Declaration)

Public Shared Function CountWhere(Of T) ( _
    collection As IEnumerable(Of T), _
    predicate As Predicate(Of T) _
) As Integer

Visual C++

public:
    generic<typename T>
    static int CountWhere ( 
        IEnumerable<T>^ collection,
        Predicate<T>^ predicate
    )

Parameters

collection
    IEnumerable<(Of <T>)>GRE
    The collection to count items in.

predicate
    Predicate<(Of <T>)>GRE
    A delegate that defines the condition to check for.

Return Value

The number of items in the collection that satisfy predicate.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::DisjointSets Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DisjointSets</strong>&lt;Of &lt;T&gt;&gt;</td>
<td>Determines if two collections are disjoint, considered as sets. Two sets are disjoint if they have no common items.</td>
</tr>
<tr>
<td><strong>DisjointSets</strong>&lt;Of &lt;T&gt;&gt;&lt;IEnumerable&lt;Of &lt;T&gt;&gt;&gt;, IEnumerable&lt;Of &lt;T&gt;&gt;</td>
<td>Determines if two collections are disjoint, considered as sets. Two sets are disjoint if they have no common items.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::DisjointSets(Of <T>[]) Method (IEnumerable(Of <T>), IEnumerable(Of <T>))

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if two collections are disjoint, considered as sets. Two sets are disjoint if they have no common items.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static bool DisjointSets<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function DisjointSets(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
    generic<typename T>
    static bool DisjointSets ( 
        IEnumerable<T>^ collection1,
        IEnumerable<T>^ collection2
    )
```

Parameters

collection1

```csharp
IEnumerable<Of <T> )
```

The first collection.

collection2

```csharp
IEnumerable<Of <T> )
```

The second collection.

Return Value

True if collection1 are collection2 are disjoint, considered as sets.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the IsDisjoint method on that class.
### Exceptions

<table>
<thead>
<tr>
<th><strong>Exception</strong></th>
<th><strong>Condition</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

**Algorithms** Class
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms...:::DisjointSets(Of T) Method (IEnumerable(Of T),
IEnumerable(Of T), IEquityComparer(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if two collections are disjoint, considered as sets. Two sets are disjoint if they have no common items.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static bool DisjointSets<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2,
    IEqualityComparer<T> equalityComparer
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function DisjointSets(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
    generic<typename T>
    static bool DisjointSets ( 
        IEnumerable<T>^ collection1, 
        IEnumerable<T>^ collection2, 
        IEqualityComparer<T>^ equalityComparer
    )
```

Parameters

collection1
    IEnumerable<(Of <T>)><Of Resultset>()
    The first collection.

collection2
    IEnumerable<(Of <T>)><Of Resultset>()
    The second collection.

equalityComparer
**IEqualityComparer\<(\<T\>)\>**

The IEqualityComparer\<\<T\>> used to compare items for equality. Only the Equals and GetHashCode member functions of this interface are called.

**Return Value**

True if collection1 are collection2 are disjoint, considered as sets.
Type Parameters

T
Remarks

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the IsDisjoint method on that class.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentNullException</td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::EqualCollections Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>EqualCollections(Of T)&gt;(IEnumerable(Of T&gt;, IEnumerable(Of T&gt;)</code></td>
<td>Determines if the two collections contain equal items in the same order. The two collections do not need to be of the same type; it is permissible to compare an array and an <code>OrderedBag</code>, for instance.</td>
</tr>
<tr>
<td><code>EqualCollections(Of T)&gt;(IEnumerable(Of T&gt;, IEnumerable(Of T&gt;</code>, <code>IEqualityComparer(Of T)&gt;)</code></td>
<td>Determines if the two collections contain equal items in the same order. The passed instance of <code>IEqualityComparer&lt;T&gt;</code> is used for determining if two items are equal.</td>
</tr>
<tr>
<td><code>EqualCollections(Of T)&gt;(IEnumerable(Of T&gt;, IEnumerable(Of T&gt;</code>, <code>BinaryPredicate(Of T)&gt;)</code></td>
<td>Determines if the two collections contain &quot;equal&quot; items in the same order. The passed <code>BinaryPredicate</code> is used to determine if two items are &quot;equal&quot;.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if the two collections contain equal items in the same order. The two collections do not need to be of the same type; it is permissible to compare an array and an OrderedBag, for instance.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static bool EqualCollections<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function EqualCollections(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T) _
) As Boolean
```

Visual C++

```csharp
public:
    generic<typename T>
    static bool EqualCollections ( 
        IEnumerable<T>^ collection1,
        IEnumerable<T>^ collection2
    )
```

Parameters

collection1

```csharp
IEnumerable<(Of <T>)>
```
The first collection to compare.

collection2

```csharp
IEnumerable<(Of <T>)>
```
The second collection to compare.

Return Value

True if the collections have equal items in the same order. If both collections are
empty, true is returned.
**Type Parameters**

T

The type of items in the collections.
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.
See Also

**Algorithms** Class
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
Determines if the two collections contain equal items in the same order. The passed instance of IEqualityComparer\(\langle T\rangle\) is used for determining if two items are equal.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static bool EqualCollections<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2,
    IEqualityComparer<T> equalityComparer)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function EqualCollections(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
) As Boolean
```

**Visual C++**

```cpp
generic<typename T>
public:
static bool EqualCollections ( _
    IEnumerable<T>^ collection1,
    IEnumerable<T>^ collection2,
    IEqualityComparer<T>^ equalityComparer
)
```

**Parameters**

collection1
    IEnumerable<(Of <T>)> The first collection to compare.

collection2
    IEnumerable<(Of <T>)> The second collection to compare.

equalityComparer
IEqualityComparer<Of <T> >
The IEqualityComparer<T> used to compare items for equality. Only the Equals member function of this interface is called.

Return Value

True if the collections have equal items in the same order. If both collections are empty, true is returned.
Type Parameters

T
The type of items in the collections.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>collection1, collection2, or equalityComparer is null.</td>
</tr>
</tbody>
</table>
See Also

[Algorithm] Class
[Wintellect.PowerCollections] Namespace

Send [comments] about this topic to Microsoft.
Algorithms...:::EqualCollections<(Of <T>)> Method (IEnumerable<(Of <T>)>,
IEnumerable<(Of <T>)>, BinaryPredicate<(Of <T>)>)

See Also

[(Visual Basic (Declaration)](Visual Basic (Declaration)) [Visual Basic (Usage)]
[(C#)](C#)
[(Visual C++)](Visual C++)
[(J#)](J#)
[(JScript)](JScript)
[(XAML)](XAML)

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if the two collections contain "equal" items in the same order. The passed BinaryPredicate is used to determine if two items are "equal".

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static bool EqualCollections<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2,
    BinaryPredicate<T> predicate
)

Visual Basic (Declaration)

Public Shared Function EqualCollections(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T), _
    predicate As BinaryPredicate(Of T) _
) As Boolean

Visual C++

public:
    generic<typename T>
    static bool EqualCollections ( 
        IEnumerable<T>^ collection1, 
        IEnumerable<T>^ collection2, 
        BinaryPredicate<T>^ predicate 
    )

Parameters

collection1
    IEnumerable<(Of <T>)> 
    The first collection to compare.

collection2
    IEnumerable<(Of <T>)> 
    The second collection to compare.

predicate
**BinaryPredicate<(Of <T>))**

The BinaryPredicate used to compare items for "equality". This predicate can compute any relation between two items; it need not represent equality or an equivalence relation.

**Return Value**

True if predicate returns true for each corresponding pair of items in the two collections. If both collections are empty, true is returned.
Type Parameters

T
The type of items in the collections.
Remarks

Since an arbitrary BinaryPredicate is passed to this function, what is being tested for need not be equality. For example, the following code determines if each integer in list1 is less than or equal to the corresponding integer in list2.

```csharp
List<int> list1, list2;
if (EqualCollections(list1, list2, delegate(int x, int y)
    // the check is true...
}
```
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>collection1, collection2, or predicate is null.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](Wintellect.PowerCollections) Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms:::EqualSets Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EqualSets&lt;Of T&gt;</td>
<td>Determines if two collections are equal, considered as sets. Two sets are equal if they have the same items, with order not being significant.</td>
</tr>
<tr>
<td>(IEnumerable&lt;Of T&gt;, IEnumerable&lt;Of T&gt;)</td>
<td></td>
</tr>
<tr>
<td>EqualSets&lt;Of T&gt;</td>
<td>Determines if two collections are equal, considered as sets. Two sets are equal if they have the same items, with order not being significant.</td>
</tr>
<tr>
<td>(IEnumerable&lt;Of T&gt;, IEqualityComparer&lt;Of T&gt;)</td>
<td></td>
</tr>
</tbody>
</table>
See Also

[Algorithms](#) Class
[Algorithms](#) Members
[Winintellect.PowerCollections](#) Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms...:::EqualSets<Of <T>> Method (IEnumerable<Of <T>>, IEnumerable<Of <T>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if two collections are equal, considered as sets. Two sets are equal if they have the same items, with order not being significant.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static bool EqualSets<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function EqualSets(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T) _
) As Boolean
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static bool EqualSets ( 
        IEnumerable<T>^ collection1, 
        IEnumerable<T>^ collection2
    )
```

**Parameters**

collection1
   `IEnumerable<Of <T>>`
   The first collection.

collection2
   `IEnumerable<Of <T>>`
   The second collection.

**Return Value**

True if collection1 are collection2 are equal, considered as sets.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the EqualTo method on that class.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms..:::EqualSets(Of <T>), IEnumerable(Of <T>), IEnumerable(Of <T>), IEquityComparer(Of <T>)}

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if two collections are equal, considered as sets. Two sets are equal if they have the same items, with order not being significant.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

**C#**

```csharp
public static bool EqualSets<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2,
    IEqualityComparer<T> equalityComparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function EqualSets(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
) As Boolean
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static bool EqualSets ( 
        IEnumerable<T>^ collection1, 
        IEnumerable<T>^ collection2, 
        IEqualityComparer<T>^ equalityComparer 
    )
```

## Parameters

**collection1**

- `IEnumerable<(Of <T>)>`
  - The first collection.

**collection2**

- `IEnumerable<(Of <T>)>`
  - The second collection.

**equalityComparer**
**IEqualityComparer**< Of <T> >
The IEqualityComparer<T> used to compare items for equality. Only the Equals and GetHashCode member functions of this interface are called.

**Return Value**

True if collection1 are collection2 are equal, considered as sets.
Type Parameters

T
Remarks

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the EqualTo method on that class.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::ArgumentNullException</code></td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#)
[WinTellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Algorithms...:::Exists(Of T) Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if a collection contains any item that satisfies the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static bool Exists<T>(
    IEnumerable<T> collection,
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function Exists(Of T) (_
    collection As IEnumerable(Of T), _
    predicate As Predicate(Of T) _
) As Boolean
```

**Visual C++**

```csharp
public:
    generic<typename T>
    static bool Exists ( 
        IEnumerable<T>^ collection, 
        Predicate<T>^ predicate 
    )
```

**Parameters**

- **collection**
  ```csharp
  IEnumerable<(Of <T>)> 
  ```
  The collection to check all the items in.

- **predicate**
  ```csharp
  Predicate<(Of <T>)> 
  ```
  A delegate that defines the condition to check for.

**Return Value**

True if the collection contains one or more items that satisfy the condition
defined by predicate. False if the collection does not contain an item that satisfies predicate.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill&lt;(Of &lt;T&gt;), (IList&lt;(Of &lt;T&gt;), T))</td>
<td>Replaces each item in a list with a given value. The list does not change in size.</td>
</tr>
<tr>
<td>Fill&lt;(Of &lt;T&gt;), (array&lt;T&gt;<a href=""></a>, T))</td>
<td>Replaces each item in an array with a given value.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#)
[Algorithms Members](#)
[Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Replaces each item in a list with a given value. The list does not change in size.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
# Syntax

**C#**

```csharp
public static void Fill<T>(
    IList<T> list,
    T value
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Sub Fill(Of T) (_
    list As IList(Of T), _
    value As T _
)
```

**Visual C++**

```cpp
public:
    template<typename T>
    static void Fill ( 
        IList<T>^ list, 
        T value 
    )
```

# Parameters

**list**
- `IList<Of <T>>`
  - The list to modify.

**value**
- `T`
  - The value to fill with.
Type Parameters

T

The type of items in the list.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentException</code></td>
<td>list is a read-only list.</td>
</tr>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>list is null.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#)
[**Wintellect.PowerCollections** Namespace](#)

Send [comments](#) about this topic to Microsoft.
Replaces each item in an array with a given value.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static void Fill<T>(
    T[] array,
    T value
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Sub Fill(Of T) ( _
    array As T(), _
    value As T _
)
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static void Fill ( 
        array<T>^ array,
        T value
    )
```

### Parameters

**array**

```csharp
array<T>[]()
```

The array to modify.

**value**

```csharp
T
```

The value to fill with.
Type Parameters

T
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentNullException</td>
<td>array is null.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#)
[WinELect.PowerCollections](#) Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms...::FillRange Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FillRange&lt;(Of &lt;T&gt;),(IList&lt;(Of &lt;T&gt;), Int32, Int32, T)&gt;</td>
<td>Replaces each item in a part of a list with a given value.</td>
</tr>
<tr>
<td>FillRange&lt;(Of &lt;T&gt;),(array&lt;T&gt;[]), Int32, Int32, T)</td>
<td>Replaces each item in a part of an array with a given value.</td>
</tr>
</tbody>
</table>
See Also

- [Algorithms Class](#)
- [Algorithms Members](#)
- [Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Algorithms...:::FillRange<(Of <T>)> Method (IList<(Of <T>)>, Int32, Int32, T)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Replaces each item in a part of a list with a given value.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static void FillRange<T>(
    IList<T> list,
    int start,
    int count,
    T value
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Sub FillRange(Of T) ( _
    list As IList(Of T), _
    start As Integer, _
    count As Integer, _
    value As T _
)
```

Visual C++

```cpp
public:
    generic<typename T>
    static void FillRange ( 
        IList<T>^ list,
        int start,
        int count,
        T value
    )
```

Parameters

list

```csharp
IList<(Of <T>)>
```

The list to modify.

start

```csharp
Int32
```

The index at which to start filling. The first index in the list has index 0.
count
  \texttt{Int32}
  The number of items to fill.

value
  \texttt{T}
  The value to fill with.
Type Parameters

T

The type of items in the list.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentException</td>
<td>list is a read-only list.</td>
</tr>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>start or count is negative, or start + count is greater than list.Count.</td>
</tr>
<tr>
<td>System::ArgumentNullException</td>
<td>list is null.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class]
[WinTellecT.PowerColecTions Namespace]

Send comments about this topic to Microsoft.
Algorithms...:::FillRange(Of T>) Method (array<T>[], Int32, Int32, T)

**See Also**
- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Replaces each item in a part of a array with a given value.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static void FillRange<T>(
    T[,] array,
    int start,
    int count,
    T value
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Sub FillRange(Of T) ( _
    array As T(), _
    start As Integer, _
    count As Integer, _
    value As T _
)
```

**Visual C++**

```cpp
public:
    template<typename T>
    static void FillRange (
        array<T>^ array,
        int start,
        int count,
        T value
    )
```

**Parameters**

- array  
  array<T>[][]
  The array to modify.

- start  
  `int`  
  The index at which to start filling. The first index in the array has index 0.
count

\texttt{Int32}

The number of items to fill.

value

\texttt{T}

The value to fill with.
Type Parameters

T
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>start or count is negative, or start + count is greater than array.Length.</td>
</tr>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>array is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Winellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::FindFirstIndexWhere<Of <T>> Method

See Also

Visual Basic (Declaration) ▶ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the first item in a list that satisfies the condition defined by predicate.

**Namespace:** Wintellect.POWERCOLLECTIONS

**Assembly:** PowerCollections (in PowerCollections.dll)
C#

```csharp
public static int FindFirstIndexWhere<T>(
    IList<T> list,
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function FindFirstIndexWhere(Of T) ( _
    list As IList(Of T), _
    predicate As Predicate(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:

generic<typename T>

static int FindFirstIndexWhere ( 
    IList<T>^ list,
    Predicate<T>^ predicate
)
```

**Parameters**

- **list**
  - `IList<Of <T>>`
  - The list to search.

- **predicate**
  - `Predicate<Of <T>>`
  - A delegate that defined the condition to check for.

**Return Value**

The index of the first item satisfying the condition. -1 if no such item exists in
the list.
- Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::FindFirstWhere(Of <T>)> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the first item in a collection that satisfies the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public static T FindFirstWhere<T>(
    IEnumerable<T> collection,
    Predicate<T> predicate
)
```

### Visual Basic (Declaration)

```vbnet
Public Shared Function FindFirstWhere(Of T) ( _
    collection As IEnumerable(Of T), _
    predicate As Predicate(Of T) _
) As T
```

### Visual C++

```cpp
public:
    generic<typename T>
    static T FindFirstWhere ( 
        IEnumerable<T>^ collection,
        Predicate<T>^ predicate
    )
```

### Parameters

- **collection**
  ```csharp
  IEnumerable<T> collection,
  ```
  The collection to search.

- **predicate**
  ```csharp
  Predicate<T> predicate
  ```
  A delegate that defined the condition to check for.

### Return Value

The first item in the collection that matches the condition, or the default value.
for T (0 or null) if no item that matches the condition is found.
| Type Parameters | \( T \) |
Remarks

If the default value for T could be present in the collection, and would be matched by the predicate, then this method is inappropriate, because you cannot distinguish whether the default value for T was actually present in the collection, or no items matched the predicate. In this case, use TryFindFirstWhere.
See Also

**Algorithms** Class  
**Wintellect.PowerCollections** Namespace  
**Algorithms...::TryFindFirstWhere(Of <T>)(IEnumerable(Of <T>), Predicate(Of <T>), T%)**

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
FindIndicesWhere<(Of <T>)> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerates the indices of all the items in list that satisfy the condition defined by predicate.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEnumerable<int> FindIndicesWhere<T>(
    IList<T> list,
    Predicate<T> predicate
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function FindIndicesWhere(Of T) ( _
    list As IList(Of T), _
    predicate As Predicate(Of T) _
) As IEnumerable(Of Integer)
```

Visual C++

```csharp
public:
    generic<typename T>
    static IEnumerable<int>^ FindIndicesWhere ( 
        IList<T>^ list,
        Predicate<T>^ predicate
    )
```

Parameters

list
   IList<(Of <T>)>
   The list to check all the items in.

predicate
   Predicate<(Of <T>)>
   A delegate that defines the condition to check for.

Return Value

An IEnumerable<T> that enumerates the indices of items that satisfy the
condition.
Type Parameters

T
See Also

* Algorithms Class
  * Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
FindLastIndexWhere<(Of <T>)> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.

Finds the index of the last item in a list that satisfies the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static int FindLastIndexWhere<T>(
    IList<T> list,
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function FindLastIndexWhere(Of T) ( _
    list As IList(Of T), _
    predicate As Predicate(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static int FindLastIndexWhere ( 
        IList<T>^ list,
        Predicate<T>^ predicate
    )
```

### Parameters

**list**

```csharp
IList<T>
```

The list to search.

**predicate**

```csharp
Predicate<T>
```

A delegate that defined the condition to check for.

### Return Value

The index of the last item satisfying the condition. -1 if no such item exists in the
list.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::FindLastWhere<Of <T>> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the last item in a collection that satisfies the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static T FindLastWhere<T>(
    IEnumerable<T> collection,
    Predicate<T> predicate
)

Visual Basic (Declaration)

Public Shared Function FindLastWhere(Of T) ( _
    collection As IEnumerable(Of T), _
    predicate As Predicate(Of T) _) As T

Visual C++

public:
    generic<typename T>
    static T FindLastWhere ( 
        IEnumerable<T>^ collection,
        Predicate<T>^ predicate
    )

Parameters

collection
    IEnumerable<(Of <T>)>
    The collection to search.

predicate
    Predicate<(Of <T>)>
    A delegate that defined the condition to check for.

Return Value

The last item in the collection that matches the condition, or the default value for
T (0 or null) if no item that matches the condition is found.
Type Parameters

T
Remarks

If the collection implements IList<T>, then the list is scanned in reverse until a matching item is found. Otherwise, the entire collection is iterated in the forward direction.

If the default value for T could be present in the collection, and would be matched by the predicate, then this method is inappropriate, because you cannot distinguish whether the default value for T was actually present in the collection, or no items matched the predicate. In this case, use TryFindFirstWhere.
See Also

* Algorithms Class
* Wintellect.PowerCollections Namespace
* Algorithms...::TryFindLastWhere<(Of <T>))(IEnumerable<(Of <T>)), Predicate<(Of <T>)), T%)

Send [comments](#) about this topic to Microsoft.
Enumerates all the items in collection that satisfy the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static IEnumerable<T> FindWhere<T>(
    IEnumerable<T> collection,
    Predicate<T> predicate
)

Visual Basic (Declaration)

Public Shared Function FindWhere(Of T) ( _
    collection As IEnumerable(Of T), _
    predicate As Predicate(Of T) _
) As IEnumerable(Of T)

Visual C++

public:
    generic<typename T>
    static IEnumerable<T>^ FindWhere ( 
        IEnumerable<T>^ collection,
        Predicate<T>^ predicate
    )

Parameters

collection
    IEnumerable<
        Of<T>>
    The collection to check all the items in.

predicate
    Predicate<
        Of<T>>
    A delegate that defines the condition to check for.

Return Value

An IEnumerable<T> that enumerates the items that satisfy the condition.
**Type Parameters**

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::FirstConsecutiveEqual Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FirstConsecutiveEqual&lt;(Of&lt;T&gt;&gt;(IList&lt;(Of&lt;T&gt;&gt;), Int32)</td>
<td>Finds the first occurrence of count consecutive equal items in the list.</td>
</tr>
<tr>
<td>FirstConsecutiveEqual&lt;(Of&lt;T&gt;&gt;(IList&lt;(Of&lt;T&gt;&gt;), Int32, IEqualityComparer&lt;(Of&lt;T&gt;&gt;))</td>
<td>Finds the first occurrence of count consecutive equal items in the list. A passed IEqualityComparer is used to determine equality.</td>
</tr>
<tr>
<td>FirstConsecutiveEqual&lt;(Of&lt;T&gt;&gt;(IList&lt;(Of&lt;T&gt;&gt;), Int32, BinaryPredicate&lt;(Of&lt;T&gt;&gt;))</td>
<td>Finds the first occurrence of count consecutive &quot;equal&quot; items in the list. The passed BinaryPredicate is used to determine if two items are &quot;equal&quot;.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::FirstConsecutiveEqual<Of <T>> Method (IList<Of <T>>, Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the first occurrence of count consecutive equal items in the list.

**Namespace**: Wintellect.PowerCollections

**Assembly**: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static int FirstConsecutiveEqual<T>(
    IList<T> list,
    int count
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function FirstConsecutiveEqual(Of T) ( _
    list As IList(Of T), _
    count As Integer _
) As Integer
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static int FirstConsecutiveEqual (
        IList<T>* list,
        int count
    )
```

### Parameters

**list**

`IList<Of <T>>`

The list to examine.

**count**

`Int32`

The number of consecutive equal items to look for. The count must be at least 1.

### Return Value
The index of the first item in the first run of count consecutive equal items, or -1 if no such run exists..
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::FirstConsecutiveEqual<(Of <T>)> Method (IList<(Of <T>)> , Int32, IEqualityComparer<(Of <T>)>)

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the first occurrence of count consecutive equal items in the list. A passed IEqualityComparer is used to determine equality.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static int FirstConsecutiveEqual<T>(
    IList<T> list,
    int count,
    IEqualityComparer<T> equalityComparer
)

Visual Basic (Declaration)

Public Shared Function FirstConsecutiveEqual(Of T) ( _
    list As IList(Of T), _
    count As Integer, _
    equalityComparer As IEqualityComparer(Of T) _
) As Integer

Visual C++

class contains
    public:
        template<
            typename T>
        static int FirstConsecutiveEqual ( _
            IList<T>* list,
            int count,
            IEqualityComparer<T>* equalityComparer
        )

Parameters

list
    IList<Of <T>>>
    The list to examine.

count
    Int32
    The number of consecutive equal items to look for. The count must be at least 1.
equalityComparer

`IEqualityComparer<T>`

The `IEqualityComparer<T>` used to compare items for equality. Only the `Equals` method will be called.

**Return Value**

The index of the first item in the first run of count consecutive equal items, or -1 if no such run exists.
Type Parameters

T
See Also

**Algorithms** Class

**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
FirstConsecutiveEqual<
(OfType)>
Method (IList<
(OfType)>,
Int32,
BinaryPredicate<
(OfType)>)

Finds the first occurrence of count consecutive "equal" items in the list. The passed BinaryPredicate is used to determine if two items are "equal".

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static int FirstConsecutiveEqual<T>(
    IList<T> list,
    int count,
    BinaryPredicate<T> predicate
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function FirstConsecutiveEqual(Of T) ( _
    list As IList(Of T), _
    count As Integer, _
    predicate As BinaryPredicate(Of T) _
) As Integer
```

Visual C++

```cpp
public:
    generic<typename T>
    static int FirstConsecutiveEqual ( 
        IList<T>^ list,
        int count,
        BinaryPredicate<T>^ predicate
    )
```

Parameters

**list**

```csharp
IList<Of <T>>
```

The list to examine.

**count**

```csharp
int
```

The number of consecutive equal items to look for. The count must be at least 1.
predicate

\texttt{BinaryPredicate<\text{Of} \ <T\,>)}\n
The BinaryPredicate used to compare items for "equality".

\textbf{Return Value}

The index of the first item in the first run of count consecutive equal items, or -1 if no such run exists.
Type Parameters

T
Remarks

Since an arbitrary BinaryPredicate is passed to this function, what is being tested for need not be true equality.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...::FirstConsecutiveWhere<Of <T>> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the first occurrence of count consecutive items in the list for which a given predicate returns true.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static int FirstConsecutiveWhere<T>(
    IList<T> list,
    int count,
    Predicate<T> predicate
)

Visual Basic (Declaration)

Public Shared Function FirstConsecutiveWhere(Of T) ( _
    list As IList(Of T), _
    count As Integer, _
    predicate As Predicate(Of T) _
) As Integer

Visual C++

public:
    generic<typename T>
    static int FirstConsecutiveWhere ( 
        IList<T>^ list,
        int count,
        Predicate<T>^ predicate
    )

Parameters

list
    IList<Of <T>>>
    The list to examine.

count
    Int32
    The number of consecutive items to look for. The count must be at least 1.

predicate
**Predicate<**(Of <T>**)>**

The predicate used to test each item.

**Return Value**

The index of the first item in the first run of count items where predicate returns true for all items in the run, or -1 if no such run exists.
Type Parameters

T
See Also

[Algorithms Class](#)  
[Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Algorithms:::FirstIndexOf Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FirstIndexOf(Of T)&gt;(IList(Of T), T)</td>
<td>Finds the index of the first item in a list equal to a given item.</td>
</tr>
<tr>
<td>FirstIndexOf(Of T)&gt;(IList(Of T), T, IEqualityComparer(Of T))</td>
<td>Finds the index of the first item in a list equal to a given item. A passed IEqualityComparer is used to determine equality.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::FirstIndexOf<Of <T>> Method (IList<Of <T>>, T)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the first item in a list equal to a given item.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static int FirstIndexOf<T>(
    IList<T> list,
    T item
)

Visual Basic (Declaration)

Public Shared Function FirstIndexOf(Of T) ( _
    list As IList(Of T), _
    item As T _
) As Integer

Visual C++

public:
    generic<typename T>
    static int FirstIndexOf ( 
        IList<T>^ list,
        T item
    )

Parameters

list
    IList<(Of <T>)> 
    The list to search.

item
    T 
    The item to search for.

Return Value

The index of the first item equal to item. -1 if no such item exists in the list.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.
See Also

*Algorithms* Class  
*Wintellect.PowerCollections* Namespace

Send [comments](#) about this topic to Microsoft.
Finds the index of the first item in a list equal to a given item. A passed IEqualityComparer is used to determine equality.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static int FirstIndexOf<T>(
    IList<T> list,
    T item,
    IEqualityComparer<T> equalityComparer
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function FirstIndexOf(Of T) ( _
    list As IList(Of T), _
    item As T, _
    equalityComparer As IEqualityComparer(Of T) _
) As Integer
```

Visual C++

```cpp
public:
    generic<typename T>
    static int FirstIndexOf ( 
        IList<T>* list,
        T item,
        IEqualityComparer<T>* equalityComparer
    )
```

Parameters

- **list**
  - `IList<Of <T>>`
  - The list to search.

- **item**
  - `T`
  - The item to search for.

- **equalityComparer**
**IEqualityComparer<Of <T>>**
The IEqualityComparer<T> used to compare items for equality. Only the Equals method will be called.

**Return Value**

The index of the first item equal to item. -1 if no such item exists in the list.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::FirstIndexOfMany Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FirstIndexOfMany(Of </code>&lt;T&gt;<code>)</code>&lt;br&gt;<code>(IList(Of </code>&lt;T&gt;<code>), IEnumerable(Of </code>&lt;T&gt;<code>))</code>&lt;br&gt;<code>S</code></td>
<td>Finds the index of the first item in a list equal to one of several given items.</td>
</tr>
<tr>
<td><code>FirstIndexOfMany(Of </code>&lt;T&gt;<code>)</code>&lt;br&gt;<code>(IList(Of </code>&lt;T&gt;<code>), IEnumerable(Of </code>&lt;T&gt;<code>), IEqualityComparer(Of </code>&lt;T&gt;<code>)</code>)<code>&lt;br&gt;</code>S`</td>
<td>Finds the index of the first item in a list equal to one of several given items. A passed IEqualityComparer is used to determine equality.</td>
</tr>
<tr>
<td><code>FirstIndexOfMany(Of </code>&lt;T&gt;<code>)</code>&lt;br&gt;<code>(IList(Of </code>&lt;T&gt;<code>), IEnumerable(Of </code>&lt;T&gt;<code>), BinaryPredicate(Of </code>&lt;T&gt;<code>)</code>)<code>&lt;br&gt;</code>S`</td>
<td>Finds the index of the first item in a list &quot;equal&quot; to one of several given items. The passed BinaryPredicate is used to determine if two items are &quot;equal&quot;.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms..:::FirstIndexOfMany<(Of <T>),> Method (IList<(Of <T>),>, IEnumerable<(Of <T>),>)

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the first item in a list equal to one of several given items.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static int FirstIndexOfMany<T>(
    IList<T> list,
    IEnumerable<T> itemsToLookFor
)

Visual Basic (Declaration)

Public Shared Function FirstIndexOfMany(Of T) ( _
    list As IList(Of T), _
    itemsToLookFor As IEnumerable(Of T) _
) As Integer

Visual C++

public:

generic<typename T>

static int FirstIndexOfMany ( 
    IList<T>^ list, 
    IEnumerable<T>^ itemsToLookFor
)

Parameters

list
    IList<(Of <T>)> The list to search.

itemsToLookFor
    IEnumerable<(Of <T>)> The items to search for.

Return Value

The index of the first item equal to any of the items in the collection
itemsToLookFor. -1 if no such item exists in the list.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
FirstIndexOfMany<
(Of <T>)>
Method (IList<
(Of <T>)>,
IEnumerable<
(Of <T>)>,
IEqualityComparer<
(Of <T>)>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future
releases. Blank topics are included as placeholders.]

Finds the index of the first item in a list equal to one of several given items. A
passed IEqualityComparer is used to determine equality.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public static int FirstIndexOfMany<T>(
    IList<T> list,
    IEnumerable<T> itemsToLookFor,
    IEqualityComparer<T> equalityComparer
)
```

#### Visual Basic (Declaration)

```vbnet
Public Shared Function FirstIndexOfMany(Of T) (_
    list As IList(Of T), _
    itemsToLookFor As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
) As Integer
```

#### Visual C++

```cpp
public:
    generic<typename T>
    static int FirstIndexOfMany ( 
        IList<T>* list,
        IEnumerable<T>* itemsToLookFor,
        IEqualityComparer<T>* equalityComparer
    )
```

### Parameters

- **list**
  - `IList<Of <T>>`
  - The list to search.

- **itemsToLookFor**
  - `IEnumerable<Of <T>>`
  - The items to search for.

- **equalityComparer**
**IEqualityComparer**

The `IEqualityComparer<Of <T>>` used to compare items for equality. Only the `Equals` and `GetHashCode` methods will be called.

**Return Value**

The index of the first item equal to any of the items in the collection `itemsToLookFor`. -1 if no such item exists in the list.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Finds the index of the first item in a list "equal" to one of several given items. The passed BinaryPredicate is used to determine if two items are "equal".

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public static int FirstIndexOfMany<T>(
    IList<T> list,
    IEnumerable<T> itemsToLookFor,
    BinaryPredicate<T> predicate
)
```

### Visual Basic (Declaration)

```vbnet
Public Shared Function FirstIndexOfMany(Of T) ( _
    list As IList(Of T), _
    itemsToLookFor As IEnumerable(Of T), _
    predicate As BinaryPredicate(Of T) _
) As Integer
```

### Visual C++

```cpp
public:
    generic<typename T>
    static int FirstIndexOfMany (
        IList<T>* list,
        IEnumerable<T>* itemsToLookFor,
        BinaryPredicate<T>* predicate
    )
```

## Parameters

**list**
- `IList<Of <T>>`
  - The list to search.

**itemsToLookFor**
- `IEnumerable<Of <T>>`
  - The items to search for.

**predicate**
- `BinaryPredicate<Of <T>>`
  - The predicate to use for searching.
**BinaryPredicate**<Of <T> >
The BinaryPredicate used to compare items for "equality".

**Return Value**

The index of the first item "equal" to any of the items in the collection `itemsToLookFor`, using as the test for equality. -1 if no such item exists in the list.
Type Parameters

T
Remarks

Since an arbitrary BinaryPredicate is passed to this function, what is being removed need not be true equality. This methods finds first item X which satisfies BinaryPredicate(X,Y), where Y is one of the items in itemsToLookFor.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::forEach<Of <T>> Method

See Also

- Visual Basic (Declaration) - Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Performs the specified action on each item in a collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static void ForEach<T>(
    IEnumerable<T> collection,
    Action<T> action
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Sub ForEach(Of T) ( _
    collection As IEnumerable(Of T), _
    action As Action(Of T) _
)
```

Visual C++

```cpp
public:
    generic<typename T>
    static void ForEach ( 
        IEnumerable<T>^ collection, 
        Action<T>^ action 
    )
```

Parameters

collection

- `IEnumerable<T>`
  - The collection to process.

action

- `Action<T>`
  - An Action delegate which is invoked for each item in collection.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Generates all the possible permutations of the items in collection. If collection has N items, then N factorial permutations will be generated. This method does not compare the items to determine if any of them are equal. If some items are equal, the same permutation may be generated more than once. For example, if the collections contains the three items A, A, and B, then this method will generate the six permutations, AAB, AAB, ABA, ABA, BAA, BAA (not necessarily in that order). To take equal items into account, use the GenerateSortedPermutations method.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static IEnumerable<T[]> GeneratePermutations<T>(
    IEnumerable<T> collection
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function GeneratePermutations(Of T) ( _
    collection As IEnumerable(Of T) _
) As IEnumerable(Of T())
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static IEnumerable<Array<T>^>^ GeneratePermutations ( 
    IEnumerable<T>^ collection
    )
```

### Parameters

**collection**

*IEnumerable<Of <T>>*

The collection of items to permute.

### Return Value

An IEnumerable<T[]> that enumerations all the possible permutations of the items in collection. Each permutations is returned as an array. The items in the array should be copied if they need to be used after the next permutation is generated; each permutation may reuse the same array instance.
Type Parameters

T

The type of items to permute.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::GenerateSortedPermutations Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenerateSortedPermutations&lt;Of T&gt;(IEnumerable&lt;Of T&gt;)</td>
<td>Generates all the possible permutations of the items in collection, in lexicographical order. Even if some items are equal, the same permutation will not be generated more than once. For example, if the collections contains the three items A, A, and B, then this method will generate only the three permutations, AAB, ABA, BAA.</td>
</tr>
<tr>
<td>GenerateSortedPermutations&lt;Of T&gt;(IEnumerable&lt;Of T&gt;, IComparer&lt;Of T&gt;)</td>
<td>Generates all the possible permutations of the items in collection, in lexicographical order. A supplied IComparer&lt;T&gt; instance is used to compare the items. Even if some items are equal, the same permutation will not be generated more than once. For example, if the collections contains the three items A, A, and B, then this method will generate only the three permutations, AAB, ABA, BAA.</td>
</tr>
<tr>
<td>GenerateSortedPermutations&lt;Of T&gt;(IEnumerable&lt;Of T&gt;, Comparison&lt;Of T&gt;)</td>
<td>Generates all the possible permutations of the items in collection, in lexicographical order. A supplied Comparison&lt;T&gt; delegate is used to compare the items. Even if some items are equal, the same permutation will not be generated more than once. For example, if the collections contains the three items A, A, and B, then this method will generate only the three permutations, AAB, ABA, BAA.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
GenerateSortedPermutations<(Of <T>)> Method
(IEnumerable<(Of <T>)>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

Generates all the possible permutations of the items in collection, in lexicographical order. Even if some items are equal, the same permutation will not be generated more than once. For example, if the collections contains the three items A, A, and B, then this method will generate only the three permutations, AAB, ABA, BAA.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static IEnumerable<T[]> GenerateSortedPermutations<T>(
    IEnumerable<T> collection
) where T : Object, IComparable<T>
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function GenerateSortedPermutations(Of T As {Object, } collection As IEnumerable(Of T) _
) As IEnumerable(Of T())
```

**Visual C++**

```cpp
public:
    generic<typename T>
    where T : Object, IComparable<T>
    static IEnumerable<array<T>^>^ GenerateSortedPermutations ( 
        IEnumerable<T>^ collection
    )
```

**Parameters**

- `collection` 
  - `IEnumerable<(Of <T>)>`
  - The collection of items to permute.

**Return Value**

An `IEnumerable<T[]>` that enumerations all the possible permutations of the items in `collection`. Each permutations is returned as an array. The items in the array should be copied if they need to be used after the next permutation is generated; each permutation may reuse the same array instance.
Type Parameters

T

The type of items to permute.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::GenerateSortedPermutations< Of< T >> Method
(IEnumerable< Of< T >>, IComparer< Of< T >>)

See Also

- Visual Basic (Declaration) - Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Generates all the possible permutations of the items in collection, in lexicographical order. A supplied IComparer< T > instance is used to compare the items. Even if some items are equal, the same permutation will not be generated more than once. For example, if the collections contains the three items A, A, and B, then this method will generate only the three permutations, AAB, ABA, BAA.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static IEnumerable<T[]> GenerateSortedPermutations<T>(
    IEnumerable<T> collection,
    IComparer<T> comparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function GenerateSortedPermutations(Of T) ( _
    collection As IEnumerable(Of T), _
    comparer As IComparer(Of T) _
) As IEnumerable(Of T())
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static IEnumerable<array<T>^>^ GenerateSortedPermutations ( 
        IEnumerable<T>^ collection,
        IComparer<T>^ comparer
    )
```

**Parameters**

- `collection`: `IEnumerable<Of <T>>`
  - The collection of items to permute.

- `comparer`: `IComparer<Of <T>>`
  - The IComparer<T> used to compare the items.

**Return Value**

An IEnumerable<T[]> that enumerations all the possible permutations of the
items in collection. Each permutations is returned as an array. The items in the array should be copied if they need to be used after the next permutation is generated; each permutation may reuse the same array instance.
Type Parameters

T
The type of items to permute.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::GenerateSortedPermutations(Of <T> )> Method
(IEnumerable(Of <T> ), Comparison(Of <T> ))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Generates all the possible permutations of the items in collection, in lexicographical order. A supplied Comparison<T> delegate is used to compare the items. Even if some items are equal, the same permutation will not be generated more than once. For example, if the collections contains the three items A, A, and B, then this method will generate only the three permutations, AAB, ABA, BAA.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEnumerable<T[]> GenerateSortedPermutations<T>(
    IEnumerable<T> collection,
    Comparison<T> comparison
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function GenerateSortedPermutations(Of T) ( _
    collection As IEnumerable(Of T), _
    comparison As Comparison(Of T) _
) As IEnumerable(Of T())
```

Visual C++

```cpp
public:
    generic<typename T>
    static IEnumerable<array<T>^>^ GenerateSortedPermutations ( _
        IEnumerable<array<T>^> collection,
        Comparison<array<T>^> comparison
    )
```

Parameters

collection
    IEnumerable<Of <T>>
    The collection of items to permute.

comparison
    Comparison<Of <T>>
    The Comparison<T> delegate used to compare the items.

Return Value

An IEnumerable<T[]> that enumerations all the possible permutations of the
items in collection. Each permutations is returned as an array. The items in the array should be copied if they need to be used after the next permutation is generated; each permutation may reuse the same array instance.
Type Parameters

T
The type of items to permute.
See Also

[Algorithms Class](#)
[Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetCollectionEqualityComparer(Of T)()</code></td>
<td>Gets an <code>IEqualityComparer&lt;IEnumerable&lt;T&gt;&gt;</code> implementation that can be used to compare collections of elements (of type T). Two collections of T's are equal if they have the same number of items, and corresponding items are equal, considered in order. This is the same notion of equality as in Algorithms.EqualCollections, but encapsulated in an <code>IEqualityComparer&lt;IEnumerable&lt;T&gt;&gt;</code> implementation.</td>
</tr>
<tr>
<td><code>GetCollectionEqualityComparer(Of T)(IEqualityComparer(Of T))</code></td>
<td>Gets an <code>IEqualityComparer&lt;IEnumerable&lt;T&gt;&gt;</code> implementation that can be used to compare collections of elements (of type T). Two collections of T's are equal if they have the same number of items, and corresponding items are equal, considered in order. This is the same notion of equality as in Algorithms.EqualCollections, but encapsulated in an <code>IEqualityComparer&lt;IEnumerable&lt;T&gt;&gt;</code> implementation.</td>
</tr>
</tbody>
</table>

An `IEqualityComparer<T>` is used to determine if individual T's are equal.
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
GetCollectionEqualityComparer\<(Of \<T\>\)> Method

See Also: Example

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets an IEqualityComparer\<IEnumerable\<T\>> implementation that can be used to compare collections of elements (of type T). Two collections of T's are equal if they have the same number of items, and corresponding items are equal, considered in order. This is the same notion of equality as in Algorithms.EqualCollections, but encapsulated in an IEqualityComparer\<IEnumerable\<T\>> implementation.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static IEqualityComparer<IEnumerable<T>> GetCollectionEqualityComparer

Visual Basic (Declaration)

Public Shared Function GetCollectionEqualityComparer(Of T) As IEqualityComparer<IEnumerable<T>>

Visual C++

public:
generic<typename T>
static IEqualityComparer<IEnumerable<T>> GetCollectionEqualityComparer

Return Value

IEqualityComparer<IEnumerable<T>> implementation suitable for comparing collections of T for equality.
Type Parameters

T
Examples

The following code creates a Dictionary where the keys are a collection of strings.

```csharp
Dictionary<IE numelable<string>, int> =
    new Dictionary<IE numelable<string>, int>(Algorit
See Also

Algorithms Class
Wintellect.PowerCollections Namespace
Algorithms..:::EqualCollections<Of <T>>>(IEnumerable<Of <T>>, IEnumerable<Of <T>>)

Send comments about this topic to Microsoft.
Algorithms...:::GetCollectionEqualityComparer(Of <T>) Method
(IEqualityComparer(Of IEnumerable<T>))

See Also  Example

❑ Visual Basic (Declaration) ❑ Visual Basic (Usage)
❑ C#
❑ Visual C++
❑ J#
❑ JScript
❑ XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets an IEqualityComparer<IEnumerable<T>> implementation that can be used to compare collections of elements (of type T). Two collections of T's are equal if they have the same number of items, and corresponding items are equal, considered in order. This is the same notion of equality as in Algorithms.EqualCollections, but encapsulated in an IEqualityComparer<IEnumerable<T>> implementation.

An IEqualityComparer<T> is used to determine if individual T's are equal

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static IEqualityComparer<IEnumerable<T>> GetCollectionEqualityComparer(IEqualityComparer<T> equalityComparer)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function GetCollectionEqualityComparer(Of T) (equalityComparer As IEqualityComparer(Of T)) As IEqualityComparer(Of IEnumerable(Of T))
```

**Visual C++**

```cpp
public:
generic<typename T>
static IEqualityComparer<IEnumerable<T>>^ GetCollectionEqualityComparer(IEqualityComparer<T>^ equalityComparer)
```

**Parameters**

equalityComparer

* IEqualityComparer<T>*
    
An IEqualityComparer<T> implementation used to compare individual T's.

**Return Value**

IEqualityComparer<IEnumerable<T>> implementation suitable for comparing collections of T for equality.
Type Parameters

T
Examples

The following code creates a Dictionary where the keys are a collection of strings, compared in a case-insensitive way

```csharp
Dictionary<IEnumerable<string>, int> =
    new Dictionary<IEnumerable<string>, int>(Algorithms.GetCollectionEqualityComparer<string>(StringComparer.CurrentCultureIgnoreCase));
```
See Also

Algorithms Class
Wintellect.PowerCollections Namespace
Algorithms..:::EqualCollections(Of<T>)(IEnumerable(Of<T>), IEnumerable(Of<T>))

Send comments about this topic to Microsoft.
Given a comparison delegate that compares two items of type T, gets an IComparer<T> instance that performs the same comparison.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IComparer<T> GetComparerFromComparison<T>(
    Comparison<T> comparison
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function GetComparerFromComparison(Of T) ( _
    comparison As Comparison(Of T) _
) As IComparer(Of T)
```

Visual C++

```cpp
public:
    generic<typename T>
    static IComparer<T>^ GetComparerFromComparison ( _
        Comparison<T>^ comparison
    )
```

Parameters

comparison
    Comparison(Of T)> The comparison delegate to use.

Return Value

An IComparer<T> that performs the same comparing operation as comparison.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Given in `IComparer<T>` instance that comparers two items from type `T`, gets a `Comparison` delegate that performs the same comparison.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static Comparison<T> GetComparisonFromComparer<T>(
    IComparer<T> comparer
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function GetComparisonFromComparer(Of T) ( _
    comparer As IComparer(Of T) _
) As Comparison(Of T)
```

Visual C++

```cpp
public:
    generic<
        typename T>
    static Comparison<T>^ GetComparisonFromComparer ( 
        IComparer<T>^ comparer
    )
```

Parameters

comparer
    IComparer(Of <T>)>
    The IComparer<T> instance to use.

Return Value

A Comparison<T> delegate that performans the same comparing operation as comparer.
Type Parameters

T
See Also

[ Algorithms Class

[ Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::GetDictionaryConverter Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDictionaryConverter&lt;TKey, TValue&gt; &gt; (IDictionary(Of&lt;TKey, TValue&gt;)&gt;)</td>
<td>Creates a delegate that converts keys to values by used a dictionary to map values. Keys that a not present in the dictionary are converted to the default value (zero or null).</td>
</tr>
<tr>
<td>GetDictionaryConverter&lt;TKey, TValue&gt; &gt; (IDictionary(Of&lt;TKey, TValue&gt;)&gt;, TValue)</td>
<td>Creates a delegate that converts keys to values by used a dictionary to map values. Keys that a not present in the dictionary are converted to a supplied default value.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::GetDictionaryConverter(Of TKey, TValue) Method (IDictionary(Of TKey, TValue>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a delegate that converts keys to values by used a dictionary to map values. Keys that are not present in the dictionary are converted to the default value (zero or null).

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public static Converter<TKey, TValue> GetDictionaryConverter<TKey, TValue>(
    IDictionary<TKey, TValue> dictionary
)
```

### Visual Basic (Declaration)

```vbnet
Public Shared Function GetDictionaryConverter(Of TKey, TValue) (
    _
    dictionary As IDictionary(Of TKey, TValue)
) As Converter(Of TKey, TValue)
```

### Visual C++

```cpp
public:
    generic<typename TKey, typename TValue>
    static Converter<TKey, TValue>^ GetDictionaryConverter (
        IDictionary<TKey, TValue>^ dictionary
    )
```

## Parameters

**dictionary**

IDictionary<(Of <TKey, TValue>)>

The dictionary used to perform the conversion.

## Return Value

A delegate to a method that converts keys to values.
Type Parameters

TKey
TValue
Remarks

This delegate can be used as a parameter in Convert or ConvertAll methods to convert entire collections.
See Also

[Algorithms Class](Wintellect.PowerCollections) Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms...:::GetDictionaryConverter<Of <TKey, TValue>> Method
(IDictionary<Of <TKey, TValue>>, TValue)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a delegate that converts keys to values by used a dictionary to map values. Keys that a not present in the dictionary are converted to a supplied default value.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

public static Converter<TKey, TValue> GetDictionaryConverter<TKey, TValue>(IDictionary<TKey, TValue> dictionary, TValue defaultValue)

**Visual Basic (Declaration)**

Public Shared Function GetDictionaryConverter(Of TKey, TValue) ( _
    dictionary As IDictionary(Of TKey, TValue), _
    defaultValue As TValue _
) As Converter(Of TKey, TValue)

**Visual C++**

public:
    generic<typename TKey, typename TValue>
    static Converter<TKey, TValue>^ GetDictionaryConverter ( _
        IDictionary<TKey, TValue>^ dictionary, _
        TValue defaultValue _
    )

**Parameters**

dictionary
    IDictionary(Of TKey, TValue>)>
    The dictionary used to perform the conversion.

defaultValue
    TValue
    The result of the conversion for keys that are not present in the dictionary.

**Return Value**

A delegate to a method that converts keys to values.
Type Parameters

TKey
TValue
Remarks

This delegate can be used as a parameter in Convert or ConvertAll methods to convert entire collections.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentNullException</td>
<td>dictionary is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::GetIdentityComparer<(Of <T>)> Method

See Also

Visual Basic (Declaration) ➫ Visual Basic (Usage)

C#

Visual C++

J#

JScript

XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets an IEqualityComparer<T> instance that can be used to compare objects of type T for object identity only. Two objects compare equal only if they are references to the same object.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEqualityComparer<T> GetIdentityComparer() where T
```

Visual Basic (Declaration)

```
Public Shared Function GetIdentityComparer(Of T As Class) As IEqualityComparer
```

Visual C++

```cpp
public:
generic<typename T>
where T : ref class
static IEqualityComparer<T>^ GetIdentityComparer()
```

Return Value

An IEqualityComparer<T> instance for identity comparison.
Type Parameters

T
See Also

- **Algorithms** Class
- **Wintellect.PowerCollections** Namespace

Send [comments](mailto:) about this topic to Microsoft.
Algorithms...::GetLexicographicalComparer Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetLexicographicalComparer(Of T)</code></td>
<td>Creates an IComparer instance that can be used for comparing ordered</td>
</tr>
<tr>
<td></td>
<td>sequences of type T; that is IEnumerable&lt;Tgt;. This comparer can be used</td>
</tr>
<tr>
<td></td>
<td>for collections or algorithms that use sequences of T as an item type.</td>
</tr>
<tr>
<td></td>
<td>The Lexicographical ordered of sequences is for comparison.</td>
</tr>
<tr>
<td><code>GetLexicographicalComparer(Of T)(IComparer(Of T))</code></td>
<td>Creates an IComparer instance that can be used for comparing ordered</td>
</tr>
<tr>
<td></td>
<td>sequences of type T; that is IEnumerable&lt;Tgt;. This comparer can be used</td>
</tr>
<tr>
<td></td>
<td>for collections or algorithms that use sequences of T as an item type.</td>
</tr>
<tr>
<td></td>
<td>The Lexicographics ordered of sequences is for comparison.</td>
</tr>
<tr>
<td><code>GetLexicographicalComparer(Of T)(Comparison(Of T))</code></td>
<td>Creates an IComparer instance that can be used for comparing ordered</td>
</tr>
<tr>
<td></td>
<td>sequences of type T; that is IEnumerable&lt;Tgt;. This comparer can be used</td>
</tr>
<tr>
<td></td>
<td>for collections or algorithms that use sequences of T as an item type.</td>
</tr>
<tr>
<td></td>
<td>The Lexicographics ordered of sequences is for comparison.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#) [Algorithms Members](#) [Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Algorithms::GetLexicographicalComparer(Of <T>.) Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates an IComparer instance that can be used for comparing ordered sequences of type T; that is IEnumerable<Tgt; . This comparer can be used for collections or algorithms that use sequences of T as an item type. The Lexicographical ordered of sequences is for comparison.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static IComparer<IEnumerable<T>> GetLexicographicalComparer<T>()

Visual Basic (Declaration)

Public Shared Function GetLexicographicalComparer(Of T As IComparable)

Visual C++

public:
generic<typename T>
where T : IComparable<T>
static IComparer<IEnumerable<T>>^ GetLexicographicalComparer ()

Return Value

At IComparer<IEnumerable<T>> that compares sequences of T.
- **Type Parameters**

  T
Remarks

T must implement either IComparable<T> and this implementation is used to compare the items.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::GetLexicographicalComparer(Of <T>) Method (IComparer(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates an IComparer instance that can be used for comparing ordered sequences of type T; that is IEnumerable<Tgt>. This comparer can be used for collections or algorithms that use sequences of T as an item type. The Lexicographically ordered sequences is for comparison.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static IComparer<IEnumerable<T>> GetLexicographicalComparer(IComparer<T> comparer)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function GetLexicographicalComparer(Of T)(comparer As IComparer(Of T)) As IComparer(Of IEnumerable(Of T))
```

**Visual C++**

```cpp
public:
generic<

typename T>
static IComparer<IEnumerable<T>^>^ GetLexicographicalComparer (IComparer<T>^ comparer)
```

**Parameters**

comparer

**IComparer<(Of <T>)>**

A comparer instance used to compare individual items of type T.

**Return Value**

At IComparer<IEnumerable<T>> that compares sequences of T.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
GetLexicographicalComparer<Of<T>> Method (Comparison<Of<T>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates an IComparer instance that can be used for comparing ordered sequences of type T; that is IEnumerable<Tgt;. This comparer can be uses for collections or algorithms that use sequences of T as an item type. The Lexicographics ordered of sequences is for comparison.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static IComparer(IEnumerable<T>> GetLexicographicalComparer(Comparison<T> comparison)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function GetLexicographicalComparer(Of T) ( comparison As Comparison(Of T) ) As IComparer(Of IEnumerable(Of T))
```

**Visual C++**

```cpp
public:
generic< typename T >
static IComparer<IEnumerable<T>>^ GetLexicographicalComparer ( Comparison<T>^ comparison)
```

### Parameters

- **comparison**
  - `Comparison<Of <T>>`:
    - A comparison delegate used to compare individual items of type T.

### Return Value

- At `IComparer(IEnumerable<T>>` that compares sequences of T.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::GetReverseComparer<Of <T>> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Reverses the order of comparison of an IComparer<T>. The resulting comparer can be used, for example, to sort a collection in descending order. Equality and hash codes are unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static IComparer<T> GetReverseComparer<T>(
    IComparer<T> comparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function GetReverseComparer(Of T) ( _
    comparer As IComparer(Of T) _
) As IComparer(Of T)
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static IComparer<T>* GetReverseComparer ( 
        IComparer<T>* comparer
    )
```

### Parameters

**comparer**

[**IComparer(Of T)**](#)

The comparer to reverse.

### Return Value

An IComparer<T> that compares items in the reverse order of comparer.
Type Parameters

T
    The type of items that are being compared.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentNullException</td>
<td>comparer is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::GetReverseComparison<Of <T>> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Reverses the order of comparison of an `Comparison<T>`. The resulting comparison can be used, for example, to sort a collection in descending order.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static Comparison<T> GetReverseComparison<T>(
    Comparison<T> comparison
)

Visual Basic (Declaration)

Public Shared Function GetReverseComparison(Of T) ( _
    comparison As Comparison(Of T) _
) As Comparison(Of T)

Visual C++

public:
    generic<typename T>
    static Comparison<T>^ GetReverseComparison ( 
        Comparison<T>^ comparison
    )

Parameters

comparison
    Comparison(Of <T>)>
    The comparison to reverse.

Return Value

A Comparison<T> that compares items in the reverse order of comparison.
Type Parameters

T
The type of items that are being compared.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...::ArgumentNullException</td>
<td>comparison is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::GetSetEqualityComparer Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetSetEqualityComparer&lt;Of &lt;T&gt;&gt;()</code></td>
<td>Gets an IEqualityComparer&lt;IEnumerable&lt;T&gt;&gt; implementation that can be used to compare collections of elements (of type T). Two collections of T's are equal if they have the same number of items, and corresponding items are equal, without regard to order. This is the same notion of equality as in Algorithms.EqualSets, but encapsulated in an IEqualityComparer&lt;IEnumerable&lt;T&gt;&gt; implementation.</td>
</tr>
<tr>
<td><code>GetSetEqualityComparer&lt;Of &lt;T&gt;&gt;((IEqualityComparer&lt;Of &lt;T&gt;&gt;))</code></td>
<td>An IEqualityComparer&lt;T&gt; is used to determine if individual T's are equal</td>
</tr>
<tr>
<td><code>GetSetEqualityComparer&lt;Of &lt;T&gt;&gt;((IEqualityComparer&lt;Of &lt;T&gt;&gt;))</code></td>
<td>Gets an IEqualityComparer&lt;IEnumerable&lt;T&gt;&gt; implementation that can be used to compare collections of elements (of type T). Two collections of T's are equal if they have the same number of items, and corresponding items are equal, without regard to order. This is the same notion of equality as in Algorithms.EqualSets, but encapsulated in an IEqualityComparer&lt;IEnumerable&lt;T&gt;&gt; implementation.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::GetSetEqualityComparer<(Of <T>)> Method

See Also  Example

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets an IEqualityComparer<IEnumerable<T>> implementation that can be used to compare collections of elements (of type T). Two collections of T's are equal if they have the same number of items, and corresponding items are equal, without regard to order. This is the same notion of equality as in Algorithms.EqualSets, but encapsulated in an IEqualityComparer<IEnumerable<T>> implementation.

An IEqualityComparer<T> is used to determine if individual T's are equal

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#
public static IEqualityComparer<IEnumerable<T>> GetSetEqualityComparer();

Visual Basic (Declaration)
Public Shared Function GetSetEqualityComparer(Of T) As IEqualityComparer<IEnumerable(Of T)>;

Visual C++

public:
static IEqualityComparer IEnumerable<T>:: GetSetEqualityComparer();

Return Value

IEqualityComparer IEnumerable<T>:: implementation suitable for comparing collections of T for equality, without regard to order.
Type Parameters

T
**Examples**

The following code creates a Dictionary where the keys are a set of strings, without regard to order

```csharp
Dictionary<IEnumerable<string>, int> =
    new Dictionary<IEnumerable<string>, int>(Algorit
See Also

Algorithms Class
Wintellect.PowerCollections Namespace
Algorithms...::EqualSets<Of <T>>(IEnumerable<Of <T>>, IEnumerable<Of <T>>)

Send comments about this topic to Microsoft.
Algorithms::GetSetEqualityComparer(Of T) Method
(IEqualityComparer(Of T))

See Also Example

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets an IEqualityComparer<IEnumerable<T>> implementation that can be used to compare collections of elements (of type T). Two collections of T's are equal if they have the same number of items, and corresponding items are equal, without regard to order. This is the same notion of equality as in Algorithms.EqualSets, but encapsulated in an IEqualityComparer<IEnumerable<T>> implementation.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)


**Syntax**

**C#**

public static IEqualityComparer< IEnumerable<T>> GetSetEqualityComparer(IEqualityComparer<T> equalityComparer)

**Visual Basic (Declaration)**

Public Shared Function GetSetEqualityComparer(Of T) ( _
equalityComparer As IEqualityComparer(Of T) _
) As IEqualityComparer(Of IEnumerable(Of T))

**Visual C++**

public:

generic< typename T>

static IEqualityComparer< IEnumerable<T>^>^ GetSetEqualityComparer ( IEqualityComparer< T>^ equalityComparer

**Parameters**

equalityComparer

IEqualityComparer<Of <T>>

An IEqualityComparer<T> implementation used to compare individual T's.

**Return Value**

IEqualityComparer<IEnumerable<T>> implementation suitable for comparing collections of T for equality, without regard to order.
Type Parameters

T
Examples

The following code creates a Dictionary where the keys are a set of strings, without regard to order

```
Dictionary<IEnumerable<string>, int> =
    new Dictionary<IEnumerable<string>, int>(Algorit
See Also

Algorithms Class
Wintellect.PowerCollections Namespace
Algorithms...::EqualSets<Of <T>> (IEnumerable<Of <T>>, IEnumerable<Of <T>>)

Send comments about this topic to Microsoft.
Algorithms...:::IndexOfMaximum Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overload List</td>
<td>Finds the index of the maximum value in a list.</td>
</tr>
<tr>
<td>IndexOfMaximum(Of <code>&lt;T&gt;</code>&gt;(IList(Of <code>&lt;T&gt;</code>)&gt;)</td>
<td>Finds the index of the maximum value in a list.</td>
</tr>
<tr>
<td>IndexOfMaximum(Of <code>&lt;T&gt;</code>&gt;(IList(Of <code>&lt;T&gt;</code>&gt;), IComparer(Of <code>&lt;T&gt;</code>))</td>
<td>A supplied IComparer&lt;T&gt; is used to compare the items in the collection.</td>
</tr>
<tr>
<td>IndexOfMaximum(Of <code>&lt;T&gt;</code>&gt;(IList(Of <code>&lt;T&gt;</code>&gt;), Comparison(Of <code>&lt;T&gt;</code>))</td>
<td>Finds the index of the maximum value in a list.</td>
</tr>
<tr>
<td>A supplied Comparison&lt;T&gt; delegate is used to compare the items in the collection.</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::IndexOfMaximum(Of <T>)(Of <T>)(Of <T>) Method (IList(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the maximum value in a list.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public static int IndexOfMaximum<T>(
    IList<T> list
) where T : IComparable<T>
```

#### Visual Basic (Declaration)

```vbnet
Public Shared Function IndexOfMaximum(Of T As IComparable(Of T)) ( _
    list As IList(Of T) _
) As Integer
```

#### Visual C++

```cpp
public:
    generic< typename T>
    where T : IComparable<T>
    static int IndexOfMaximum ( 
        IList<T>^ list
    )
```

### Parameters

- **list**
  - `IList<T>`
  - The list to search.

### Return Value

The index of the largest item in the list. If the maximum value appears multiple times, the index of the first appearance is used. If the list is empty, -1 is returned.
Type Parameters

T

The type of items in the list.
Remarks

Values in the list are compared by using the IComparable<T> interfaces implementation on the type T.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>list is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::IndexOfMaximum(Of T) Method (IList(Of T), IComparer(Of T))

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the maximum value in a list. A supplied IComparer<T> is used to compare the items in the collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static int IndexOfMaximum<T>(
    IList<T> list,
    IComparer<T> comparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IndexOfMaximum(Of T) ( _
    list As IList(Of T), _
    comparer As IComparer(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:
    template<typename T>
    static int IndexOfMaximum ( 
        IList<T>* list,
        IComparer<T>* comparer
    )
```

### Parameters

- **list**
  - `IList<Of <T>>`
  - The list to search.

- **comparer**
  - `IComparer<Of <T>>`
  - The comparer instance used to compare items in the collection.

### Return Value

The index of the largest item in the list. If the maximum value appears multiple
times, the index of the first appearance is used. If the list is empty, -1 is returned.
Type Parameters

T

The type of items in the list.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgumentNullException</code></td>
<td>list or comparer is null.</td>
</tr>
</tbody>
</table>

**Exceptions**
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
IndexOfMaximum<
(T)> Method (IList<
(T)>, Comparison<
(T)>)

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[Finds the index of the maximum value in a list. A supplied Comparison<T> delegate is used to compare the items in the collection.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static int IndexOfMaximum<T>(
    IList<T> list,
    Comparison<T> comparison
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function IndexOfMaximum(Of T) ( _
    list As IList(Of T), _
    comparison As Comparison(Of T) _
) As Integer
```

Visual C++

```csharp
public:
    generic< typename T >
    static int IndexOfMaximum ( 
        IList<T>^ list,
        Comparison<T>^ comparison
    )
```

Parameters

list

```csharp
IList<(Of <T)>>
```

The list to search.

comparison

```csharp
Comparison<(Of <T)>>
```

The comparison used to compare items in the collection.

Return Value

The index of the largest item in the list. If the maximum value appears multiple
times, the index of the first appearance is used. If the list is empty, -1 is returned.
Type Parameters

T
    The type of items in the list.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>list or comparison is null.</td>
</tr>
</tbody>
</table>
See Also

- **Algorithms** Class
- **Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms...:::IndexofMinimum Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IndexOfMinimum(Of T)</code>(IList(Of T))</td>
<td>Finds the index of the minimum value in a list.</td>
</tr>
<tr>
<td><code>IndexOfMinimum(Of T)</code>(IList(Of T), IComparer(Of T))</td>
<td>Finds the index of the minimum value in a list. A supplied IComparer&lt;T&gt; is used to compare the items in the collection.</td>
</tr>
<tr>
<td><code>IndexOfMinimum(Of T)</code>(IList(Of T), Comparison(Of T))</td>
<td>Finds the index of the minimum value in a list. A supplied Comparison&lt;T&gt; delegate is used to compare the items in the collection.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::IndexOfMinimum(Of <T>)> Method (IList(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the minimum value in a list.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static int IndexOfMinimum<T>(
    IList<T> list
) where T : IComparable<T>
```

Visual Basic (Declaration)

```vbnet
Public Shared Function IndexOfMinimum(Of T As IComparable(Of T)) ( _
    list As IList(Of T) _
) As Integer
```

Visual C++

```cpp
public:
    generic<typename T>
    where T : IComparable<T>
    static int IndexOfMinimum ( 
        IList<T>^ list
    )
```

Parameters

list

`IList<Of <T>>`

The list to search.

Return Value

The index of the smallest item in the list. If the minimum value appears multiple times, the index of the first appearance is used.
Type Parameters

T

The type of items in the list.
Remarks

Values in the list are compared by using the IComparable<T> interfaces implementation on the type T.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>The collection is empty.</td>
</tr>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>list is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::IndexOfMinimum<Of <T>> Method (IList<Of <T>>, IComparer<Of <T>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the minimum value in a list. A supplied IComparer< T> is used to compare the items in the collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

**C#**

```csharp
public static int IndexOfMinimum<T>(
    IList<T> list,
    IComparer<T> comparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IndexOfMinimum(Of T) ( _
    list As IList(Of T), _
    comparer As IComparer(Of T) _
) As Integer
```

**Visual C++**

```csharp
public:
    generic<typename T>
    static int IndexOfMinimum ( 
        IList<T>^ list,
        IComparer<T>^ comparer
    )
```

### Parameters

- **list**
  - `IList<T>`
  - The list to search.

- **comparer**
  - `IComparer<T>`
  - The comparer instance used to compare items in the collection.

### Return Value

The index of the smallest item in the list. If the minimum value appears multiple
times, the index of the first appearance is used.
Type Parameters

\( T \)

The type of items in the list.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::InvalidOperationException</td>
<td>The collection is empty.</td>
</tr>
<tr>
<td>System::ArgumentNullException</td>
<td>list or comparer is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::IndexOfMinimum(Of <T>)-> Method (IList(Of <T>), Comparison(Of <T>))

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the minimum value in a list. A supplied Comparison<T> delegate is used to compare the items in the collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static int IndexOfMinimum<T>(
    IList<T> list,
    Comparison<T> comparison
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function IndexOfMinimum(Of T) ( _
    list As IList(Of T), _
    comparison As Comparison(Of T) _
) As Integer
```

Visual C++

```cpp
public:
    generic<typename T>
    static int IndexOfMinimum ( 
        IList<T>* list,
        Comparison<T>* comparison
    )
```

Parameters

list
    IList<Of <T>>
    The list to search.

comparison
    Comparison<Of <T>>
    The comparison delegate used to compare items in the collection.

Return Value

The index of the smallest item in the list. If the minimum value appears multiple
times, the index of the first appearance is used.
Type Parameters

T

The type of items in the list.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::InvalidOperationException</td>
<td>The collection is empty.</td>
</tr>
<tr>
<td>System::ArgumentNullException</td>
<td>list or comparison is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::IndicesOf Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IndicesOf(Of&lt;T&gt;)(IList(Of&lt;T&gt;), T)</code></td>
<td>Enumerates the indices of all the items in a list equal to a given item.</td>
</tr>
<tr>
<td><code>IndicesOf(Of&lt;T&gt;)(IList(Of&lt;T&gt;), T, IEqualityComparer(Of&lt;T&gt;))</code></td>
<td>Enumerates the indices of all the items in a list equal to a given item. A passed <code>IEqualityComparer</code> is used to determine equality.</td>
</tr>
</tbody>
</table>
See Also

- Algorithms Class
- Algorithms Members
- Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Enumerates the indices of all the items in a list equal to a given item.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public static IEnumerable<int> IndicesOf<T>(
    IList<T> list,
    T item
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IndicesOf(Of T) ( _
    list As IList(Of T), _
    item As T _
) As IEnumerable(Of Integer)
```

**Visual C++**

```cpp
public:
    template<typename T>
    static IEnumerable<int>^ IndicesOf( 
        IList<T>^ list,
        T item
    )
```

**Parameters**

- **list**
  - `IList<Of <T>>`
  - The list to search.

- **item**
  - `T`
  - The item to search for.

**Return Value**

An `IEnumerable<T>` that enumerates the indices of items equal to `item`. 
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::IndicesOf<(Of <T>), Method (IList<(Of <T>), T, IEqualityComparer<(Of <T>))>

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerates the indices of all the items in a list equal to a given item. A passed IEqualityComparer is used to determine equality.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
 Syntax

C#

public static IEnumerable<int> IndicesOf<T>(
    IList<T> list,
    T item,
    IEqualityComparer<T> equalityComparer
)

Visual Basic (Declaration)

Public Shared Function IndicesOf(Of T) ( _
    list As IList(Of T), _
    item As T, _
    equalityComparer As IEqualityComparer(Of T) _
) As IEnumerable(Of Integer)

Visual C++

public:
    generic<typename T>
    static IEnumerable<int>^ IndicesOf ( 
        IList<T>^ list,
        T item,
        IEqualityComparer<T>^ equalityComparer
    )

Parameters

list
    IList<Of <T>>
    The list to search.

time
    T
    The item to search for.

equalityComparer
**IEqualityComparer<T>**
The IEqualityComparer<T> used to compare items for equality. Only the Equals method will be called.

**Return Value**
An IEnumerable<T> that enumerates the indices of items equal to item.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::IndicesOfMany Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndicesOfMany&lt;Of&lt;T&gt;&gt;&lt;Of&lt;T&gt;&gt;&lt;IList&lt;Of&lt;T&gt;&gt;, IEnumerable&lt;Of&lt;T&gt;&gt;, IEqualityComparer&lt;Of&lt;T&gt;&gt;, BinaryPredicate&lt;Of&lt;T&gt;&gt;&gt;</td>
<td>Enumerates the indices of all the items in a list equal to one of several given items. A passed IEqualityComparer is used to determine equality.</td>
</tr>
<tr>
<td>IndicesOfMany&lt;Of&lt;T&gt;&gt;&lt;Of&lt;T&gt;&gt;&lt;IList&lt;Of&lt;T&gt;&gt;, IEnumerable&lt;Of&lt;T&gt;&gt;, IEqualityComparer&lt;Of&lt;T&gt;&gt;&gt;</td>
<td>Enumerates the indices of all the items in a list equal to one of several given items.</td>
</tr>
<tr>
<td>IndicesOfMany&lt;Of&lt;T&gt;&gt;&lt;Of&lt;T&gt;&gt;&lt;IList&lt;Of&lt;T&gt;&gt;, IEnumerable&lt;Of&lt;T&gt;&gt;&gt;</td>
<td>Enumerates the indices of all the items in a list equal to one of several given items.</td>
</tr>
<tr>
<td>IndicesOfMany&lt;Of&lt;T&gt;&gt;&lt;Of&lt;T&gt;&gt;&lt;IList&lt;Of&lt;T&gt;&gt;, IEnumerable&lt;Of&lt;T&gt;&gt;&gt;</td>
<td>Enumerates the indices of all the items in a list equal to one of several given items.</td>
</tr>
<tr>
<td>IndicesOfMany&lt;Of&lt;T&gt;&gt;&lt;Of&lt;T&gt;&gt;</td>
<td>Enumerates the indices of all the items in a list equal to one of several given items.</td>
</tr>
</tbody>
</table>

Enumerates the indices of all the items in a list equal to one of several given items.
See Also

[Algorithms Class](#)
[Algorithms Members](#)
[Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Enumerates the indices of all the items in a list equal to one of several given items.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static IEnumerable<int> IndicesOfMany<T>(
    IList<T> list,
    IEnumerable<T> itemsToLookFor
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IndicesOfMany(Of T) ( _
    list As IList(Of T), _
    itemsToLookFor As IEnumerable(Of T) _
) As IEnumerable(Of Integer)
```

**Visual C++**

```cpp
public:
    template<typename T>
    static IEnumerable<int>^ IndicesOfMany ( 
        IList<T>^ list,
        IEnumerable<T>^ itemsToLookFor
    )
```

**Parameters**

- **list**
  - `IList<Of <T>>`
  - The list to search.

- **itemsToLookFor**
  - `IEnumerable<Of <T>>`
  - A collection of items to search for.

**Return Value**

An `IEnumerable<T>` that enumerates the indices of items equal to any of the
items in the collection itemsToLookFor.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms..:::IndicesOfMany(Of <T>)> Method (IList(Of <T>), IEnumerable(Of <T>), IEqualityComparer(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerates the indices of all the items in a list equal to one of several given items. A passed IEqualityComparer is used to determine equality.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static IEnumerable<int> IndicesOfMany<T>(
    IList<T> list,
    IEnumerable<T> itemsToLookFor,
    IEqualityComparer<T> equalityComparer
)

Visual Basic (Declaration)

Public Shared Function IndicesOfMany(Of T) ( _
    list As IList(Of T), _
    itemsToLookFor As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
) As IEnumerable(Of Integer)

Visual C++

public:
    generic<typename T>
    static IEnumerable<int>^ IndicesOfMany ( 
        IList<T>^ list,
        IEnumerable<T>^ itemsToLookFor,
        IEqualityComparer<T>^ equalityComparer
    )

Parameters

list
    IList<Of <T>>
    The list to search.

itemsToLookFor
    IEnumerable<Of <T>>
    A collection of items to search for.

equalityComparer
**IEqualityComparer<Of <T>>**

The IEqualityComparer<T> used to compare items for equality.

**Return Value**

An IEnumerable<T> that enumerates the indices of items equal to any of the items in the collection itemsToLookFor.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Enumerates the indices of all the items in a list equal to one of several given items. The passed BinaryPredicate is used to determine if two items are "equal".

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEnumerable<int> IndicesOfMany<T>(
    IList<T> list,
    IEnumerable<T> itemsToLookFor,
    BinaryPredicate<T> predicate
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function IndicesOfMany(Of T) ( _
    list As IList(Of T), _
    itemsToLookFor As IEnumerable(Of T), _
    predicate As BinaryPredicate(Of T) _
) As IEnumerable(Of Integer)
```

Visual C++

```cpp
public:
    template<typename T>
    static IEnumerable<int>^ IndicesOfMany ( 
        IList<T>^ list, 
        IEnumerable<T>^ itemsToLookFor, 
        BinaryPredicate<T>^ predicate
    )
```

Parameters

list

- `IList<Of <T>>`
  - The list to search.

itemsToLookFor

- `IEnumerable<Of <T>>`
  - A collection of items to search for.

predicate
**BinaryPredicate**<(Of <T> )>
The BinaryPredicate used to compare items for "equality".

**Return Value**

An IEnumerable<T> that enumerates the indices of items "equal" to any of the items in the collection itemsToLookFor, using as the test for equality.
Type Parameters

T
Remarks

Since an arbitrary BinaryPredicate is passed to this function, what is being removed need not be true equality. This method finds last item X which satisfies BinaryPredicate(X,Y), where Y is one of the items in itemsToLookFor.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::IsProperSubsetOf Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsProperSubsetOf&lt;(Of </code>&lt;T&gt;<code>)&gt; (IEnumerable&lt;(Of </code>&lt;T&gt;<code>)&gt;, IEnumerable&lt;(Of </code>&lt;T&gt;<code>)&gt;)</code></td>
<td>Determines if one collection is a proper subset of another, considered as sets. The first set is a proper subset of the second set if every item in the first set also occurs in the second set, and the first set is strictly smaller than the second set. If an item appears X times in the first set, it must appear at least X times in the second set.</td>
</tr>
<tr>
<td><code>IsProperSubsetOf&lt;(Of </code>&lt;T&gt;<code>)&gt; (IEnumerable&lt;(Of </code>&lt;T&gt;<code>)&gt;, IEnumerable&lt;(Of </code>&lt;T&gt;<code>), IEquivalenceComparer&lt;(Of </code>&lt;T&gt;<code>)&gt;)</code></td>
<td>Determines if one collection is a proper subset of another, considered as sets. The first set is a proper subset of the second set if every item in the first set also occurs in the second set, and the first set is strictly smaller than the second set. If an item appears X times in the first set, it must appear at least X times in the second set.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#) [Algorithms Members](#) [Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Determines if one collection is a proper subset of another, considered as sets. The first set is a proper subset of the second set if every item in the first set also occurs in the second set, and the first set is strictly smaller than the second set. If an item appears \( X \) times in the first set, it must appear at least \( X \) times in the second set.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
- Syntax

C#

```csharp
public static bool IsProperSubsetOf<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function IsProperSubsetOf(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
    generic<typename T>
    static bool IsProperSubsetOf ( 
        IEnumerable<T>^ collection1,
        IEnumerable<T>^ collection2
    )
```

Parameters

collection1
```
IEnumerable<(Of <T>)>
```
The first collection.

collection2
```
IEnumerable<(Of <T>)>
```
The second collection.

Return Value

True if collection1 is a subset of collection2, considered as sets.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the IsSubsetOf method on that class.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::ArgumentNullException</code></td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::IsProperSubsetOf<(Of <T>)> Method (IEnumerable<(Of <T>)>, IEnumerable<(Of <T>)>, IEqualityComparer<(Of <T>)>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

Determines if one collection is a proper subset of another, considered as sets. The first set is a proper subset of the second set if every item in the first set also occurs in the second set, and the first set is strictly smaller than the second set. If an item appears X times in the first set, it must appear at least X times in the second set.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static bool IsProperSubsetOf<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2,
    IEqualityComparer<T> equalityComparer
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function IsProperSubsetOf(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
) As Boolean
```

Visual C++

```c++
public:
    generic<typename T>
    static bool IsProperSubsetOf ( 
        IEnumerable<T>^ collection1, 
        IEnumerable<T>^ collection2, 
        IEqualityComparer<T>^ equalityComparer 
    )
```

Parameters

collection1
    IEnumerable<(Of <T>)> The first collection.

collection2
    IEnumerable<(Of <T>)> The second collection.
equalityComparer
**IEqualityComparer<Of <T>>**
The IEqualityComparer<T> used to compare items for equality. Only the Equals and GetHashCode member functions of this interface are called.

**Return Value**

True if collection1 is a proper subset of collection2, considered as sets.
Type Parameters

T
Remarks

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the IsSubsetOf method on that class.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentNullException</td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

**Algorithms** Class  
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms...:::IsSubsetOf Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsSubsetOf(Of&lt;T&gt;, IEnumerable(Of&lt;T&gt;))</td>
<td>Determines if one collection is a subset of another, considered as sets. The first set is a subset of the second set if every item in the first set also occurs in the second set. If an item appears X times in the first set, it must appear at least X times in the second set.</td>
</tr>
<tr>
<td>IsSubsetOf(Of&lt;T&gt;, IEnumerable(Of&lt;T&gt;), IEqualityComparer(Of&lt;T&gt;))</td>
<td>Determines if one collection is a subset of another, considered as sets. The first set is a subset of the second set if every item in the first set also occurs in the second set. If an item appears X times in the first set, it must appear at least X times in the second set.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::IsSubsetOf(Of <T>[]) Method (IEnumerable(Of <T>),
IEnumerable(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if one collection is a subset of another, considered as sets. The first set is a subset of the second set if every item in the first set also occurs in the second set. If an item appears X times in the first set, it must appear at least X times in the second set.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static bool IsSubsetOf<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function IsSubsetOf(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
    generic<typename T>
    static bool IsSubsetOf ( 
        IEnumerable<T>^ collection1,
        IEnumerable<T>^ collection2
    )
```

Parameters

collection1
    IEnumerable<(Of <T>)>  
    The first collection.

collection2
    IEnumerable<(Of <T>)>  
    The second collection.

Return Value

True if collection1 is a subset of collection2, considered as sets.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable\<T\>.Equals or object.Equals.

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the IsSubsetOf method on that class.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

*Algorithms* Class  
*Wintellect.PowerCollections* Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms...:::IsSubsetOf(Of T)> Method (IEnumerable(Of T), IEnumerable(Of T), IEqualityComparer(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if one collection is a subset of another, considered as sets. The first set is a subset of the second set if every item in the first set also occurs in the second set. If an item appears X times in the first set, it must appear at least X times in the second set.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static bool IsSubsetOf<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2,
    IEqualityComparer<T> equalityComparer
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function IsSubsetOf(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
    generic<typename T>
    static bool IsSubsetOf ( 
        IEnumerable<T>^ collection1,
        IEnumerable<T>^ collection2,
        IEqualityComparer<T>^ equalityComparer
    )
```

Parameters

`collection1`  
**IEnumerable**<(Of <T>>)  
The first collection.

`collection2`  
**IEnumerable**<(Of <T>>)  
The second collection.

`equalityComparer`
**IEqualityComparer<**Of **<>**>

The IEqualityComparer<T> used to compare items for equality.

**Return Value**

True if collection1 is a subset of collection2, considered as sets.
Type Parameters

T
Remarks

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the IsSubsetOf method on that class.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::LastIndexOf Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LastIndexOf&lt;(Of &lt;T&gt;), (IList&lt;(Of &lt;T&gt;), T)&gt;</code></td>
<td>Finds the index of the last item in a list equal to a given item.</td>
</tr>
<tr>
<td><code>LastIndexOf&lt;(Of &lt;T&gt;), (IList&lt;(Of &lt;T&gt;), T, IEqualityComparer&lt;(Of &lt;T&gt;))&gt;</code></td>
<td>Finds the index of the last item in a list equal to a given item. A passed IEqualityComparer is used to determine equality.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::<LastIndexOf<(Of <T>)> Method (IList<(Of <T>)>, T)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the last item in a list equal to a given item.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static int LastIndexOf<T>(
    IList<T> list,
    T item
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function LastIndexOf(Of T) ( _
    list As IList(Of T), _
    item As T _
) As Integer
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static int LastIndexOf( (}
        IList<T>^ list,  
        T item
    )
```

**Parameters**

`list`

`IList<Of <T>>`

The list to search.

`item`

`T`

The item to search for.

**Return Value**

The index of the last item equal to item. -1 if no such item exists in the list.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.
See Also

See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Finds the index of the last item in a list equal to a given item. A passed IEqualityComparer is used to determine equality.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
- Syntax

C#

```csharp
public static int LastIndexOf<T>(
    IList<T> list,
    T item,
    IEqualityComparer<T> equalityComparer
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function LastIndexOf(Of T) ( _
    list As IList(Of T), _
    item As T, _
    equalityComparer As IEqualityComparer(Of T) _
) As Integer
```

Visual C++

```cpp
public:
    template<typename T>
    static int LastIndexOf( 
        IList<T>^ list,
        T item,
        IEqualityComparer<T>^ equalityComparer
    )
```

Parameters

- `list` of type `IList<Of <T>>`
  - The list to search.

- `item` of type `T`
  - The item to search for.

- `equalityComparer`
**IEqualityComparer<T>**

The `IEqualityComparer<T>` used to compare items for equality. Only the `Equals` method will be called.

**Return Value**

The index of the last item equal to item. -1 if no such item exists in the list.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::LastIndexOfMany Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LastIndexOfMany&lt;T&gt;</td>
<td>Finds the index of the last item in a list equal to one of several given items.</td>
</tr>
<tr>
<td>LastIndexOfMany&lt;T&gt;</td>
<td>Finds the index of the last item in a list equal to one of several given items. A passed IEqualityComparer is used to determine equality.</td>
</tr>
<tr>
<td>LastIndexOfMany&lt;T&gt;</td>
<td>Finds the index of the last item in a list &quot;equal&quot; to one of several given items. The passed BinaryPredicate is used to determine if two items are &quot;equal&quot;.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#)
[Algorithms Members](#)
[Wintellect.PowerCollections Namespace](#)

Send comments about this topic to Microsoft.
Algorithms:::LastIndexOfMany(Of <T>) Method (IList(Of <T>), IEnumerable(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the last item in a list equal to one of several given items.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static int LastIndexOfMany<T>(
    IList<T> list,
    IEnumerable<T> itemsToLookFor
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function LastIndexOfMany(Of T) ( _
    list As IList(Of T), _
    itemsToLookFor As IEnumerable(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:
    generic< typename T >
    static int LastIndexOfMany ( 
        IList<T>* list,
        IEnumerable<T>* itemsToLookFor
    )
```

### Parameters

**list**

`IList<Of <T>>`

The list to search.

**itemsToLookFor**

`IEnumerable<Of <T>>`

The items to search for.

### Return Value

The index of the last item equal to any of the items in the collection
itemsToLookFor. -1 if no such item exists in the list.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.
See Also

 Algorithms Class
 Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Finds the index of the last item in a list equal to one of several given items. A passed IEqualityComparer is used to determine equality.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static int LastIndexOfMany<T>(
   IList<T> list,
   IEnumerable<T> itemsToLookFor,
   IEqualityComparer<T> equalityComparer
)

Visual Basic (Declaration)

Public Shared Function LastIndexOfMany(Of T) ( _
   list As IList(Of T), _
   itemsToLookFor As IEnumerable(Of T), _
   equalityComparer As IEqualityComparer(Of T) _
) As Integer

Visual C++

public:
   generic<typename T>
   static int LastIndexOfMany ( 
      IList<T>* list,
      IEnumerable<T>* itemsToLookFor,
      IEqualityComparer<T>* equalityComparer
   )

Parameters

list
   IList<Of <T>>
   The list to search.

itemsToLookFor
   IEnumerable<Of <T>>
   The items to search for.

equalityComparer
**IEqualityComparer<T>**

The IEqualityComparer<T> used to compare items for equality.

**Return Value**

The index of the last item equal to any of the items in the collection itemsToLookFor. -1 if no such item exists in the list.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms..::.LastIndexOfMany(Of <T>), Method (IList(Of <T>), IEnumerable(Of <T>), BinaryPredicate(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the last item in a list "equal" to one of several given items. The passed BinaryPredicate is used to determine if two items are "equal".

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static int LastIndexOfMany<T>(
    IList<T> list,
    IEnumerable<T> itemsToLookFor,
    BinaryPredicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function LastIndexOfMany(Of T) ( _
    list As IList(Of T), _
    itemsToLookFor As IEnumerable(Of T), _
    predicate As BinaryPredicate(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static int LastIndexOfMany ( 
        IList<T>* list,
        IEnumerable<T>* itemsToLookFor,
        BinaryPredicate<T>* predicate
    )
```

**Parameters**

**list**

```csharp
IList<Of <T>>
```

The list to search.

**itemsToLookFor**

```csharp
IEnumerable<Of <T>>
```

The items to search for.

**predicate**
BinaryPredicate<(Of <T>)}
The BinaryPredicate used to compare items for "equality".

**Return Value**

The index of the last item "equal" to any of the items in the collection itemsToLookFor, using as the test for equality. -1 if no such item exists in the list.
Type Parameters

T
Remarks

Since an arbitrary BinaryPredicate is passed to this function, what is being removed need not be true equality. This method finds last item X which satisfies BinaryPredicate(X, Y), where Y is one of the items in itemsToLookFor.
See Also

[Algorithms Class](#)

[Wintellect.PowerCollections Namespace](#)

Send comments about this topic to Microsoft.
Algorithms:::LexicographicalCompare Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LexicographicalCompare</strong>&lt;<em>(Of &lt;T&gt;)</em>&gt;(IEnumerable&lt;<em>(Of &lt;T&gt;)</em>&gt;, IEnumerable&lt;<em>(Of &lt;T&gt;)</em>&gt;)</td>
<td>Performs a lexicographical comparison of two sequences of values. A lexicographical comparison compares corresponding pairs of elements from two sequences in order. If the first element of sequence1 is less than the first element of sequence2, then the comparison ends and the first sequence is lexicographically less than the second. If the first elements of each sequence are equal, then the comparison proceeds to the second element of each sequence. If one sequence is shorter than the other, but corresponding elements are all equal, then the shorter sequence is considered less than the longer one.</td>
</tr>
</tbody>
</table>

Performing a lexicographical comparison of two sequences of values, using a supplied comparer interface. A lexicographical comparison compares corresponding pairs of elements from two sequences in order. If the first element of sequence1 is less than the first element of sequence2, then the comparison ends and the first sequence is lexicographically less than the second. If the first elements of each sequence are equal, then the comparison proceeds to the second element of each sequence. If one sequence is shorter than the other, but corresponding elements are all equal, then the shorter sequence is considered less than the longer one. |

Performing a lexicographical comparison of two sequences of values, using a supplied comparer interface. A lexicographical comparison compares corresponding pairs of elements from two sequences in order. If the first element of sequence1 is less than the first element of sequence2, then the comparison ends and the first sequence is lexicographically less than the second. If the first elements of each sequence are equal, then the comparison proceeds to the second element of each sequence. If one sequence is shorter than the other, but corresponding elements are all equal, then the shorter sequence is considered less than the longer one. |
LexicographicalCompare<(Of <T>),(IEnumerable(Of <T>), IEnumerable(Of <T>), Comparison(Of <T>))>

comparison delegate. A lexicographical comparison compares corresponding pairs of elements from two sequences in order. If the first element of sequence1 is less than the first element of sequence2, then the comparison ends and the first sequence is lexicographically less than the second. If the first elements of each sequence are equal, then the comparison proceeds to the second element of each sequence. If one sequence is shorter than the other, but corresponding elements are all equal, then the shorter sequence is considered less than the longer one.
See Also

- [Algorithms Class](#)
- [Algorithms Members](#)
- [Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Algorithms::LexicographicalCompare(Of <T>*) Method (IEnumerable(Of <T>), IEnumerable(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Performs a lexicographical comparison of two sequences of values. A lexicographical comparison compares corresponding pairs of elements from two sequences in order. If the first element of sequence1 is less than the first element of sequence2, then the comparison ends and the first sequence is lexicographically less than the second. If the first elements of each sequence are equal, then the comparison proceeds to the second element of each sequence. If one sequence is shorter than the other, but corresponding elements are all equal, then the shorter sequence is considered less than the longer one.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static int LexicographicalCompare<T>(
    IEnumerable<T> sequence1,
    IEnumerable<T> sequence2
) where T : IComparable<T>

Visual Basic (Declaration)

Public Shared Function LexicographicalCompare(Of T As IComparable(Of T), _
    sequence1 As IEnumerable(Of T), _
    sequence2 As IEnumerable(Of T) _) As Integer

Visual C++

public:
    generic<typename T>
    where T : IComparable<T>
    static int LexicographicalCompare (/
        IEnumerable<T>^ sequence1,
        IEnumerable<T>^ sequence2
    )

Parameters

sequence1
    IEnumerable(Of T)>
    The first sequence to compare.

sequence2
    IEnumerable(Of T)>
    The second sequence to compare.

Return Value

Less than zero if sequence1 is lexicographically less than sequence2. Greater
than zero if \( \text{sequence1} \) is lexicographically greater than \( \text{sequence2} \). Zero if \( \text{sequence1} \) is equal to \( \text{sequence2} \).
Type Parameters

T

Types of items to compare. This type must implement IComparable<T> to allow items to be compared.
Remarks

T must implement either IComparable<T> and this implementation is used to compare the items.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::NotSupportedException</code></td>
<td>T does not implement <code>IComparable&lt;T&gt;</code> or <code>IComparable</code>.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::LexicographicalCompare<(Of <T>))> Method (IEnumerable<(Of <T>)), IEnumerable<(Of <T>)), IComparer<(Of <T>))>

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Performs a lexicographical comparison of two sequences of values, using a supplied comparer interface. A lexicographical comparison compares corresponding pairs of elements from two sequences in order. If the first element of sequence1 is less than the first element of sequence2, then the comparison ends and the first sequence is lexicographically less than the second. If the first elements of each sequence are equal, then the comparison proceeds to the second element of each sequence. If one sequence is shorter than the other, but corresponding elements are all equal, then the shorter sequence is considered less than the longer one.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static int LexicographicalCompare<T>(
    IEnumerable<T> sequence1,
    IEnumerable<T> sequence2,
    IComparer<T> comparer
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function LexicographicalCompare(Of T) (_
    sequence1 As IEnumerable(Of T), _
    sequence2 As IEnumerable(Of T), _
    comparer As IComparer(Of T) _) As Integer
)
```

Visual C++

```cpp
public:
    generic<typename T>
    static int LexicographicalCompare (  
        IEnumerable<T>^ sequence1,  
        IEnumerable<T>^ sequence2,  
        IComparer<T>^ comparer  
    )
```

Parameters

sequence1
    IEnumerable<(Of <T>)>  
    The first sequence to compare.

sequence2
    IEnumerable<(Of <T>)>  
    The second sequence to compare.

comparer
**IComparer<T]**
The IComparer<T> used to compare items. Only the Compare member function of this interface is called.

**Return Value**

Less than zero if sequence1 is lexicographically less than sequence2. Greater than zero if sequence1 is lexicographically greater than sequence2. Zero if sequence1 is equal to sequence2.
Type Parameters

T

Types of items to compare.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>sequence1, sequence2, or comparer is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Perform a lexicographical comparison of two sequences of values, using a supplied comparison delegate. A lexicographical comparison compares corresponding pairs of elements from two sequences in order. If the first element of sequence1 is less than the first element of sequence2, then the comparison ends and the first sequence is lexicographically less than the second. If the first elements of each sequence are equal, then the comparison proceeds to the second element of each sequence. If one sequence is shorter than the other, but corresponding elements are all equal, then the shorter sequence is considered less than the longer one.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static int LexicographicalCompare<T>(
        IEnumerable<T> sequence1,
        IEnumerable<T> sequence2,
        Comparison<T> comparison
    )

Visual Basic (Declaration)

Public Shared Function LexicographicalCompare(Of T) ( _
    sequence1 As IEnumerable(Of T), _
    sequence2 As IEnumerable(Of T), _
    comparison As Comparison(Of T) _
) As Integer

Visual C++

public:
    generic<typename T>
    static int LexicographicalCompare ( 
        IEnumerable<T>^ sequence1,
        IEnumerable<T>^ sequence2,
        Comparison<T>^ comparison
    )

Parameters

sequence1
    IEnumerable<(Of <T>)> The first sequence to compare.

sequence2
    IEnumerable<(Of <T>)> The second sequence to compare.

comparison
**Comparison<Of <T>>**

The IComparison<T> delegate to compare items. Only the Compare member function of this interface is called.

**Return Value**

Less than zero if sequence1 is lexicographically less than sequence2. Greater than zero if sequence1 is lexicographically greater than sequence2. Zero if sequence1 is equal to sequence2.
Type Parameters

T

Types of items to compare.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::Maximum Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Maximum&lt;(Of &lt;T&gt;)&gt;</code> (IEnumerable&lt;(Of &lt;T&gt;)&gt; )</td>
<td>Finds the maximum value in a collection.</td>
</tr>
<tr>
<td><code>Maximum&lt;(Of &lt;T&gt;)&gt;</code> (IEnumerable&lt;(Of &lt;T&gt;)&gt; , IComparer&lt;(Of &lt;T&gt;)&gt; )</td>
<td>Finds the maximum value in a collection. A supplied IComparer&lt;T&gt; is used to compare the items in the collection.</td>
</tr>
<tr>
<td><code>Maximum&lt;(Of &lt;T&gt;)&gt;</code> (IEnumerable&lt;(Of &lt;T&gt;)&gt; , Comparison&lt;(Of &lt;T&gt;)&gt; )</td>
<td>Finds the maximum value in a collection. A supplied Comparison&lt;T&gt; delegate is used to compare the items in the collection.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Finds the maximum value in a collection.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
- Syntax

C#

```csharp
public static T Maximum<T>(
    IEnumerable<T> collection
) where T : IComparable<T>
```

Visual Basic (Declaration)

```vbs
Public Shared Function Maximum(Of T As IComparable(Of T)) ( _
    collection As IEnumerable(Of T) _
) As T
```

Visual C++

```cpp
public:
    generic<typename T>
    where T : IComparable<T>
    static T Maximum(
        IEnumerable<T>^ collection
    )
```

Parameters

collection
    IEnumerable<T>
    The collection to search.

Return Value

The largest item in the collection.
Type Parameters

T

The type of items in the collection.
Remarks

Values in the collection are compared by using the IComparable<T> interfaces implementation on the type T.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>The collection is empty.</td>
</tr>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#)
[Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Algorithms...:::Maximum(Of <T> )> Method (IEnumerable(Of <T> ), IComparer(Of <T> ))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the maximum value in a collection. A supplied IComparer<T> is used to compare the items in the collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```
public static T Maximum<T>(
    IEnumerable<T> collection,
    IComparer<T> comparer
)
```

Visual Basic (Declaration)

```
Public Shared Function Maximum(Of T) ( _
    collection As IEnumerable(Of T), _
    comparer As IComparer(Of T) _
) As T
```

Visual C++

```
public:
    generic<typename T>
    static T Maximum (n
    IEnumerable<T>^ collection,
    IComparer<T>^ comparer
    )
```

Parameters

collection
    IEnumerable<(Of <T>)>
    The collection to search.

comparer
    IComparer<(Of <T>)>
    The comparer instance used to compare items in the collection.

Return Value

The largest item in the collection.
Type Parameters

T
The type of items in the collection.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.InvalidOperationException</td>
<td>The collection is empty.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>collection or comparer is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:: Maximum(Of <T>)> Method (IEnumerable(Of <T>), Comparison(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the maximum value in a collection. A supplied Comparison<T> delegate is used to compare the items in the collection.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static T Maximum<T>(
    IEnumerable<T> collection,
    Comparison<T> comparison
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function Maximum(Of T) ( _
    collection As IEnumerable(Of T), _
    comparison As Comparison(Of T) _
) As T
```

**Visual C++**

```csharp
public:
    generic<typename T>
    static T Maximum ( 
        IEnumerable<T>^ collection,
        Comparison<T>^ comparison
    )
```

### Parameters

**collection**

`IEnumerable<(Of <T>)>`

The collection to search.

**comparison**

`Comparison<(Of <T>)>`

The comparison used to compare items in the collection.

### Return Value

The largest item in the collection.
Type Parameters

T
The type of items in the collection.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::InvalidOperationException</td>
<td>The collection is empty.</td>
</tr>
<tr>
<td>System..::ArgumentNullException</td>
<td>collection or comparison is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...::MergeSorted Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `MergeSorted(Of T)>
  (array(IEnumerable(Of T)))[]>[]()` | Merge several sorted collections into a single sorted collection. Each input collection must be sorted by the natural ordering of the type (it's implementation of IComparable<T>). The merging is stable; equal items maintain their ordering, and equal items in different collections are placed in the order of the collections. |
| `MergeSorted(Of T)>
  (Comparer(Of T>>, array(IEnumerable(Of T)))[]>[]()` | Merge several sorted collections into a single sorted collection. Each input collection must be sorted by the ordering in the passed Comparison<T> delegate. The merging is stable; equal items maintain their ordering, and equal items in different collections are placed in the order of the collections. |
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::MergeSorted(Of <T>)-> Method (array(IEnumerable(Of <T>))->[])

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Merge several sorted collections into a single sorted collection. Each input collection must be sorted by the natural ordering of the type (it's implementation of IComparable<T>). The merging is stable; equal items maintain their ordering, and equal items in different collections are placed in the order of the collections.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEnumerable<T> MergeSorted<T>(
    params IEnumerable<T>[] collections
) where T : IComparable<T>
```

Visual Basic (Declaration)

```vbnet
Public Shared Function MergeSorted(Of T As IComparable(Of T)) ( _
    ParamArray collections As IEnumerable(Of T)() _
) As IEnumerable(Of T)
```

Visual C++

```cpp
public:
    generic<typename T>
    where T : IComparable<T>
    static IEnumerable<T>^ MergeSorted ( _
        ... array<IEnumerable<T>^>^ collections
    )
```

Parameters

- **collections**: `array<IEnumerable<T>>(Of <T>)>[]>()`
  - The set of collections to merge. In many languages, this parameter can be specified as several individual parameters.

Return Value

An `IEnumerable<T>` that enumerates all the items in all the collections in sorted order.
Type Parameters

T
See Also

Algorithms Class

Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::MergeSorted<(Of <T>), Method (IComparer<(Of <T>)>, array<IEEnumerable<(Of <T>)>[]>>()

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Merge several sorted collections into a single sorted collection. Each input collection must be sorted by the ordering in the passed instance of IComparer<T>. The merging is stable; equal items maintain their ordering, and equal items in different collections are placed in the order of the collections.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static IEnumerable<T> MergeSorted<T>(
    IComparer<T> comparer,
    params IEnumerable<T>[] collections
)

Visual Basic (Declaration)

Public Shared Function MergeSorted(Of T) ( _
    comparer As IComparer(Of T), _
    ParamArray collections As IEnumerable(Of T)() _
) As IEnumerable(Of T)

Visual C++

public:
    generic<typename T>
    static IEnumerable<T>^ MergeSorted ( 
        IComparer<T>^ comparer, 
        ... array IEnumerable<T>^[] collections
    )

Parameters

comparer
    IComparer<Of <T>>
    The comparer instance used to sort the list. Only the Compare method is used.

collections
    array(IEnumerable<Of <T>>[])
    The set of collections to merge. In many languages, this parameter can be specified as several individual parameters.

Return Value
An IEnumerable<T> that enumerates all the items in all the collections in sorted order.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::MergeSorted(Of <T>) Method (Comparison(Of <T>), array<IEnumerable(Of <T>>>[]))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Merge several sorted collections into a single sorted collection. Each input collection must be sorted by the ordering in the passed Comparison<T> delegate. The merging is stable; equal items maintain their ordering, and equal items in different collections are placed in the order of the collections.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

public static IEnumerable<T> MergeSorted<T>(
    Comparison<T> comparison,
    params IEnumerable<T>[] collections
)

**Visual Basic (Declaration)**

Public Shared Function MergeSorted(Of T) ( _
    comparison As Comparison(Of T), _
    ParamArray collections As IEnumerable(Of T)() _
) As IEnumerable(Of T)

**Visual C++**

public: 
    generic<typename T> 
    static IEnumerable<T>^ MergeSorted ( 
        Comparison<T>^ comparison, 
        ... array IEnumerable<T>[]^ collections
    )

**Parameters**

**comparison**

`Comparison<Of <T>>` 
The comparison delegate used to sort the collections.

**collections**

`array<IEnergyble<Of <T>>>[]()` 
The set of collections to merge. In many languages, this parameter can be specified as several individual parameters.

**Return Value**
An IEnumerable<T> that enumerates all the items in all the collections in sorted order.
Type Parameters

T
See Also

[Algorithms Class](#)
[Wintellect.PowerCollections](#) Namespace

Send comments about this topic to Microsoft.
Algorithms...::Minimum Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum&lt;(Of &lt;T&gt;))&gt; (IEnumerable&lt;(Of &lt;T&gt;))&gt;</td>
<td>Finds the minimum value in a collection.</td>
</tr>
<tr>
<td>Minimum&lt;(Of &lt;T&gt;))&gt;</td>
<td>Finds the minimum value in a collection. A supplied IComparer&lt;T&gt; is used to compare the items in the collection.</td>
</tr>
<tr>
<td>Minimum&lt;(Of &lt;T&gt;))&gt; (IEnumerable&lt;(Of &lt;T&gt;)), IComparer&lt;(Of &lt;T&gt;))&gt;</td>
<td>Finds the minimum value in a collection. A supplied Comparison&lt;T&gt; delegate is used to compare the items in the collection.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#)
[Algorithms Members](#)
[Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Algorithms...:::Minimum(Of <T>)) Method (IEnumerable(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the minimum value in a collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static T Minimum<T>(
    IEnumerable<T> collection
) where T : IComparable<T>

Visual Basic (Declaration)

Public Shared Function Minimum(Of T As IComparable(Of T)) ( _
    collection As IEnumerable(Of T) _
) As T

Visual C++

public:
    generic<typename T>
    where T : IComparable<T>
    static T Minimum ( _
        IEnumerable<T>^ collection
    )

Parameters

collection
    IEnumerable<(Of <T>)> The collection to search.

Return Value

The smallest item in the collection.
Type Parameters

\[ T \]

The type of items in the collection.
Remarks

Values in the collection are compared by using the IComparable<T> interfaces implementation on the type T.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::InvalidOperationException</td>
<td>The collection is empty.</td>
</tr>
<tr>
<td>System::ArgumentNullException</td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::Minimum(Of <T>)(IEnumerable(Of <T>), IComparer(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the minimum value in a collection. A supplied IComparer<T> is used to compare the items in the collection.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static T Minimum<T>(
    IEnumerable<T> collection,
    IComparer<T> comparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function Minimum(Of T) ( _
    collection As IEnumerable(Of T), _
    comparer As IComparer(Of T) _) As T
```

**Visual C++**

```csharp
public:
    generic<typename T>
    static T Minimum ( 
        IEnumerable<T>^ collection, 
        IComparer<T>^ comparer
    )
```

**Parameters**

collection

    IEnumerable<(Of <T>)>
    The collection to search.

comparer

    IComparer<(Of <T>)>
    The comparer instance used to compare items in the collection.

**Return Value**

The smallest item in the collection.
Type Parameters

T

The type of items in the collection.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>The collection is empty.</td>
</tr>
<tr>
<td><code>System::::ArgumentNullException</code></td>
<td>collection or comparer is null.</td>
</tr>
</tbody>
</table>
See Also

*Algorithms* Class  
*Wintellect.PowerCollections* Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms...:::Minimum<Of <T>> Method (IEnumerable<Of <T>>, Comparison<Of <T>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the minimum value in a collection. A supplied Comparison<T> delegate is used to compare the items in the collection.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public static T Minimum<T>(
    IEnumerable<T> collection,
    Comparison<T> comparison
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function Minimum(Of T) ( _
    collection As IEnumerable(Of T), _
    comparison As Comparison(Of T) _
) As T
```

**Visual C++**

```csharp
public:
    generic<typename T>
static T Minimum ( _
    IEnumerable<T>^ collection,
    Comparison<T>^ comparison
)
```

**Parameters**

collection
    `IEnumerable<T>`
    The collection to search.

collection
    `Comparison<T>`
    The comparison used to compare items in the collection.

**Return Value**

The smallest item in the collection.
Type Parameters

T
  The type of items in the collection.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>The collection is empty.</td>
</tr>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>collection or comparison is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::NCopiesOf<Of<T>> Method

See Also  Example

See Also  Example

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates an IEnumerator that enumerates a given item n times.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEnumerable<T> NCopiesOf<T>(
    int n,
    T item
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function NCopiesOf(Of T) ( _
    n As Integer, _
    item As T _
) As IEnumerable(Of T)
```

Visual C++

```cpp
public:
    generic<typename T>
    static IEnumerable<T>^ NCopiesOf ( 
        int n,
        T item
    )
```

Parameters

n

[Int32]
The number of times to enumerate the item.

item

[T]
The item that should occur in the enumeration.

Return Value

An IEnumerable<T> that yields n copies of item.
Type Parameters

T
Examples

The following creates a list consisting of 1000 copies of the double 1.0.

Copy Code

```csharp
List<double> list = new List<double>(Algorithms.NCopiesOf(1000, 1.0));
```
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>The argument n is less than zero.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Partition a list or array based on a predicate. After partitioning, all items for which the predicate returned true precede all items for which the predicate returned false.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static int Partition<T>(
    IList<T> list,
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

Public Shared Function Partition(Of T) ( _
    list As IList(Of T), _
    predicate As Predicate(Of T) _
) As Integer

**Visual C++**

```csharp
public:
    generic<typename T>
    static int Partition ( 
        IList<T>^ list,
        Predicate<T>^ predicate
    )
```

### Parameters

**list**

- **IList<(Of <T)>**
  - The list or array to partition.

**predicate**

- **Predicate<(Of <T)>**
  - A delegate that defines the partitioning condition.

### Return Value

The index of the first item in the second half of the partition; i.e., the first item
for which predicate returned false. If the predicate was true for all items in the list, list.Count is returned.
Type Parameters

T
Remarks

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::RandomShuffle Method

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RandomShuffle&lt;Of T&gt;() (IEnumerable&lt;Of T&gt;())</td>
<td>Randomly shuffles the items in a collection, yielding a new collection.</td>
</tr>
<tr>
<td>RandomShuffle&lt;Of T&gt;() (IEnumerable&lt;Of T&gt;(), Random)</td>
<td>Randomly shuffles the items in a collection, yielding a new collection.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#) [Algorithms Members](#) [Wintellect.PowerCollections Namespace](#)

Send comments about this topic to Microsoft.
Randomly shuffles the items in a collection, yielding a new collection.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static T[] RandomShuffle<T>(
    IEnumerable<T> collection
)

Visual Basic (Declaration)

Public Shared Function RandomShuffle(Of T) ( _
    collection As IEnumerable(Of T) _
) As T()

Visual C++

public:
    generic<typename T>
    static array<T>^ RandomShuffle ( _
        IEnumerable<T>^ collection
    )

Parameters

collection
    IEnumerable<Of <T>>
The collection to shuffle.

Return Value

An array with the same size and items as collection, but the items in a randomly chosen order.
Type Parameters

T

The type of the items in the collection.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Randomly shuffles the items in a collection, yielding a new collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public static T[] RandomShuffle<T>(
    IEnumerable<T> collection,
    Random randomGenerator
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function RandomShuffle(Of T) ( _
    collection As IEnumerable(Of T), _
    randomGenerator As Random _
) As T()
```

**Visual C++**

```csharp
public:
    generic<typename T>
static array<T>^ RandomShuffle ( 
    IEnumerable<T>^ collection,
    Random^ randomGenerator
)
```

**Parameters**

collection

*IEnumerable<Of <T>>*

The collection to shuffle.

randomGenerator

*Random*

The random number generator to use to select the random order.

**Return Value**

An array with the same size and items as collection, but the items in a randomly
chosen order.
**Type Parameters**

T

The type of the items in the collection.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...::RandomShuffleInPlace Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RandomShuffleInPlace(Of&lt;T&gt;)</code>&lt;br&gt;(IList(Of&lt;T&gt;))</td>
<td>Randomly shuffles the items in a list or array, in place.</td>
</tr>
<tr>
<td><code>RandomShuffleInPlace(Of&lt;T&gt;)</code>&lt;br&gt;(IList(Of&lt;T&gt;), Random)</td>
<td>Randomly shuffles the items in a list or array, in place.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Randomly shuffles the items in a list or array, in place.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
C#

```csharp
public static void RandomShuffleInPlace<T>(
    IList<T> list
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Sub RandomShuffleInPlace(Of T) ( _
    list As IList(Of T) _
)
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static void RandomShuffleInPlace ( 
        IList<T>^ list
    )
```

**Parameters**

`list`  
`IList<Of <T>>`  
The list or array to shuffle.
Type Parameters

T
Remarks

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Randomly shuffles the items in a list or array, in place.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static void RandomShuffleInPlace<T>(
    IList<T> list,
    Random randomGenerator
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Sub RandomShuffleInPlace(Of T) ( _
    list As IList(Of T), _
    randomGenerator As Random _
)
```

**Visual C++**

```csharp
public:
    generic<typename T>
    static void RandomShuffleInPlace (  
        IList<T>^ list,  
        Random^ randomGenerator  
    )
```

**Parameters**

**list**

- `IList<Of<T>>`
  - The list or array to shuffle.

**randomGenerator**

- `Random`
  - The random number generator to use to select the random order.
Type Parameters

T
**Remarks**

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

[Algorithms](#) Class  
[Wintellect.PowerCollections](#) Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms::<RandomSubset Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RandomSubset&lt;Of T&gt; (IEnumerable&lt;Of T&gt;, Int32)</td>
<td>Picks a random subset of count items from collection, and places those items into a random order. No item is selected more than once.</td>
</tr>
<tr>
<td>RandomSubset&lt;Of T&gt; (IEnumerable&lt;Of T&gt;, Int32, Random)</td>
<td>Picks a random subset of count items from collection, and places those items into a random order. No item is selected more than once.</td>
</tr>
</tbody>
</table>
See Also

 Algorithms Class
 Algorithms Members
 Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::RandomSubset<(Of <T>)) Method (IEnumerable<(Of <T>)), Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Picks a random subset of count items from collection, and places those items into a random order. No item is selected more than once.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static T[] RandomSubset<T>(
    IEnumerable<T> collection,
    int count
)

Visual Basic (Declaration)

Public Shared Function RandomSubset(Of T) ( _
    collection As IEnumerable(Of T), _
    count As Integer _
) As T()

Visual C++

public:
    generic<typename T>
    static array<T>^ RandomSubset ( 
        IEnumerable<T>^ collection,
        int count
    )

Parameters

collection
    IEnumerable<(Of <T)>>
    The collection of items to select from. This collection is not changed.

count
    Int32
    The number of items in the subset to choose.

Return Value

An array of count items, selected at random from collection.
Type Parameters

T

The type of items in the collection.
Remarks

If the collection implements IList<T>, then this method takes time $O(\text{count})$. Otherwise, this method takes time $O(N)$, where $N$ is the number of items in the collection.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>count is negative or greater than <code>collection.Count</code>.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::RandomSubset<(Of <T>)> Method (IEnumerable<(Of <T>)>, Int32, Random)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Picks a random subset of `count` items from collection, and places those items into a random order. No item is selected more than once.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static T[] RandomSubset<T>(
    IEnumerable<T> collection,
    int count,
    Random randomGenerator
)

Visual Basic (Declaration)

Public Shared Function RandomSubset(Of T) ( _
    collection As IEnumerable(Of T), _
    count As Integer, _
    randomGenerator As Random _
) As T()

Visual C++

public:
    generic<typename T>
    static array<T>^ RandomSubset ( 
        IEnumerable<T>^ collection,
        int count,
        Random^ randomGenerator
    )

Parameters

collection
    IEnumerable<(Of <T>)> The collection of items to select from. This collection is not changed.

count
    Int32 The number of items in the subset to choose.

randomGenerator
**Random**
The random number generates used to make the selection.

**Return Value**

An array of count items, selected at random from collection.
Type Parameters

T

The type of items in the collection.
Remarks

If the collection implements IList<T>, then this method takes time \(O(\text{count})\). Otherwise, this method takes time \(O(N)\), where \(N\) is the number of items in the collection.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentOutOfRangeException</code></td>
<td>count is negative or greater than <code>collection.Count</code></td>
</tr>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td><code>randomGenerator</code> is null</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::Range Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Range&lt;(Of &lt;T&gt;)&gt;</strong>&lt;sup&gt;S&lt;/sup&gt; <strong>(IList&lt;(Of &lt;T&gt;)&gt;, Int32, Int32)</strong></td>
<td>Returns a view onto a sub-range of a list. Items from list are not copied; the returned IList&lt;T&gt; is simply a different view onto the same underlying items. Changes to list are reflected in the view, and vice versa. Insertions and deletions in the view change the size of the view, but insertions and deletions in the underlying list do not.</td>
</tr>
<tr>
<td><strong>Range&lt;(Of &lt;T&gt;)&gt;</strong>&lt;sup&gt;S&lt;/sup&gt; <strong>(array&lt;T&gt;[][], Int32, Int32)</strong></td>
<td>Returns a view onto a sub-range of an array. Items from array are not copied; the returned IList&lt;T&gt; is simply a different view onto the same underlying items. Changes to array are reflected in the view, and vice versa. Insertions and deletions in the view change the size of the view. After an insertion, the last item in array &quot;falls off the end&quot;. After a deletion, the last item in array becomes the default value (0 or null).</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Returns a view onto a sub-range of a list. Items from list are not copied; the returned IList<T> is simply a different view onto the same underlying items. Changes to list are reflected in the view, and vice versa. Insertions and deletions in the view change the size of the view, but insertions and deletions in the underlying list do not.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

generic<T>

public static <T> Range<T>(
    <T> list,
    int start,
    int count
)

Visual Basic (Declaration)

Public Shared Function Range(Of T) ( _
    list As <T>, _
    start As Integer, _
    count As Integer _
) As <T>

Visual C++

public:

generic<

static <T>^ Range ( _
    ^ list,
    int start,
    int count
)

Parameters

list

<T> list,

The list to view.

start

Int32

The starting index of the view.

count
Int32
The number of items in the view.

Return Value

A list that is a view onto the given sub-list.
Type Parameters

T

The type of the items in the list.
Remarks

This method can be used to apply an algorithm to a portion of a list. For example:

```csharp
Copy Code

Algorithms.ReverseInPlace(Algorithms.Range(list, 3, 6))
```

will reverse the 6 items beginning at index 3.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>list is null.</td>
</tr>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>start or count is negative.</td>
</tr>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>start + count is greater than the size of list.</td>
</tr>
</tbody>
</table>
See Also

[Algorithm]s Class
[Wintellect.PowerCollections] Namespace

Send [comments] about this topic to Microsoft.
Algorithms::Range<Of <T>> Method (array<T>[], Int32, Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a view onto a sub-range of an array. Items from array are not copied; the returned IList<T> is simply a different view onto the same underlying items. Changes to array are reflected in the view, and vice versa. Insertions and deletions in the view change the size of the view. After an insertion, the last item in array "falls off the end". After a deletion, the last item in array becomes the default value (0 or null).

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static IList<T> Range<T>(
    T[] array,
    int start,
    int count
)

Visual Basic (Declaration)

Public Shared Function Range(Of T) ( _
    array As T(), _
    start As Integer, _
    count As Integer _
) As IList(Of T)

Visual C++

public:
    generic<typename T>
    static IList<T>^ Range ( 
        array<T>^ array, 
        int start, 
        int count 
    )

Parameters

array
    array<T>[][]() 
    The array to view.

start
    Int32 
    The starting index of the view.

count
**Int32**  
The number of items in the view.

**Return Value**

A list that is a view onto the given sub-array.
Type Parameters

T
Remarks

This method can be used to apply an algorithm to a portion of an array. For example:

```csharp
Algorithms.ReverseInPlace(Algorithms.Range(array, 3, 6))
```

will reverse the 6 items beginning at index 3.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::ArgumentNullException</td>
<td>array is null.</td>
</tr>
<tr>
<td>System..::ArgumentOutOfRangeException</td>
<td>start or count is negative.</td>
</tr>
<tr>
<td>System..::ArgumentOutOfRangeException</td>
<td>start + count is greater than the size of array.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms](#) Class
[Winellect.PowerCollections](#) Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms::ReadOnly Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ReadOnly&lt;OF T&gt;() (ICollection&lt;OF T&gt;())</code></td>
<td>Returns a read-only view onto a collection. The returned <code>ICollection&lt;T&gt;</code> interface only allows operations that do not change the collection: <code>GetEnumerator</code>, <code>Contains</code>, <code>CopyTo</code>, <code>Count</code>. The <code>ReadOnly</code> property returns false, indicating that the collection is read-only. All other methods on the interface throw a <code>NotSupportedException</code>.</td>
</tr>
<tr>
<td><code>ReadOnly&lt;OF TKey, TValue&gt;() (IDictionary&lt;OF TKey, TValue&gt;())</code></td>
<td><code>ReadOnly&lt;TKey, TValue&gt;()</code> interface only allows operations that do not change the dictionary. The <code>IsReadOnly</code> property returns true, indicating that the dictionary is read-only. All other methods on the interface throw a <code>NotSupportedException</code>.</td>
</tr>
<tr>
<td><code>ReadOnly&lt;OF T&gt;() (IList&lt;OF T&gt;())</code></td>
<td>Returns a read-only view onto a list. The returned <code>IList&lt;T&gt;</code> interface only allows operations that do not change the list: <code>GetEnumerator</code>, <code>Contains</code>, <code>CopyTo</code>, <code>Count</code>, <code>IndexOf</code>, and the get accessor of the indexer. The <code>IsReadOnly</code> property returns true, indicating that the list is read-only. All other methods on the interface throw a <code>NotSupportedException</code>.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#)
[Algorithms Members](#)
[Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Algorithms...:::ReadOnly(Of T)> Method (ICollection(Of T)>

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a read-only view onto a collection. The returned ICollection<T> interface only allows operations that do not change the collection: GetEnumerator, Contains, CopyTo, Count. The ReadOnly property returns false, indicating that the collection is read-only. All other methods on the interface throw a NotSupportedException.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static ICollection<T> ReadOnly<T>(
    ICollection<T> collection
)

Visual Basic (Declaration)

Public Shared Function ReadOnly(Of T) ( _
    collection As ICollection(Of T) _
) As ICollection(Of T)

Visual C++

public:
    generic<typename T>
    static ICollection<T>^ ReadOnly ( 
        ICollection<T>^ collection
    )

Parameters

collection
    ICollection(Of T>)
    The collection to wrap.

Return Value

A read-only view onto collection. If collection is null, then null is returned.
Type Parameters

T
The type of items in the collection.
Remarks

The data in the underlying collection is not copied. If the underlying collection is changed, then the read-only view also changes accordingly.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...::ReadOnly<(Of <TKey, TValue>)> Method (IDictionary<(Of <TKey, TValue>)>)

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a read-only view onto a dictionary. The returned IDictionary<TKey,TValue> interface only allows operations that do not change the dictionary. The IsReadOnly property returns true, indicating that the dictionary is read-only. All other methods on the interface throw a NotSupportedException.

Namespace:  Wintellect.PowerCollections
Assembly:  PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static IDictionary<TKey, TValue> ReadOnly<TKey, TValue>(
    IDictionary<TKey, TValue> dictionary
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function ReadOnly(Of TKey, TValue) ( _
    dictionary As IDictionary(Of TKey, TValue) _
) As IDictionary(Of TKey, TValue)
```

**Visual C++**

```cpp
public:
    generic<typename TKey, typename TValue>
    static IDictionary<TKey, TValue>* ReadOnly ( 
        IDictionary<TKey, TValue>* dictionary
    )
```

**Parameters**

.dictionary
    IDictionary(Of <TKey, TValue>)
    The dictionary to wrap.

**Return Value**

A read-only view onto dictionary. Returns null if dictionary is null. If dictionary is already read-only, returns dictionary.
- Type Parameters
  TKey
  TValue
Remarks

The data in the underlying dictionary is not copied. If the underlying dictionary is changed, then the read-only view also changes accordingly.
See Also

Algorithms Class  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::ReadOnly(Of T) > Method (IList(Of T) >)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a read-only view onto a list. The returned IList<T> interface only allows operations that do not change the list: GetEnumerator, Contains, CopyTo, Count, IndexOf, and the get accessor of the indexer. The IsReadOnly property returns true, indicating that the list is read-only. All other methods on the interface throw a NotSupportedException.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IList<T> ReadOnly<T>(
    IList<T> list
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function ReadOnly(Of T) ( _
    list As IList(Of T) _
) As IList(Of T)
```

Visual C++

```cpp
public:
    generic<typename T>
    static IList<T>^ ReadOnly (  
        IList<T>^ list
    )
```

Parameters

list
    IList<(Of <T>)>
    The list to wrap.

Return Value

A read-only view onto list. Returns null if list is null. If list is already read-only, returns list.
Type Parameters

T

The type of items in the list.
Remarks

The data in the underlying list is not copied. If the underlying list is changed, then the read-only view also changes accordingly.
See Also

- [Algorithms Class](#)
- [Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Algorithms...:::ReadWriteList<Of <T>> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a read-write IList<T> wrapper around an array. When an array is implicitly converted to an IList<T>, changes to the items in the array cannot be made through the interface. This method creates a read-write IList<T> wrapper on an array that can be used to make changes to the array.

Use this method when you need to pass an array to an algorithms that takes an IList<T> and that tries to modify items in the list. Algorithms in this class generally do not need this method, since they have been design to operate on arrays even when they are passed as an IList<T>.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static IList<T> ReadWriteList<T>(T[] array)

Visual Basic (Declaration)

Public Shared Function ReadWriteList(Of T) ( _
    array As T() _
) As IList(Of T)

Visual C++

public:
    generic<typename T>
    static IList<T>^ ReadWriteList ( _
        array<T>^ array
    )

Parameters

array
    array<T>[]( )
    The array to wrap.

Return Value

An IList<T> wrapper onto array.
Type Parameters

T
Remarks

Since arrays cannot be resized, inserting an item causes the last item in the array to be automatically removed. Removing an item causes the last item in the array to be replaced with a default value (0 or null). Clearing the list causes all the items to be replaced with a default value.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::RemoveDuplicates Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveDuplicates&lt;Of <code>T&gt;</code>(IEnumerable&lt;Of <code>T</code>&gt;)</td>
<td>Remove consecutive equal items from a collection, yielding another collection. In each run of consecutive equal items in the collection, all items after the first item in the run are removed.</td>
</tr>
<tr>
<td>RemoveDuplicates&lt;Of <code>T&gt;</code>(IEnumerable&lt;Of <code>T</code>&gt;, IEqualityComparer&lt;Of <code>T</code>&gt;)</td>
<td>Remove consecutive equal items from a collection, yielding another collection. In each run of consecutive equal items in the collection, all items after the first item in the run are removed. A passed IEqualityComparer is used to determine equality.</td>
</tr>
<tr>
<td>RemoveDuplicates&lt;Of <code>T&gt;</code>(IEnumerable&lt;Of <code>T</code>&gt;, BinaryPredicate&lt;Of <code>T</code>&gt;)</td>
<td>Remove consecutive &quot;equal&quot; items from a collection, yielding another collection. In each run of consecutive equal items in the collection, all items after the first item in the run are removed. The passed BinaryPredicate is used to determine if two items are &quot;equal&quot;.</td>
</tr>
</tbody>
</table>
See Also

- Algorithms Class
- Algorithms Members
- Wintellect.PoweCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::RemoveDuplicates<(Of <T>)> Method (IEnumerable<(Of <T>)>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Remove consecutive equal items from a collection, yielding another collection. In each run of consecutive equal items in the collection, all items after the first item in the run are removed.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static IEnumerable<T> RemoveDuplicates<T>(
    IEnumerable<T> collection
)

Visual Basic (Declaration)

Public Shared Function RemoveDuplicates(Of T) ( _
    collection As IEnumerable(Of T) _
) As IEnumerable(Of T)

Visual C++

public:
    generic<typename T>
    static IEnumerable<T>^ RemoveDuplicates ( 
        IEnumerable<T>^ collection
    )

Parameters

collection
    IEnumerable(Of T>)
        The collection to process.

Return Value

An new collection with the items from collection, in the same order, with consecutive duplicates removed.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentNullException</td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
RemoveDuplicates(Of T) Method (IEnumerable(Of T), IEqualityComparer(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Remove consecutive equal items from a collection, yielding another collection. In each run of consecutive equal items in the collection, all items after the first item in the run are removed. A passed IEqualityComparer is used to determine equality.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public static IEnumerable<T> RemoveDuplicates<T>(
    IEnumerable<T> collection,
    IEqualityComparer<T> equalityComparer
)
```

#### Visual Basic (Declaration)

```vbnet
Public Shared Function RemoveDuplicates(Of T) ( _
    collection As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
) As IEnumerable(Of T)
```

#### Visual C++

```cpp
public:
    generic<typename T>
    static IEnumerable<T>^ RemoveDuplicates ( _
        IEnumerable<T>^ collection,
        IEqualityComparer<T>^ equalityComparer
    )
```

### Parameters

- **collection**
  - `IEnumerable<T>`
  - The collection to process.

- **equalityComparer**
  - `IEqualityComparer<T>`
  - The `IEqualityComparer<T>` used to compare items for equality. Only the `Equals` method will be called.

### Return Value
An new collection with the items from collection, in the same order, with consecutive duplicates removed.
Type Parameters

T
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>collection or equalityComparer is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Remove consecutive "equal" items from a collection, yielding another collection. In each run of consecutive equal items in the collection, all items after the first item in the run are removed. The passed BinaryPredicate is used to determine if two items are "equal".

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEnumerable<T> RemoveDuplicates<T>(
    IEnumerable<T> collection,
    BinaryPredicate<T> predicate
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function RemoveDuplicates(Of T) ( _
    collection As IEnumerable(Of T), _
    predicate As BinaryPredicate(Of T) _
) As IEnumerable(Of T)
```

Visual C++

```cpp
public:
    generic<typename T>
    static IEnumerable<T>^ RemoveDuplicates ( 
        IEnumerable<T>^ collection, 
        BinaryPredicate<T>^ predicate 
    )
```

Parameters

collection
    IEnumerable<Of <T)>>
    The collection to process.

predicate
    BinaryPredicate<Of <T)>>
    The BinaryPredicate used to compare items for "equality". An item current
    is removed if predicate(first, current)==true, where first is the first item in
    the group of "duplicate" items.

Return Value
An new collection with the items from collection, in the same order, with consecutive "duplicates" removed.
Type Parameters

T
Remarks

Since an arbitrary BinaryPredicate is passed to this function, what is being removed need not be true equality.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::RemoveDuplicatesInPlace Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveDuplicatesInPlace(&lt;\text{Of } \langle T \rangle)(\langle\text{IList}\langle\text{Of } \langle T \rangle\rangle)</td>
<td>Remove consecutive equal items from a list or array. In each run of consecutive equal items in the list, all items after the first item in the run are removed. The removal is done in-place, changing the list.</td>
</tr>
<tr>
<td>RemoveDuplicatesInPlace(&lt;\text{Of } \langle T \rangle)(\langle\text{IList}\langle\text{Of } \langle T \rangle\rangle), IEquivalenceComparer(&lt;\text{Of } \langle T \rangle\rangle)</td>
<td>Remove subsequent consecutive equal items from a list or array. In each run of consecutive equal items in the list, all items after the first item in the run are removed. The replacement is done in-place, changing the list. A passed IEquivalenceComparer is used to determine equality.</td>
</tr>
<tr>
<td>RemoveDuplicatesInPlace(&lt;\text{Of } \langle T \rangle)(\langle\text{IList}\langle\text{Of } \langle T \rangle\rangle), BinaryPredicate(&lt;\text{Of } \langle T \rangle\rangle)</td>
<td>Remove consecutive &quot;equal&quot; items from a list or array. In each run of consecutive equal items in the list, all items after the first item in the run are removed. The replacement is done in-place, changing the list. The passed BinaryPredicate is used to determine if two items are &quot;equal&quot;.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Remove consecutive equal items from a list or array. In each run of consecutive equal items in the list, all items after the first item in the run are removed. The removal is done in-place, changing the list.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static void RemoveDuplicatesInPlace<T>(
    IList<T> list
)

Visual Basic (Declaration)

Public Shared Sub RemoveDuplicatesInPlace(Of T) ( _
    list As IList(Of T) _
)

Visual C++

public:
    generic<typename T>
    static void RemoveDuplicatesInPlace (
        IList<T>^ list
    )

Parameters

list
    IList(Of <T>)>
    The list or array to process.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::RemoveDuplicatesInPlace(Of T) Method (IList(Of T), IEqualityComparer(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Remove subsequent consecutive equal items from a list or array. In each run of consecutive equal items in the list, all items after the first item in the run are removed. The replacement is done in-place, changing the list. A passed IEqualityComparer is used to determine equality.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static void RemoveDuplicatesInPlace<T>(
    IList<T> list,
    IEqualityComparer<T> equalityComparer
)

Visual Basic (Declaration)

Public Shared Sub RemoveDuplicatesInPlace(Of T) ( _
    list As_IList(Of T), _
    equalityComparer As _IEqualityComparer(Of T) _
)

Visual C++

public:
    generic<typename T>
    static void RemoveDuplicatesInPlace ( 
        IList<T>^ list,
        IEqualityComparer<T>^ equalityComparer
    )

Parameters

list

    IList<(Of <T>)>
    The list or array to process.

equalityComparer

    IEqualityComparer<(Of <T>)>
    The IEqualityComparer<T> used to compare items for equality. Only the
    Equals method will be called.
Type Parameters

T
Remarks

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

**Algorithms** Class
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
RemoveDuplicatesInPlace<(Of <T>)) Method (IList<(Of <T>)), BinaryPredicate<(Of <T>))>

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Remove consecutive "equal" items from a list or array. In each run of consecutive equal items in the list, all items after the first item in the run are removed. The replacement is done in-place, changing the list. The passed BinaryPredicate is used to determine if two items are "equal".

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public static void RemoveDuplicatesInPlace<T>(
    IList<T> list,
    BinaryPredicate<T> predicate
)
```

#### Visual Basic (Declaration)

```vbnet
Public Shared Sub RemoveDuplicatesInPlace(Of T) ( _
    list As IList(Of T), _
    predicate As BinaryPredicate(Of T) _
)
```

#### Visual C++

```csharp
public:
    generic<typename T>
    static void RemoveDuplicatesInPlace ( 
        IList<T>^ list,
        BinaryPredicate<T>^ predicate
    )
```

### Parameters

**list**

- `IList<Of <T>>`
- The list or array to process.

**predicate**

- `BinaryPredicate<Of <T>>`
- The BinaryPredicate used to compare items for "equality".
Type Parameters

T
Remarks

Since an arbitrary BinaryPredicate is passed to this function, what is being tested for need not be true equality.

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

**Algorithms** Class  
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:) about this topic to Microsoft.
Algorithms...:::RemoveWhere<(Of <T>)> Method

See Also

- Visual Basic (Declaration) - Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all the items in the collection that satisfy the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static ICollection<T> RemoveWhere<T>(
    ICollection<T> collection,
    Predicate<T> predicate
)

Visual Basic (Declaration)

Public Shared Function RemoveWhere(Of T) ( _
    collection As ICollection(Of T), _
    predicate As Predicate(Of T) _
) As ICollection(Of T)

Visual C++

public:
    generic<typename T>
    static ICollection<T>^ RemoveWhere ( 
        ICollection<T>^ collection,
        Predicate<T>^ predicate
    )

Parameters

collection
    ICollection<(Of <T>)>
    The collection to check all the items in.

predicate
    Predicate<(Of <T>)>
    A delegate that defines the condition to check for.

Return Value

Returns a collection of the items that were removed. This collection contains the
items in the same order that they orginally appeared in collection.
Type Parameters

T
Remarks

If the collection is an array or implements IList<T>, an efficient algorithm that compacts items is used. If not, then ICollection<T>.Remove is used to remove items from the collection. If the collection is an array or fixed-size list, the non-removed elements are placed, in order, at the beginning of the list, and the remaining list items are filled with a default value (0 or null).
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::Replace Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replace&lt;(Of &lt;T&gt;), IEnumerable&lt;(Of &lt;T&gt;)&gt;, Predicate&lt;(Of &lt;T&gt;)&gt;, T)&gt;</td>
<td>Replace all items in a collection that a predicate evaluates at true with a value, yielding another collection.</td>
</tr>
<tr>
<td>Replace&lt;(Of &lt;T&gt;), IEnumerable&lt;(Of &lt;T&gt;)&gt;, T, T)&gt;</td>
<td>Replace all items in a collection equal to a particular value with another values, yielding another collection.</td>
</tr>
<tr>
<td>Replace&lt;(Of &lt;T&gt;), IEnumerable&lt;(Of &lt;T&gt;)&gt;, T, T, IEqualityComparer&lt;(Of &lt;T&gt;)&gt;)</td>
<td>Replace all items in a collection equal to a particular value with another values, yielding another collection. A passed IEqualityComparer is used to determine equality.</td>
</tr>
</tbody>
</table>
See Also

- Algorithms Class
- Algorithms Members
- Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Replace all items in a collection that a predicate evaluates at true with a value, yielding another collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEnumerable<T> Replace<T>(
    IEnumerable<T> collection,
    Predicate<T> predicate,
    T replaceWith
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function Replace(Of T) ( _
    collection As IEnumerable(Of T), _
    predicate As Predicate(Of T), _
    replaceWith As T _
) As IEnumerable(Of T)
```

Visual C++

```cpp
public:
    generic<typename T>
    static IEnumerable<T>^ Replace ( 
        IEnumerable<T>^ collection, 
        Predicate<T>^ predicate, 
        T replaceWith
    )
```

Parameters

collection

```
IEnumerable<(Of <T>)> 
```

The collection to process.

predicate

```
Predicate<(Of <T>)> 
```

The predicate used to evaluate items with the collection. If the predicate returns true for a particular item, the item is replaces with replaceWith.
replaceWith
   T
   The new value to replace with.

Return Value

An new collection with the items from collection, in the same order, with the appropriate replacements made.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Replace all items in a collection equal to a particular value with another values, yielding another collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:**  PowerCollections (in PowerCollections.dll)
Syntax

C#

public static IEnumerable<T> Replace<T>(
    IEnumerable<T> collection,
    T itemFind,
    T replaceWith
)

Visual Basic (Declaration)

Public Shared Function Replace(Of T) ( _
    collection As IEnumerable(Of T), _
    itemFind As T, _
    replaceWith As T _
) As IEnumerable(Of T)

Visual C++

public:
    generic<typename T>
    static IEnumerable<T>^ Replace ( 
        IEnumerable<T>^ collection,
        T itemFind,
        T replaceWith
    )

Parameters

collection
    IEnumerable(Of T>)
    The collection to process.

itemFind
    T
    The value to find and replace within collection.

replaceWith
T
The new value to replace with.

**Return Value**

An new collection with the items from collection, in the same order, with the appropriate replacements made.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.
See Also

[Algorithms Class](#)
[Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Algorithms...::Replace<(Of <T>)) Method (IEnumerable<(Of <T>)), T, T, IEqualityComparer<(Of <T>))>

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Replace all items in a collection equal to a particular value with another values, yielding another collection. A passed IEqualityComparer is used to determine equality.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEnumerable<T> Replace<T>(
    IEnumerable<T> collection,
    T itemFind,
    T replaceWith,
    IEqualityComparer<T> equalityComparer)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function Replace(Of T) ( _
    collection As IEnumerable(Of T), _
    itemFind As T, _
    replaceWith As T, _
    equalityComparer As IEqualityComparer(Of T) _
) As IEnumerable(Of T)
```

Visual C++

```cpp
public:

generic<typename T>
static IEnumerable<T>^ Replace ( 
    IEnumerable<T>^ collection,
    T itemFind,
    T replaceWith,
    IEqualityComparer<T>^ equalityComparer
)
```

Parameters

collection
    IEnumerable<Of <T>>
    The collection to process.

itemFind
    T
    The value to find and replace within collection.
replaceWith
    T
    The new value to replace with.

equalityComparer
    IEqualityComparer<T>
    The IEqualityComparer<T> used to compare items for equality. Only the
    Equals method will be called.

**Return Value**

An new collection with the items from collection, in the same order, with the
appropriate replacements made.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::ReplaceInPlace Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReplaceInPlace(Of <code>T</code>)(IList(Of <code>T</code>), Predicate(Of <code>T</code>), <code>T</code>)</td>
<td>Replace all items in a list or array that a predicate evaluates at true with a value. The replacement is done in-place, changing the list.</td>
</tr>
<tr>
<td>ReplaceInPlace(Of <code>T</code>)(IList(Of <code>T</code>), <code>T</code>, <code>T</code>)</td>
<td>Replace all items in a list or array equal to a particular value with another value. The replacement is done in-place, changing the list.</td>
</tr>
<tr>
<td>ReplaceInPlace(Of <code>T</code>)(IList(Of <code>T</code>), <code>T</code>, <code>T</code>, IEqualityComparer(Of <code>T</code>))</td>
<td>Replace all items in a list or array equal to a particular value with another values. The replacement is done in-place, changing the list. A passed IEqualityComparer is used to determine equality.</td>
</tr>
</tbody>
</table>
See Also

- **Algorithms** Class
- **Algorithms** Members
- **Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
Replace all items in a list or array that a predicate evaluates at true with a value. The replacement is done in-place, changing the list.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static void ReplaceInPlace<T>(
    IList<T> list,
    Predicate<T> predicate,
    T replaceWith
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Sub ReplaceInPlace(Of T) ( _
    list As IList(Of T), _
    predicate As Predicate(Of T), _
    replaceWith As T _
)
```

Visual C++

```cpp
public:
    generic<typename T>
    static void ReplaceInPlace ( 
        IList<T>^ list,
        Predicate<T>^ predicate,
        T replaceWith
    )
```

Parameters

list

```csharp
IList<Of <T>>
```
The list or array to process.

predicate

```csharp
Predicate<Of <T>>
```
The predicate used to evaluate items with the collection. If the predicate returns true for a particular item, the item is replaces with replaceWith.
replaceWith
   T
   The new value to replace with.
Type Parameters

T
Remarks

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Replace all items in a list or array equal to a particular value with another value.
The replacement is done in-place, changing the list.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static void ReplaceInPlace<T>(
    IList<T> list,
    T itemFind,
    T replaceWith
)
```

Visual Basic (Declaration)

Public Shared Sub ReplaceInPlace(Of T) ( _
    list As IList(Of T), _
    itemFind As T, _
    replaceWith As T _
)
```

Visual C++

```cpp
public:
    generic<typename T>
    static void ReplaceInPlace ( 
        IList<T>^ list,
        T itemFind,
        T replaceWith
    )
```

Parameters

list
    IList<Of <T> >
    The list or array to process.

itemFind
    T
    The value to find and replace within .

replaceWith
T
The new value to replace with.
Type Parameters

T
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Replace all items in a list or array equal to a particular value with another values. The replacement is done in-place, changing the list. A passed IEqualityComparer is used to determine equality.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static void ReplaceInPlace<T>(
    IList<T> list,
    T itemFind,
    T replaceWith,
    IEqualityComparer<T> equalityComparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Sub ReplaceInPlace(Of T) ( _
    list As IList(Of T), _
    itemFind As T, _
    replaceWith As T, _
    equalityComparer As IEqualityComparer(Of T) _
)
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static void ReplaceInPlace ( 
        IList<T>^ list,
        T itemFind,
        T replaceWith,
        IEqualityComparer<T>^ equalityComparer
    )
```

**Parameters**

**list**

```csharp
IList<Of <T>>
```

The list or array to process.

**itemFind**

```csharp
T
```

The value to find and replace within.
replaceWith

    T
    The new value to replace with.

equalityComparer

    IEqualityComparer<T>
    The IEqualityComparer<T> used to compare items for equality. Only the
    Equals method will be called.
Type Parameters

T
Remarks

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

[Algorithms Class]
[WinEllect.PowerCollections Namespace]

Send [comments] about this topic to Microsoft.
Algorithms...:::Reverse(Of <T>)-> Method

See Also

☐ Visual Basic (Declaration) ☐ Visual Basic (Usage)
☐ C#
☐ Visual C++
☐ J#
☐ JScript
☐ XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Reverses a list and returns the reversed list, without changing the source list.

Namespace: Wintellect.PowerCollections
Assembly:  PowerCollections (in PowerCollections.dll)
Syntax

C#

public static IEnumerable<T> Reverse<T>(
    IList<T> source
)

Visual Basic (Declaration)

Public Shared Function Reverse(Of T) ( _
    source As IList(Of T) _
) As IEnumerable(Of T)

Visual C++

public:
    generic<typename T>
    static IEnumerable<T>^ Reverse ( 
        IList<T>^ source
    )

Parameters

source
    IList<Of <T>>
    The list to reverse.

Return Value

A collection that contains the items from source in reverse order.
Type Parameters

T
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>source is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::ReverseInPlace(Of <T> )> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Reverses a list or array in place.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
# Syntax

**C#**

```csharp
public static void ReverseInPlace<T>(
    IList<T> list
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Sub ReverseInPlace(Of T) ( _
    list As IList(Of T) _
)
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static void ReverseInPlace ( 
       IList<T>^ list
    )
```

## Parameters

**list**

`IList<(Of <T>)>`

The list or array to reverse.
Type Parameters

T
Remarks

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>list is null.</td>
</tr>
<tr>
<td><code>System::ArgumentException</code></td>
<td>list is read only.</td>
</tr>
</tbody>
</table>
See Also

*Algorithms* Class

*Wintellect.PowerCollections* Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
Algorithms:::Rotate(Of T)> Method

See Also

Visual Basic (Declaration) \ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Rotates a list and returns the rotated list, without changing the source list.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEnumerable<T> Rotate<T>(
    IList<T> source,
    int amountToRotate
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function Rotate(Of T) ( _
    source As IList(Of T), _
    amountToRotate As Integer _
) As IEnumerable(Of T)
```

Visual C++

```cpp
public:
    generic<typename T>
    static IEnumerable<T>^ Rotate ( 
        IList<T>^ source, 
        int amountToRotate 
    )
```

Parameters

source
    IList<(Of <T>)>
The list to rotate.

amountToRotate
    Int32
The number of elements to rotate. This value can be positive or negative. For example, rotating by positive 3 means that source[3] is the first item in the returned collection. Rotating by negative 3 means that source[source.Count - 3] is the first item in the returned collection.
Return Value

A collection that contains the items from source in rotated order.
Type Parameters

T
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...::ArgumentNullException</td>
<td>source is null.</td>
</tr>
</tbody>
</table>
See Also

- Algorithms Class
- Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms:::RotateInPlace(Of <T>) Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Rotates a list or array in place.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static void RotateInPlace<T>(
    IList<T> list,
    int amountToRotate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Sub RotateInPlace(Of T) ( _
    list As IList(Of T), _
    amountToRotate As Integer _
)
```

**Visual C++**

```csharp
public:
    generic<typename T>
    static void RotateInPlace ( 
        IList<T>^ list, 
        int amountToRotate 
    )
```

### Parameters

**list**

- `IList<Of <T>>`
  - The list or array to rotate.

**amountToRotate**

- `Int32`
  - The number of elements to rotate. This value can be positive or negative. For example, rotating by positive 3 means that list[3] is the first item in the resulting list. Rotating by negative 3 means that list[list.Count - 3] is the first item in the resulting list.
Type Parameters

T
Remarks

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>list is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::SearchForSubsequence Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SearchForSubsequence&lt;(Of &lt;T&gt;), IList&lt;(Of &lt;T&gt;), IEnumerable&lt;(Of &lt;T&gt;)&gt;)</td>
<td>Searches a list for a sub-sequence of items that match a particular pattern. A subsequence of list matches pattern at index i if list[i] is equal to the first item in pattern, list[i+1] is equal to the second item in pattern, and so forth for all the items in pattern.</td>
</tr>
<tr>
<td>SearchForSubsequence&lt;(Of &lt;T&gt;), IList&lt;(Of &lt;T&gt;), IEnumerable&lt;(Of &lt;T)&gt;), IEqualityComparer&lt;(Of &lt;T&gt;)&gt;)</td>
<td>Searches a list for a sub-sequence of items that match a particular pattern. A subsequence of list matches pattern at index i if list[i] is equal to the first item in pattern, list[i+1] is equal to the second item in pattern, and so forth for all the items in pattern. The passed instance of IEqualityComparer&lt;T&gt; is used for determining if two items are equal.</td>
</tr>
<tr>
<td>SearchForSubsequence&lt;(Of &lt;T&gt;), IList&lt;(Of &lt;T&gt;), IEnumerable&lt;(Of &lt;T&gt;)&gt;, BinaryPredicate&lt;(Of &lt;T&gt;)&gt;)</td>
<td>Searches a list for a sub-sequence of items that match a particular pattern. A subsequence of list matches pattern at index i if list[i] is &quot;equal&quot; to the first item in pattern, list[i+1] is &quot;equal&quot; to the second item in pattern, and so forth for all the items in pattern. The passed BinaryPredicate is used to determine if two items are &quot;equal&quot;.</td>
</tr>
</tbody>
</table>
See Also

**Algorithms** Class  
**Algorithms** Members  
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
Algorithms::SearchForSubsequence(Of T)> Method (IList(Of T), IEnumerable(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Searchs a list for a sub-sequence of items that match a particular pattern. A subsequence of list matches pattern at index i if list[i] is equal to the first item in pattern, list[i+1] is equal to the second item in pattern, and so forth for all the items in pattern.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static int SearchForSubsequence<T>(
    IList<T> list,
    IEnumerable<T> pattern
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function SearchForSubsequence(Of T) ( _
    list As IList(Of T), _
    pattern As IEnumerable(Of T) _
) As Integer
```

**Visual C++**

```csharp
public:
    generic<typename T>
    static int SearchForSubsequence ( 
        IList<T>^ list,
        IEnumerable<T>^ pattern
    )
```

**Parameters**

- **list**
  - `IList<Of <T>>`
  - The list to search.

- **pattern**
  - `IEnumerable<Of <T>>`
  - The sequence of items to search for.

**Return Value**

The first index with list that matches the items in pattern.
Type Parameters

T

The type of items in the list.
Remarks

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::SearchForSubsequence<(Of <T>)> Method (IList<(Of <T>)>), IEnumerable<(Of <T>)>), IEqualityComparer<(Of <T>)>)

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Searchs a list for a sub-sequence of items that match a particular pattern. A subsequence of list matches pattern at index i if list[i] is equal to the first item in pattern, list[i+1] is equal to the second item in pattern, and so forth for all the items in pattern. The passed instance of IEqualityComparer<T> is used for determining if two items are equal.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
- Syntax

**C#**

```csharp
public static int SearchForSubsequence<T>(
    IList<T> list,
    IEnumerable<T> pattern,
    IEqualityComparer<T> equalityComparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function SearchForSubsequence(Of T) (_
    list As IList(Of T), _
    pattern As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static int SearchForSubsequence (  
        IList<T>* list,  
        IEnumerable<T>* pattern,  
        IEqualityComparer<T>* equalityComparer
    )
```

**Parameters**

- **list**
  ```csharp
  IList<Of <T>>
  The list to search.
  ```

- **pattern**
  ```csharp
  IEnumerable<Of <T>>
  The sequence of items to search for.
  ```

- **equalityComparer**
**IEqualityComparer<T>**

The `IEqualityComparer<T>` used to compare items for equality. Only the `Equals` method will be called.

**Return Value**

The first index with list that matches the items in pattern.
Type Parameters

T

The type of items in the list.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Searchs a list for a sub-sequence of items that match a particular pattern. A subsequence of list matches pattern at index i if list[i] is "equal" to the first item in pattern, list[i+1] is "equal" to the second item in pattern, and so forth for all the items in pattern. The passed BinaryPredicate is used to determine if two items are "equal".

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

public static int SearchForSubsequence<T>(
    IList<T> list,
    IEnumerable<T> pattern,
    BinaryPredicate<T> predicate
)

**Visual Basic (Declaration)**

Public Shared Function SearchForSubsequence(Of T) ( _
    list As IList(Of T), _
    pattern As IEnumerable(Of T), _
    predicate As BinaryPredicate(Of T) _
) As Integer

**Visual C++**

public:
    template<typename T>
    static int SearchForSubsequence ( 
        IList<T>* list,
        IEnumerable<T>* pattern,
        BinaryPredicate<T>* predicate
    )

**Parameters**

list
    IList<T>*
    The list to search.

pattern
    IEnumerable<T>*
    The sequence of items to search for.

predicate
**BinaryPredicate**<Of <T> >
The BinaryPredicate used to compare items for "equality".

**Return Value**

The first index with list that matches the items in pattern.
Type Parameters

T

The type of items in the list.
Remarks

Since an arbitrary BinaryPredicate is passed to this function, what is being tested for in the pattern need not be equality.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::SetDifference Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SetDifference</strong>&lt;Of <code>T</code>&gt;(IEnumerable&lt;Of <code>T</code>&gt;)</td>
<td>Computes the set-theoretic difference of two collections. The difference of two sets is all items that appear in the first set, but not in the second. If an item appears X times in the first set, and Y times in the second set, the difference contains the item X - Y times (0 times if X &lt; Y). The source collections are not changed. A new collection is created with the difference of the collections; the order of the items in this collection is undefined.</td>
</tr>
<tr>
<td><strong>SetDifference</strong>&lt;Of <code>T</code>&gt;(IEnumerable&lt;Of <code>T</code>&gt;, IEqualityComparer&lt;Of <code>T</code>&gt;)</td>
<td>Computes the set-theoretic difference of two collections. The difference of two sets is all items that appear in the first set, but not in the second. If an item appears X times in the first set, and Y times in the second set, the difference contains the item X - Y times (0 times if X &lt; Y). The source collections are not changed. A new collection is created with the difference of the collections; the order of the items in this collection is undefined.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::SetDifference(Of <T>)(IEnumerable(Of <T>), IEnumerable(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the set-theoretic difference of two collections. The difference of two sets is all items that appear in the first set, but not in the second. If an item appears X times in the first set, and Y times in the second set, the difference contains the item X - Y times (0 times if X < Y). The source collections are not changed. A new collection is created with the difference of the collections; the order of the items in this collection is undefined.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
C#  

```csharp
public static IEnumerable<T> SetDifference<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function SetDifference(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T) _
) As IEnumerable(Of T)
```

**Visual C++**

```cpp
public:  
generic<typename T>  
static IEnumerable<T>^ SetDifference (  
    IEnumerable<T>^ collection1,  
    IEnumerable<T>^ collection2  
)
```

**Parameters**

- **collection1**
  - `IEnumerable<Of <T>>`
  - The first collection to difference.

- **collection2**
  - `IEnumerable<Of <T>>`
  - The second collection to difference.

**Return Value**

The difference of collection1 and collection2, considered as sets.
- **Type Parameters**

T
Remarks

When equal items appear in both collections, the returned collection will include an arbitrary choice of one of the two equal items.

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the Difference or DifferenceWith methods on that class.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms..:::SetDifference(Of <T>), Method (IEnumerable(Of <T>), IEnumerable(Of <T>), IEqualityComparer(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the set-theoretic difference of two collections. The difference of two sets is all items that appear in the first set, but not in the second. If an item appears X times in the first set, and Y times in the second set, the difference contains the item X - Y times (0 times if X < Y). The source collections are not changed. A new collection is created with the difference of the collections; the order of the items in this collection is undefined.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static IEnumerable<T> SetDifference<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2,
    IEqualityComparer<T> equalityComparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function SetDifference(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
) As IEnumerable(Of T)
```

**Visual C++**

```cpp
public:
    template<typename T>
    static IEnumerable<T>^ SetDifference ( 
        IEnumerable<T>^ collection1, 
        IEnumerable<T>^ collection2, 
        IEqualityComparer<T>^ equalityComparer 
    )
```

### Parameters

**collection1**  
`IEnumerable<T>`  
The first collection to difference.

**collection2**  
`IEnumerable<T>`  
The second collection to difference.

**equalityComparer**
**IEqualityComparer<Of<T>>**
The IEqualityComparer<T> used to compare items for equality. Only the Equals and GetHashCode member functions of this interface are called.

**Return Value**
The difference of collection1 and collection2, considered as sets.
Type Parameters

T
Remarks

When equal items appear in both collections, the returned collection will include an arbitrary choice of one of the two equal items.

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the difference or differenceWith methods on that class.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::SetIntersection Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetIntersection&lt;Of </code>T&gt;<code>&gt;(IEnumerable&lt;Of </code>T&gt;<code>), IEnumerable&lt;Of </code>T&gt;`)</td>
<td>Computes the set-theoretic intersection of two collections. The intersection of two sets is all items that appear in both of the sets. If an item appears X times in one set, and Y times in the other set, the intersection contains the item Minimum(X, Y) times. The source collections are not changed. A new collection is created with the intersection of the collections; the order of the items in this collection is undefined.</td>
</tr>
<tr>
<td><code>SetIntersection&lt;Of </code>T&gt;<code>&gt;(IEnumerable&lt;Of </code>T&gt;<code>), IEnumerable&lt;Of </code>T&gt;`)</td>
<td>Computes the set-theoretic intersection of two collections. The intersection of two sets is all items that appear in both of the sets. If an item appears X times in one set, and Y times in the other set, the intersection contains the item Minimum(X, Y) times. The source collections are not changed. A new collection is created with the intersection of the collections; the order of the items in this collection is undefined.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::SetIntersection(Of <T>)> Method (IEnumerable(Of <T>), IEnumerable(Of <T>))

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the set-theoretic intersection of two collections. The intersection of two sets is all items that appear in both of the sets. If an item appears X times in one set, and Y times in the other set, the intersection contains the item Minimum(X,Y) times. The source collections are not changed. A new collection is created with the intersection of the collections; the order of the items in this collection is undefined.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static IEnumerable<T> SetIntersection<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function SetIntersection(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T) _
) As IEnumerable(Of T)
```

**Visual C++**

```csharp
public:
    generic<typename T>
    static IEnumerable<T>^ SetIntersection ( 
    IEnumerable<T>^ collection1,
    IEnumerable<T>^ collection2
    )
```

**Parameters**

collection1

```csharp
IEnumerable<T>(Of <T>)
```

The first collection to intersect.

collection2

```csharp
IEnumerable<T>(Of <T>)
```

The second collection to intersect.

**Return Value**

The intersection of the two collections, considered as sets.
- Type Parameters

T
Remarks

When equal items appear in both collections, the returned collection will include an arbitrary choice of one of the two equal items.

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the Intersection or IntersectionWith methods on that class.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#)
[WinTellecPoweCollection](#) Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms...:::SetIntersection(Of <T>)(IEnumerable(Of <T>), IEnumerable(Of <T>), IEqualityComparer(Of <T>))

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the set-theoretic intersection of two collections. The intersection of two sets is all items that appear in both of the sets. If an item appears X times in one set, and Y times in the other set, the intersection contains the item Minimum(X,Y) times. The source collections are not changed. A new collection is created with the intersection of the collections; the order of the items in this collection is undefined.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static IEnumerable<T> SetIntersection<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2,
    IEqualityComparer<T> equalityComparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function SetIntersection(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
) As IEnumerable(Of T)
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static IEnumerable<T>^ SetIntersection ( 
        IEnumerable<T>^ collection1,
        IEnumerable<T>^ collection2,
        IEqualityComparer<T>^ equalityComparer
    )
```

**Parameters**

**collection1**

`IEnumerable<(Of <T>>)

The first collection to intersect.

**collection2**

`IEnumerable<(Of <T>>)

The second collection to intersect.

**equalityComparer**
**IEqualityComparer<T>**

The IEqualityComparer<T> used to compare items for equality. Only the Equals and GetHashCode member functions of this interface are called.

**Return Value**

The intersection of the two collections, considered as sets.
Type Parameters

T
Remarks

When equal items appear in both collections, the returned collection will include an arbitrary choice of one of the two equal items.

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the Intersection or IntersectionWith methods on that class.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::SetSymmetricDifference Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SetSymmetricDifference</strong>&lt; (Of <code>T</code>)&gt;(IEnumerable&lt; (Of <code>T</code>)&gt;, IEnumerable&lt; (Of <code>T</code>)&gt;)</td>
<td>Computes the set-theoretic symmetric difference of two collections. The symmetric difference of two sets is all items that appear in the one of the sets, but not in the other. If an item appears X times in the one set, and Y times in the other set, the symmetric difference contains the item <code>AbsoluteValue(X - Y)</code> times. The source collections are not changed. A new collection is created with the symmetric difference of the collections; the order of the items in this collection is undefined.</td>
</tr>
<tr>
<td><strong>SetSymmetricDifference</strong>&lt; (Of <code>T</code>)&gt;(IEnumerable&lt; (Of <code>T</code>)&gt;, IEnumerable&lt; (Of <code>T</code>)&gt;, IEqualityComparer&lt; (Of <code>T</code>)&gt;)</td>
<td>Computes the set-theoretic symmetric difference of two collections. The symmetric difference of two sets is all items that appear in the one of the sets, but not in the other. If an item appears X times in the one set, and Y times in the other set, the symmetric difference contains the item <code>AbsoluteValue(X - Y)</code> times. The source collections are not changed. A new collection is created with the symmetric difference of the collections; the order of the items in this collection is undefined.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::SetSymmetricDifference(Of <T>)) Method (IEnumerable(Of <T>), IEnumerable(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the set-theoretic symmetric difference of two collections. The symmetric difference of two sets is all items that appear in the one of the sets, but not in the other. If an item appears X times in the one set, and Y times in the other set, the symmetric difference contains the item \(\text{AbsoluteValue}(X - Y)\) times. The source collections are not changed. A new collection is created with the symmetric difference of the collections; the order of the items in this collection is undefined.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEnumerable<T> SetSymmetricDifference<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function SetSymmetricDifference(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T) _
) As IEnumerable(Of T)
```

Visual C++

```cpp
public:
    generic<typename T>
    static IEnumerable<T>^ SetSymmetricDifference ( 
        IEnumerable<T>^ collection1,
        IEnumerable<T>^ collection2
    )
```

Parameters

**collection1**

```csharp
IEnumerable<Of <T>>
```

The first collection to symmetric difference.

**collection2**

```csharp
IEnumerable<Of <T>>
```

The second collection to symmetric difference.

Return Value

The symmetric difference of collection1 and collection2, considered as sets.
Type Parameters

T
Remarks

When equal items appear in both collections, the returned collection will include an arbitrary choice of one of the two equal items.

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the SymmetricDifference or SymmetricDifferenceWith methods on that class.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentNullException</td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#)
[WinInet2.PowerCollections](#) Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms...::SetSymmetricDifference(Of <T>) Method (IEnumerable(Of <T>), IEnumerable(Of <T>), IEqualityComparer(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the set-theoretic symmetric difference of two collections. The symmetric difference of two sets is all items that appear in the one of the sets, but not in the other. If an item appears X times in the one set, and Y times in the other set, the symmetric difference contains the item $\text{AbsoluteValue}(X - Y)$ times. The source collections are not changed. A new collection is created with the symmetric difference of the collections; the order of the items in this collection is undefined.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEnumerable<T> SetSymmetricDifference<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2,
    IEqualityComparer<T> equalityComparer
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function SetSymmetricDifference(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
) As IEnumerable(Of T)
```

Visual C++

```cpp
public:
    generic<typename T>
    static IEnumerable<T>^ SetSymmetricDifference ( 
        IEnumerable<T>^ collection1,
        IEnumerable<T>^ collection2,
        IEqualityComparer<T>^ equalityComparer
    )
```

Parameters

collection1

   IEnumerable<(Of <T>)>
   The first collection to symmetric difference.

collection2

   IEnumerable<(Of <T>)>
   The second collection to symmetric difference.

equalityComparer
The `IEqualityComparer<T>` used to compare items for equality. Only the `Equals` and `GetHashCode` member functions of this interface are called.

**Return Value**

The symmetric difference of collection1 and collection2, considered as sets.
Type Parameters

T
Remarks

When equal items appear in both collections, the returned collection will include an arbitrary choice of one of the two equal items.

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the symmetric difference or symmetric differenceWith methods on that class.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...::SetUnion Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong><code>SetUnion(Of &lt;T&gt;)(IEnumerable(Of &lt;T&gt;), IEnumerable(Of &lt;T&gt;), IEqualityComparer(Of &lt;T&gt;))</code></strong></td>
<td>Computes the set-theoretic union of two collections. The union of two sets is all items that appear in either of the sets. If an item appears $X$ times in one set, and $Y$ times in the other set, the union contains the item $\text{Maximum}(X, Y)$ times. The source collections are not changed. A new collection is created with the union of the collections; the order of the items in this collection is undefined.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms](https://docs.microsoft.com) Class
[Algorithms](https://docs.microsoft.com) Members
[Wintellect.PowerCollections](https://docs.microsoft.com) Namespace

Send [comments](https://docs.microsoft.com) about this topic to Microsoft.
Algorithms..:::SetUnion(Of <T>)(IEnumerable(Of <T>), IEnumerable(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the set-theoretic union of two collections. The union of two sets is all items that appear in either of the sets. If an item appears X times in one set, and Y times in the other set, the union contains the item Maximum(X,Y) times. The source collections are not changed. A new collection is created with the union of the collections; the order of the items in this collection is undefined.

Namespace: Wintellect.PowerCollections

Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public static IEnumerable<T> SetUnion<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2
)
```

### Visual Basic (Declaration)

```vbnet
Public Shared Function SetUnion(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T) _
) As IEnumerable(Of T)
```

### Visual C++

```cpp
class std::set<std::unordered_set<T, Hash, KeyEqual>>

public:
    template<typename T>
    static std::unordered_set<T> SetUnion ( _
        std::unordered_set<T> collection1,
        std::unordered_set<T> collection2
    )

```

## Parameters

**collection1**

- `IEnumerable<T>`
- The first collection to union.

**collection2**

- `IEnumerable<T>`
- The second collection to union.

## Return Value

The union of the two collections, considered as sets.
<table>
<thead>
<tr>
<th>Type Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
</tr>
</tbody>
</table>
Remarks

When equal items appear in both collections, the returned collection will include an arbitrary choice of one of the two equal items.

The default sense of equality for T is used, as defined by T's implementation of IComparable<T>.Equals or object.Equals.

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the Union or UnionWith methods on that class.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td><code>collection1</code> or <code>collection2</code> is null.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Computes the set-theoretic union of two collections. The union of two sets is all items that appear in either of the sets. If an item appears \(X\) times in one set, and \(Y\) times in the other set, the union contains the item \(\text{Maximum}(X,Y)\) times. The source collections are not changed. A new collection is created with the union of the collections; the order of the items in this collection is undefined.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEnumerable<T> SetUnion<T>(
    IEnumerable<T> collection1,
    IEnumerable<T> collection2,
    IEqualityComparer<T> equalityComparer
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function SetUnion(Of T) ( _
    collection1 As IEnumerable(Of T), _
    collection2 As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _) As IEnumerable(Of T)
)
```

Visual C++

```cpp
public:
    template<typename T>
    static IEnumerable<T>^ SetUnion ( 
        IEnumerable<T>^ collection1, 
        IEnumerable<T>^ collection2, 
        IEqualityComparer<T>^ equalityComparer 
    )
```

Parameters

collection1

```csharp
IEnumerable<(Of <T>)>
```

The first collection to union.

collection2

```csharp
IEnumerable<(Of <T>)>
```

The second collection to union.

equalityComparer
**IEqualityComparer<T>**

The IEqualityComparer<T> used to compare items for equality. Only the Equals and GetHashCode member functions of this interface are called.

**Return Value**

The union of the two collections, considered as sets.
Type Parameters

T
Remarks

When equal items appear in both collections, the returned collection will include an arbitrary choice of one of the two equal items.

If both collections are Set, Bag, OrderedSet, or OrderedBag collections, it is more efficient to use the union or unionWith methods on that class.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>collection1 or collection2 is null.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms] Class
[Wintellect.PowerCollections] Namespace

Send comments about this topic to Microsoft.
[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Sort(Of&lt;T&gt;)(IEnumerable(Of&lt;T&gt;))</code></td>
<td>Creates a sorted version of a collection.</td>
</tr>
<tr>
<td><code>Sort(Of&lt;T&gt;())</code></td>
<td>Creates a sorted version of a collection. A supplied <code>IComparer&lt;T&gt;</code> is used to compare the items in the collection.</td>
</tr>
<tr>
<td><code>Sort(Of&lt;T&gt;(), IComparer(Of&lt;T&gt;())</code></td>
<td>Creates a sorted version of a collection. A supplied <code>Comparison&lt;T&gt;</code> delegate is used to compare the items in the collection.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...::Sort(Of T) Method (IEnumerable(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a sorted version of a collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
-Syntax

C#

```
public static T[] Sort<T>(
    IEnumerable<T> collection
) where T : IComparable<T>
```

Visual Basic (Declaration)

```
Public Shared Function Sort(Of T As IComparable(Of T)) ( _
    collection As IEnumerable(Of T) _
) As T()
```

Visual C++

```
publish:
generic<typename T>
where T : IComparable<T>
static array<T>^ Sort (_
    IEnumerable<T>^ collection
)
```

Parameters

collection
    IEnumerable<(Of <T>)> The collection to sort.

Return Value

An array containing the sorted version of the collection.
Type Parameters

T
Remarks

Values are compared by using the IComparable<T> interfaces implementation on the type T.
See Also

**Algorithms** Class

**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms...::Sort(Of T) Method (IEnumerable(Of T), IComparer(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a sorted version of a collection. A supplied IComparer<T> is used to compare the items in the collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static T[] Sort<T>(
    IEnumerable<T> collection,
    IComparer<T> comparer
)

Visual Basic (Declaration)

Public Shared Function Sort(Of T) ( _
    collection As IEnumerable(Of T), _
    comparer As IComparer(Of T) _
) As T()

Visual C++

public:
    generic<
        typename T>
    static array<T>^ Sort ( 
        IEnumerable<T>^ collection,
        IComparer<T>^ comparer
    )

Parameters

collection
    IEnumerable(Of <T>)
    The collection to sort.

comparer
    IComparer(Of <T>)
    The comparer instance used to compare items in the collection. Only the
    Compare method is used.

Return Value
An array containing the sorted version of the collection.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::Sort<Of <T>> Method (IEnumerable<Of <T>>, Comparison<Of <T>>)

See Also

☐ Visual Basic (Declaration) ☐ Visual Basic (Usage)
☐ C#
☐ Visual C++
☐ J#
☐ JScript
☐ XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a sorted version of a collection. A supplied Comparison<T> delegate is used to compare the items in the collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static T[] Sort<T>(
    IEnumerable<T> collection,
    Comparison<T> comparison
)

Visual Basic (Declaration)

Public Shared Function Sort(Of T) ( _
    collection As IEnumerable(Of T), _
    comparison As Comparison(Of T) _
) As T()  

Visual C++

public:
generic<typename T>
static array<T>^ Sort (  
    IEnumerable<T>^ collection,
    Comparison<T>^ comparison
)

Parameters

collection
    IEnumerable<(Of <T>)>  
The collection to sort.

comparison
    Comparison<(Of <T>)>  
The comparison delegate used to compare items in the collection.

Return Value

An array containing the sorted version of the collection.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...::SortInPlace Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SortInPlace(Of T)&gt; (IList(Of T)&gt;)</strong></td>
<td>Sorts a list or array in place.</td>
</tr>
<tr>
<td><strong>SortInPlace(Of T)&gt; (IList(Of T&gt;, IComparer(Of T)&gt;)</strong></td>
<td>Sorts a list or array in place. A supplied IComparer&lt;T&gt; is used to compare the items in the list.</td>
</tr>
<tr>
<td><strong>SortInPlace(Of T)&gt; (IList(Of T&gt;, Comparison(Of T)&gt;)</strong></td>
<td>Sorts a list or array in place. A supplied Comparison&lt;T&gt; delegate is used to compare the items in the list.</td>
</tr>
</tbody>
</table>
See Also

- [Algorithms Class](#)
- [Algorithms Members](#)
- [Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Algorithms::SortInPlace(Of T) Method (IList(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Sorts a list or array in place.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static void SortInPlace<T>(
    IList<T> list
) where T : IComparable<T>

Visual Basic (Declaration)

Public Shared Sub SortInPlace(Of T As IComparable(Of T)) ( _
    list As IList(Of T) _
)

Visual C++

public:
    generic<T>
    where T : IComparable<T>
    static void SortInPlace ( 
        IList<T>^ list
    )

Parameters

list  
    IList<(Of <T>)>
    The list or array to sort.
Type Parameters

T
Remarks

The Quicksort algorithm is used to sort the items. In virtually all cases, this takes time $O(N \log N)$, where $N$ is the number of items in the list.

Values are compared by using the `IComparable<T>` interfaces implementation on the type $T$.

Although arrays cast to `IList<T>` are normally read-only, this method will work correctly and modify an array passed as list.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Sorts a list or array in place. A supplied IComparer<T> is used to compare the items in the list.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static void SortInPlace<T>(
    IList<T> list,
    IComparer<T> comparer
)

Visual Basic (Declaration)

Public Shared Sub SortInPlace(Of T) ( _
    list As IList(Of T), _
    comparer As IComparer(Of T) _
)

Visual C++

public:
    generic<typename T>
    static void SortInPlace ( 
        IList<T>^ list,
        IComparer<T>^ comparer
    )

Parameters

list
    IList<(Of <T>)>
    The list or array to sort.

comparer
    IComparer<(Of <T>)>
    The comparer instance used to compare items in the collection. Only the Compare method is used.
Type Parameters

T
Remarks

The Quicksort algorithms is used to sort the items. In virtually all cases, this takes time $O(N \log N)$, where $N$ is the number of items in the list.

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...::SortInPlace<(Of <T>)> Method (IList<(Of <T>)>, Comparison<(Of <T>)>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Sorts a list or array in place. A supplied Comparison<T> delegate is used to compare the items in the list.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static void SortInPlace<T>(
    IList<T> list,
    Comparison<T> comparison
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Sub SortInPlace(Of T) ( _
    list As IList(Of T), _
    comparison As Comparison(Of T) _
)
```

Visual C++

```cpp
public:
    template<typename T>
    static void SortInPlace ( 
        IList<T>^ list,
        Comparison<T>^ comparison
    )
```

Parameters

list

```csharp
IList<Of <T> )
```

The list or array to sort.

comparison

```csharp
Comparison<Of <T> )
```

The comparison delegate used to compare items in the collection.
Type Parameters

T
Remarks

The Quicksort algorithm is used to sort the items. In virtually all cases, this takes time $O(N \log N)$, where $N$ is the number of items in the list.

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::StablePartition<Of <T>> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Partition a list or array based on a predicate. After partitioning, all items for which the predicate returned true precede all items for which the predicate returned false. The partition is stable, which means that if items X and Y have the same result from the predicate, and X precedes Y in the original list, X will precede Y in the partitioned list.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

generic
class StablesPartition<T>(
    T list,
    Predicate<T> predicate
)

Visual Basic (Declaration)

Public Shared Function StablesPartition(Of T) ( _
    list As IList(Of T), _
    predicate As Predicate(Of T) _
) As Integer

Visual C++

public:
    generic<typename T>
    static int StablesPartition ( _
        IList<T>^ list,
        Predicate<T>^ predicate
)

Parameters

list
    IList<Of <T>>
    The list or array to partition.

predicate
    Predicate<Of <T>>
    A delegate that defines the partitioning condition.

Return Value

The index of the first item in the second half of the partition; i.e., the first item
for which predicate returned false. If the predicate was true for all items in the list, list.Count is returned.
Type Parameters

T
Remarks

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

[Algorithms Class](Wintellect.PowerCollections) Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms::StableSort Method

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StableSort(Of &lt;T&gt;)(IEnumerable(Of &lt;T&gt;))</code></td>
<td>Creates a sorted version of a collection. The sort is stable, which means that if items X and Y are equal, and X precedes Y in the unsorted collection, X will precede Y in the sorted collection.</td>
</tr>
<tr>
<td><code>StableSort(Of &lt;T&gt;)(IEnumerable(Of &lt;T&gt;), IComparer(Of &lt;T&gt;))</code></td>
<td>Creates a sorted version of a collection. The sort is stable, which means that if items X and Y are equal, and X precedes Y in the unsorted collection, X will precede Y in the sorted collection. A supplied IComparer&lt;T&gt; is used to compare the items in the collection.</td>
</tr>
<tr>
<td><code>StableSort(Of &lt;T&gt;)(IEnumerable(Of &lt;T&gt;), Comparison(Of &lt;T&gt;))</code></td>
<td>Creates a sorted version of a collection. The sort is stable, which means that if items X and Y are equal, and X precedes Y in the unsorted collection, X will precede Y in the sorted collection. A supplied Comparison&lt;T&gt; delegate is used to compare the items in the collection.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::StableSort(Of T) Method (IEnumerable(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a sorted version of a collection. The sort is stable, which means that if items X and Y are equal, and X precedes Y in the unsorted collection, X will precede Y in the sorted collection.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static T[] StableSort<T>(
    IEnumerable<T> collection
) where T : IComparable<T>

Visual Basic (Declaration)

Public Shared Function StableSort(Of T As IComparable(Of T)) ( _
    collection As IEnumerable(Of T) _
) As T()

Visual C++

public:
generic<
typename T>
where T : IComparable<T>
static array<T>^ StableSort ( _
    IEnumerable<T>^ collection
)

Parameters

collection
    IEnumerable<(Of <T>)>
    The collection to sort.

Return Value

An array containing the sorted version of the collection.
Type Parameters

T
Remarks

Values are compared by using the IComparable<T> interfaces implementation on the type T.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithm::StableSort(Of <T>) Method (IEnumerable(Of <T>), IComparer(Of <T>))

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a sorted version of a collection. The sort is stable, which means that if items X and Y are equal, and X precedes Y in the unsorted collection, X will precede Y in the sorted collection. A supplied IComparer<T> is used to compare the items in the collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

C#

```csharp
public static T[] StableSort<T>(
    IEnumerable<T> collection,
    IComparer<T> comparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function StableSort(Of T) ( _
    collection As IEnumerable(Of T), _
    comparer As IComparer(Of T) _
) As T()
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static array<T>^ StableSort ( 
        IEnumerable<T>^ collection, 
        IComparer<T>^ comparer
    )
```

**Parameters**

collection
    `IEnumerable<Of <T>>`
    The collection to sort.

comparer
    `IComparer<Of <T>>`
    The comparer instance used to compare items in the collection. Only the Compare method is used.

**Return Value**
An array containing the sorted version of the collection.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::StableSort<(Of <T>)> Method (IEnumerable<(Of <T>)>, Comparison<(Of <T>)>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a sorted version of a collection. The sort is stable, which means that if items X and Y are equal, and X precedes Y in the unsorted collection, X will precede Y in the sorted collection. A supplied Comparison<`T>` delegate is used to compare the items in the collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static T[] StableSort<T>(
    IEnumerable<T> collection,
    Comparison<T> comparison
)

Visual Basic (Declaration)

Public Shared Function StableSort(Of T) ( _
    collection As IEnumerable(Of T), _
    comparison As Comparison(Of T) _
) As T()

Visual C++

public:
    generic<typename T>
    static array<T>^ StableSort (    
        IEnumerable<T>^ collection,
        Comparison<T>^ comparison
    )

Parameters

collection
    IEnumerable<(Of <T>)>
    The collection to sort.

comparison
    Comparison<(Of <T>)>
    The comparison delegate used to compare items in the collection.

Return Value

An array containing the sorted version of the collection.
Type Parameters

T
Remarks

Values are compared by using the IComparable<T> interfaces implementation on the type T.
See Also

- [Algorithms Class](#)
- [Wintellect.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StableSortInPlace&lt;Of &lt;T&gt;, (IList&lt;Of &lt;T&gt;&gt;)&gt;</td>
<td>Sorts a list or array in place. The sort is stable, which means that if items X and Y are equal, and X precedes Y in the unsorted collection, X will precede Y in the sorted collection.</td>
</tr>
<tr>
<td>StableSortInPlace&lt;Of &lt;T&gt;, (IList&lt;Of &lt;T&gt;&gt;, IComparer&lt;Of &lt;T&gt;&gt;)&gt;</td>
<td>Sorts a list or array in place. The sort is stable, which means that if items X and Y are equal, and X precedes Y in the unsorted collection, X will precede Y in the sorted collection. A supplied IComparer&lt;T&gt; is used to compare the items in the list.</td>
</tr>
<tr>
<td>StableSortInPlace&lt;Of &lt;T&gt;, (IList&lt;Of &lt;T&gt;&gt;, Comparison&lt;Of &lt;T&gt;&gt;)&gt;</td>
<td>Sorts a list or array in place. The sort is stable, which means that if items X and Y are equal, and X precedes Y in the unsorted collection, X will precede Y in the sorted collection. A supplied Comparison&lt;T&gt; delegate is used to compare the items in the list.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::StableSortInPlace(Of T) Method (IList(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Sorts a list or array in place. The sort is stable, which means that if items X and Y are equal, and X precedes Y in the unsorted collection, X will precede Y in the sorted collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static void StableSortInPlace<T>(
    IList<T> list
) where T : IComparable<T>
```

Visual Basic (Declaration)

```vbnet
Public Shared Sub StableSortInPlace(Of T As IComparable(Of T)) ( _
    list As IList(Of T) _
)
```

Visual C++

```cpp
public:
    generic<typename T>
    where T : IComparable<T>
    static void StableSortInPlace (    
        IList<T>* list
    )
```

Parameters

list
    IList<(Of <T>)>
    The list or array to sort.
Type Parameters

T
Remarks

Values are compared by using the IComparable<T> interfaces implementation on the type T.

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
StableSortInPlace<
(OF <T>)>
Method
(IList<
(OF <T>)>,
IComparer<
(OF <T>)>)

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Sorts a list or array in place. The sort is stable, which means that if items X and Y are equal, and X precedes Y in the unsorted collection, X will precede Y in the sorted collection. A supplied IComparer<
<T> is used to compare the items in the list.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static void StableSortInPlace<T>(
    IList<T> list,
    IComparer<T> comparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Sub StableSortInPlace(Of T) ( _
    list As IList(Of T), _
    comparer As IComparer(Of T) _) 
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static void StableSortInPlace ( 
        IList<T>^ list,
        IComparer<T>^ comparer
    )
```

**Parameters**

**list**

`IList<(Of <T>>)`
The list or array to sort.

**comparer**

`IComparer<(Of <T>>)`
The comparer instance used to compare items in the collection. Only the Compare method is used.
Type Parameters

T
Remarks

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::StableSortInPlace(Of T) Method (IList(Of T), Comparison(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Sorts a list or array in place. The sort is stable, which means that if items X and Y are equal, and X precedes Y in the unsorted collection, X will precede Y in the sorted collection. A supplied Comparison<T> delegate is used to compare the items in the list.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

    public static void StableSortInPlace<T>(
        IList<T> list,
        Comparison<T> comparison
    )

Visual Basic (Declaration)

    Public Shared Sub StableSortInPlace(Of T) ( _
        list As IList(Of T), _
        comparison As Comparison(Of T) _
    )

Visual C++

    public:
    generic<typename T>
    static void StableSortInPlace ( 
        IList<T>^ list,
        Comparison<T>^ comparison
    )

Parameters

list

    IList<(Of <T>)>
    The list or array to sort.

comparison

    Comparison<(Of <T>)>
    The comparison delegate used to compare items in the collection.
<table>
<thead>
<tr>
<th>Type Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
</tr>
</tbody>
</table>
Remarks

Although arrays cast to IList<T> are normally read-only, this method will work correctly and modify an array passed as list.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Create an array with the items in a collection.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static T[] ToArray<T>(
    IEnumerable<T> collection
)

Visual Basic (Declaration)

Public Shared Function ToArray(Of T) ( _
    collection As IEnumerable(Of T) _
) As T()

Visual C++

public:
    generic<typename T>
    static array<T>^ ToArray ( _
        IEnumerable<T>^ collection
    )

Parameters

collection
    IEnumerable<(Of <T>)>
    Collection to create array from.

Return Value

An array with the items from the collection, in enumeration order.
Type Parameters

T

Element type of the collection.
Remarks

If collection implements ICollection<T>, then ICollection<T>.CopyTo() is used to fill the array. Otherwise, the IEnumerable<T>.GetEnumerator() is used to fill the array.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

[Algorithm]s Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::ToString Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ToString&lt;(Of &lt;TKey, TValue&gt;)(IDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;)</code></td>
<td>Gets a string representation of the mappings in a dictionary. The string representation starts with &quot;{&quot; , has a list of mappings separated by commas (&quot;, &quot;), and ends with &quot;}&quot;. Each mapping is represented by &quot;key-&gt;value&quot;. Each key and value in the dictionary is converted to a string by calling its <code>ToString</code> method (null is represented by &quot;null&quot;). Contained collections (except strings) are recursively converted to strings by this method.</td>
</tr>
<tr>
<td><code>ToString&lt;(Of &lt;T&gt;)(IEnumerable&lt;(Of &lt;T&gt;)&gt;)</code></td>
<td>Gets a string representation of the elements in the collection. The string representation starts with &quot;{&quot; , has a list of items separated by commas (&quot;,&quot;), and ends with &quot;}&quot;. Each item in the collection is converted to a string by calling its <code>ToString</code> method (null is represented by &quot;null&quot;). Contained collections (except strings) are recursively converted to strings by this method.</td>
</tr>
<tr>
<td><code>ToString&lt;(Of &lt;T&gt;)(IEnumerable&lt;(Of &lt;T&gt;), Boolean, String, String, String)&gt;)</code></td>
<td>Gets a string representation of the elements in the collection. The string to used at the beginning and end, and to separate items, and supplied by parameters. Each item in the collection is converted to a string by calling its <code>ToString</code> method (null is represented by &quot;null&quot;).</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...::ToString(Of <TKey, TValue>)(Of IDictionary(Of <TKey, TValue>)) Method (IDictionary(Of <TKey, TValue>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets a string representation of the mappings in a dictionary. The string representation starts with "{", has a list of mappings separated by commas (", "), and ends with "}". Each mapping is represented by "key->value". Each key and value in the dictionary is converted to a string by calling its ToString method (null is represented by "null"). Contained collections (except strings) are recursively converted to strings by this method.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
- **Syntax**

**C#**

```csharp
public static string ToString<TKey, TValue>(
    IDictionary<TKey, TValue> dictionary
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function ToString(Of TKey, TValue) ( _
    dictionary As IDictionary(Of TKey, TValue) _) _
) As String
```

**Visual C++**

```cpp
public:
    generic<typename TKey, typename TValue>
    static String^ ToString ( 
        IDictionary<TKey, TValue>^ dictionary
    )
```

**Parameters**

dictionary

    IDictionary<TKey, TValue>

A dictionary to get the string representation of.

**Return Value**

The string representation of the collection, or "null" if dictionary is null.
Type Parameters

TKey
TValue
See Also

[Algorithms](#) Class
[Wintellect.PowerCollections](#) Namespace

Send [comments](#) about this topic to Microsoft.
Algorithms::ToString(Of T) Method (IEnumerable(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

 Gets a string representation of the elements in the collection. The string representation starts with "{", has a list of items separated by commas (",") and ends with "}". Each item in the collection is converted to a string by calling its ToString method (null is represented by "null"). Contained collections (except strings) are recursively converted to strings by this method.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public static string ToString<T>(
    IEnumerable<T> collection
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function ToString(Of T) ( _
    collection As IEnumerable(Of T) _
) As String
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static String^ ToString ( 
        IEnumerable<T>^ collection
    )
```

**Parameters**

- **collection**
  - `IEnumerable<(Of <T>)>`
  - A collection to get the string representation of.

**Return Value**

The string representation of the collection. If collection is null, then the string "null" is returned.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...:::ToString<(Of <T>)> Method (IEnumerable<(Of <T>)), Boolean, String, String, String)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets a string representation of the elements in the collection. The string to use at the beginning and end, and to separate items, and supplied by parameters. Each item in the collection is converted to a string by calling its ToString method (null is represented by "null").

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public static string ToString<T>(
    IEnumerable<T> collection,
    bool recursive,
    string start,
    string separator,
    string end
)

Visual Basic (Declaration)

Public Shared Function ToString(Of T) ( _
    collection As IEnumerable(Of T), _
    recursive As Boolean, _
    start As String, _
    separator As String, _
    end As String _
) As String

Visual C++

public:

generic<typename T>
static String^ ToString ( 
    IEnumerable<T>^ collection,
    bool recursive,
    String^ start,
    String^ separator,
    String^ end
)

Parameters

collection
    IEnumerable<(Of <T>)>
    A collection to get the string representation of.

recursive
**Boolean**
If true, contained collections (except strings) are converted to strings by a recursive call to this method, instead of by calling ToString.

**start**

**String**
The string to appear at the beginning of the output string.

**separator**

**String**
The string to appear between each item in the string.

**end**

**String**
The string to appear at the end of the output string.

**Return Value**

The string representation of the collection. If collection is null, then the string "null" is returned.
Type Parameters

T
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>start, separator, or end is null.</td>
</tr>
</tbody>
</table>
See Also

[Algorithms Class](#)
[WinTellecT.PowerCollections Namespace](#)

Send [comments](#) about this topic to Microsoft.
Algorithms...:::TrueForAll<(Of <T>> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if all of the items in the collection satisfy the condition defined by predicate.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static bool TrueForAll<T>(
    IEnumerable<T> collection,
    Predicate<T> predicate
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function TrueForAll(Of T) ( _
    collection As IEnumerable(Of T), _
    predicate As Predicate(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
    generic<typename T>
    static bool TrueForAll ( 
        IEnumerable<T>^ collection,
        Predicate<T>^ predicate
    )
```

Parameters

collection
    IEnumerable<(Of <T>)>  
    The collection to check all the items in.
	predicate
    Predicate<(Of <T>)>  
    A delegate that defines the condition to check for.

Return Value

True if all of the items in the collection satisfy the condition defined by
predicate, or if the collection is empty. False if one or more items in the collection do not satisfy predicate.
Type Parameters

T
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms::<TryFindFirstWhere(Of <T>)> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the first item in a collection that satisfies the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public static bool TryFindFirstWhere<T>(
    IEnumerable<T> collection,
    Predicate<T> predicate,
    out T foundItem
)
```

### Visual Basic (Declaration)

```vbnet
Public Shared Function TryFindFirstWhere(Of T) ( _
    collection As IEnumerable(Of T), _
    predicate As Predicate(Of T), _
    <OutAttribute> ByRef foundItem As T _
) As Boolean
```

### Visual C++

```cpp
public:
    generic<typename T>
    static bool TryFindFirstWhere ( 
        IEnumerable<T>^ collection,
        Predicate<T>^ predicate,
        [OutAttribute] T% foundItem
    )
```

### Parameters

- **collection**
  - `IEnumerable<Of <T>>`
  - The collection to search.

- **predicate**
  - `Predicate<Of <T>>`
  - A delegate that defined the condition to check for.

- **foundItem**
T%
Outputs the first item in the collection that matches the condition, if the method returns true.

Return Value

True if an item satisfying the condition was found. False if no such item exists in the collection.
Type Parameters

T
See Also

**Algorithms** Class
**Wintellect.PowerCollections** Namespace
**Algorithms.**:::**FindFirstWhere**::<Of <T>>>(IEnumerable::<Of <T>>),
**Predicate**::<Of <T>>)

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
Algorithms:::TryFindLastWhere<(Of <T>)> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the last item in a collection that satisfies the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

**C#**

```csharp
public static bool TryFindLastWhere<T>(
    IEnumerable<T> collection,
    Predicate<T> predicate,
    out T foundItem
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Function TryFindLastWhere(Of T) ( _
    collection As IEnumerable(Of T), _
    predicate As Predicate(Of T), _
    <outAttribute> ByRef foundItem As T _
) As Boolean
```

**Visual C++**

```cpp
public:
    template<typename T>
    static bool TryFindLastWhere (
        IEnumerable<T>^ collection, 
        Predicate<T>^ predicate, 
        [outAttribute] T% foundItem
    )
```

### Parameters

**collection**

- `IEnumerable<Of T>`
  - The collection to search.

**predicate**

- `Predicate<Of T>`
  - A delegate that defined the condition to check for.

**foundItem**
T%
Outputs the last item in the collection that matches the condition, if the method returns true.

**Return Value**

True if an item satisfying the condition was found. False if no such item exists in the collection.
Type Parameters

T
Remarks

If the collection implements IList<T>, then the list is scanned in reverse until a matching item is found. Otherwise, the entire collection is iterated in the forward direction.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace
Algorithms...::FindLastWhere<(Of <T>>)(IEnumerable<(Of <T>>), Predicate<(Of <T>>))

Send comments about this topic to Microsoft.
Algorithms::TypedAs Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TypedAs&lt;Of &lt;T&gt;&gt;</strong>&lt;br&gt;(ICollection)**</td>
<td>Given a non-generic ICollection interface, wrap a generic ICollection&lt;T&gt; interface around it. The generic interface will enumerate the same objects as the underlying non-generic collection, but can be used in places that require a generic interface. The underlying non-generic collection must contain only items that are of type T or a type derived from it. This method is useful when interfacing older, non-generic collections to newer code that uses generic interfaces.</td>
</tr>
<tr>
<td><strong>TypedAs&lt;Of &lt;T&gt;&gt;</strong>&lt;br&gt;(IEnumerable)**</td>
<td>Given a non-generic IEnumerable interface, wrap a generic IEnumerable&lt;T&gt; interface around it. The generic interface will enumerate the same objects as the underlying non-generic collection, but can be used in places that require a generic interface. The underlying non-generic collection must contain only items that are of type T or a type derived from it. This method is useful when interfacing older, non-generic collections to newer code that uses generic interfaces.</td>
</tr>
<tr>
<td><strong>TypedAs&lt;Of &lt;T&gt;&gt;</strong>&lt;br&gt;(IList)**</td>
<td>Given a non-generic IList interface, wrap a generic IList&lt;T&gt; interface around it. The generic interface will enumerate the same objects as the underlying non-generic list, but can be used in places that require a generic interface. The underlying non-generic list must contain only items that are of type T or a type derived from it. This method is useful when interfacing older, non-generic lists to newer code that uses generic interfaces.</td>
</tr>
</tbody>
</table>
See Also

Algorithms Class
Algorithms Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
 Algorithms...:::TypedAs<(Of <T>)> Method (ICollection)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Given a non-generic ICollection interface, wrap a generic ICollection<T> interface around it. The generic interface will enumerate the same objects as the underlying non-generic collection, but can be used in places that require a generic interface. The underlying non-generic collection must contain only items that are of type T or a type derived from it. This method is useful when interfacing older, non-generic collections to newer code that uses generic interfaces.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static ICollection<T> TypedAs<T>(
    ICollection untypedCollection
)

Visual Basic (Declaration)

Public Shared Function TypedAs(Of T) ( _
    untypedCollection As ICollection _
) As ICollection(Of T)

Visual C++

public:
    generic<typename T>
    static ICollection<T>^ TypedAs (  
        ICollection^ untypedCollection
    )

Parameters

untypedCollection
    ICollection
    An untyped collection. This collection should only contain items of type T
    or a type derived from it.

Return Value

A generic ICollection<T> wrapper around untypedCollection. If
untypedCollection is null, then null is returned.
**Type Parameters**

*T*

The item type of the wrapper collection.
Remarks

Some collections implement both generic and non-generic interfaces. For efficiency, this method will first attempt to cast untypedCollection to ICollection<T>. If that succeeds, it is returned; otherwise, a wrapper object is created.

Unlike the generic interface, the non-generic ICollection interfaces does not contain methods for adding or removing items from the collection. For this reason, the returned ICollection<T> will be read-only.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithm::TypedAs(Of <T>)> Method (IEnumerable)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Given a non-generic IEnumerable interface, wrap a generic IEnumerable&lt;T&gt; interface around it. The generic interface will enumerate the same objects as the underlying non-generic collection, but can be used in places that require a generic interface. The underlying non-generic collection must contain only items that are of type T or a type derived from it. This method is useful when interfacing older, non-generic collections to newer code that uses generic interfaces.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static IEnumerable<T> TypedAs<T>(
    IEnumerable untypedCollection
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Function TypedAs(Of T) ( _
    untypedCollection As IEnumerable _
) As IEnumerable(Of T)
```

Visual C++

```cpp
public:
    generic<typename T>
    static IEnumerable<T>^ TypedAs (^
        IEnumerable^ untypedCollection
    )
```

Parameters

`untypedCollection`  
`IEnumerable`  
An untyped collection. This collection should only contain items of type `T` or a type derived from it.

Return Value

A generic `IEnumerable<T>` wrapper around `untypedCollection`. If `untypedCollection` is null, then null is returned.
Type Parameters

T
The item type of the wrapper collection.
Remarks

Some collections implement both generic and non-generic interfaces. For efficiency, this method will first attempt to cast untypedCollection to IEnumerable<T>. If that succeeds, it is returned; otherwise, a wrapper object is created.
See Also

 Algorithms Class
 Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Given a non-generic IList interface, wrap a generic IList<T> interface around it. The generic interface will enumerate the same objects as the underlying non-generic list, but can be used in places that require a generic interface. The underlying non-generic list must contain only items that are of type T or a type derived from it. This method is useful when interfacing older, non-generic lists to newer code that uses generic interfaces.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static IList<T> TypedAs<T>(
    IList untypedList
)

Visual Basic (Declaration)

Public Shared Function TypedAs(Of T) ( _
    untypedList As IList _
) As IList(Of T)

Visual C++

public:
    generic<typename T>
    static IList<T>^ TypedAs ( _
        IList^ untypedList
    )

Parameters

untypedList

    IList
    An untyped list. This list should only contain items of type T or a type derived from it.

Return Value

A generic IList<T> wrapper around untypedList. If untypedList is null, then null is returned.
Type Parameters

T
The item type of the wrapper list.
Remarks

Some collections implement both generic and non-generic interfaces. For efficiency, this method will first attempt to cast untypedList to IList<T>. If that succeeds, it is returned; otherwise, a wrapper object is created.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Algorithms...::Untyped Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Untyped&lt;(Of &lt;T&gt;)&gt;</td>
<td>Given a generic ICollection&lt;T&gt; interface, wrap a non-generic (untyped) ICollection interface around it. The non-generic interface will contain the same objects as the underlying generic collection, but can be used in places that require a non-generic interface. This method is useful when interfacing generic interfaces with older code that uses non-generic interfaces.</td>
</tr>
<tr>
<td>InvalidOperationException&lt;(ICollection&lt;(Of &lt;T&gt;))&gt;&gt;</td>
<td></td>
</tr>
<tr>
<td>Untyped&lt;(Of &lt;T&gt;)&gt;</td>
<td>Given a generic IList&lt;T&gt; interface, wrap a non-generic (untyped) IList interface around it. The non-generic interface will contain the same objects as the underlying generic list, but can be used in places that require a non-generic interface. This method is useful when interfacing generic interfaces with older code that uses non-generic interfaces.</td>
</tr>
</tbody>
</table>
See Also

* Algorithms Class
* Algorithms Members
* Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Given a generic ICollection<T> interface, wrap a non-generic (untyped) ICollection interface around it. The non-generic interface will contain the same objects as the underlying generic collection, but can be used in places that require a non-generic interface. This method is useful when interfacing generic interfaces with older code that uses non-generic interfaces.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public static ICollection Untyped<T>(
    ICollection<T> typedCollection
)
```

### Visual Basic (Declaration)

```vbnet
Public Shared Function Untyped(Of T) ( _
    typedCollection As ICollection(Of T) _
) As ICollection
```

### Visual C++

```cpp
public:
    generic<typeName T>
    static ICollection^ Untyped ( 
        ICollection^ typedCollection 
    )
```

## Parameters

- **typedCollection**
  - `ICollection<Of <T>>`
  - A typed collection to wrap.

## Return Value

A non-generic ICollection wrapper around typedCollection. If typedCollection is null, then null is returned.
Type Parameters

T

The item type of the underlying collection.
Remarks

Many generic collections already implement the non-generic interfaces directly. This method will first attempt to simply cast typedCollection to ICollection. If that succeeds, it is returned; if it fails, then a wrapper object is created.
See Also

 Algorithms Class
 Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Given a generic `IList<T>` interface, wrap a non-generic (untyped) `IList` interface around it. The non-generic interface will contain the same objects as the underlying generic list, but can be used in places that require a non-generic interface. This method is useful when interfacing generic interfaces with older code that uses non-generic interfaces.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public static IList Untyped<T>(
    IList<T> typedList
)
```

### Visual Basic (Declaration)

```vbnet
Public Shared Function Untyped(Of T) ( _
    typedList As IList(Of T) _
) As IList
```

### Visual C++

```cpp
public:
    generic<typename T>
    static IList^ Untyped ( _
        IList<T>^ typedList
    )
```

## Parameters

typedList

- `IList<Of <T>>`
  - A typed list to wrap.

## Return Value

A non-generic IList wrapper around typedList. If typedList is null, then null is returned.
Type Parameters

T
The item type of the underlying list.
Remarks

Many generic collections already implement the non-generic interfaces directly. This method will first attempt to simply cast typedList to IList. If that succeeds, it is returned; if it fails, then a wrapper object is created.
See Also

Algorithms Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>> Class

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Bag<T> is a collection that contains items of type T. Unlike a Set, duplicate items (items that compare equal to each other) are allowed in an Bag.

Namespace: Wintellect.PowerCollections
Assembly:  PowerCollections (in PowerCollections.dll)
Syntax

C#

[SerializableAttribute]
public class Bag<T> : CollectionBase<T>, ICloneable

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class Bag(Of T) _
    Inherits CollectionBase(Of T) _
    Implements ICloneable

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class Bag : public CollectionBase<T>,
    ICloneable
Type Parameters

T
Remarks

The items are compared in one of two ways. If T implements IComparable<T> then the Equals method of that interface will be used to compare items, otherwise the Equals method from Object will be used. Alternatively, an instance of IComparer<T> can be passed to the constructor to use to compare items.

Bag is implemented as a hash table. Inserting, deleting, and looking up an element all are done in approximately constant time, regardless of the number of items in the bag.

When multiple equal items are stored in the bag, they are stored as a representative item and a count. If equal items can be distinguished, this may be noticable. For example, if a case-insensitive comparer is used with a Bag<string>, and both "hello", and "HELLO" are added to the bag, then the bag will appear to contain two copies of "hello" (the representative item).

OrderedBag<Of <T>> is similar, but uses comparison instead of hashing, maintain the items in sorted order, and stores distinct copies of items that compare equal.
Inheritance Hierarchy

System::Object
Wintellect.PowerCollections::CollectionBase<T>
Wintellect.PowerCollections::Bag<T>
See Also

Bag(Of T)> Members
Wintellect.PowerCollections Namespace
Wintellect.PowerCollections...:::OrderedBag(Of T>)

Send comments about this topic to Microsoft.
Bag(Of <T>) Members

See Also  Methods  Constructors  Properties  Explicit Interface Implementations

☐ Include Inherited Members ☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#  

```csharp
[SerializableAttribute]
public class Bag<T>
```

**Visual Basic (Declaration)**

```vbnet
<SeriableizableAttribute> _
Public Class Bag(Of T)
```

**Visual C++**

```cpp
[SerializableAttribute]
genric<typename T>
public ref class Bag
```
Type Parameters

T

The type exposes the following members.
### Public Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bag&lt;Of&lt;T&gt;&gt;Bag&lt;Of&lt;T&gt;&gt;New</td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Adds a new item to the bag. Since bags can contain duplicate items, the item is added even if the bag already contains an item equal to item. In this case, the count of items for the representative item is increased by one, but the existing representative item is unchanged.</td>
</tr>
<tr>
<td><strong>AddMany</strong></td>
<td>Adds all the items in collection to the bag.</td>
</tr>
<tr>
<td><strong>AddRepresentative</strong></td>
<td>Adds a new item to the bag. Since bags can contain duplicate items, the item is added even if the bag already contains an item equal to item. In this case (unlike Add), the new item becomes the representative item. Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Changes the number of copies of an existing item in the bag, or adds the indicated number of copies of the item to the bag.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all items from the bag.</td>
</tr>
<tr>
<td>(Overrides CollectionBase&lt;Of&lt;T&gt;&gt;::.Clear())</td>
<td>Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment. Makes a deep clone of this bag. A new bag is created with a clone of each element of this bag,</td>
</tr>
<tr>
<td><strong>CloneContents</strong></td>
<td>by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Determines if this bag contains an item equal to item. The bag is not changed.</td>
</tr>
<tr>
<td><strong>ConvertAll</strong>&lt;br&gt;<code>&lt;TOutput&gt;</code></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.</td>
</tr>
<tr>
<td><strong>CountWhere</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td><strong>Difference</strong></td>
<td>Computes the difference of this bag with another bag. The difference of these two bags is all items that appear in this bag, but not in otherBag. If an item appears X times in this bag, and Y times in the other bag, the difference contains the item <code>X - Y</code> times (zero times if <code>Y &gt;= X</code>). A new bag is created with the difference of the bags and is returned. This bag and the other bag are unchanged.</td>
</tr>
<tr>
<td><strong>DifferenceWith</strong></td>
<td>Computes the difference of this bag with another bag. The difference of these two bags is all items that appear in this bag, but not in otherBag. If an item appears X times in this bag, and Y times in the other bag, the difference contains the item <code>X - Y</code> times (zero times if <code>Y &gt;= X</code>). This bag receives the difference of the two bags; the other bag is unchanged.</td>
</tr>
<tr>
<td><strong>Enumerates all the items in the bag, but</strong></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>DistinctItems</strong></td>
<td>Enumerates equal items just once, even if they occur multiple times in the bag.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate.</td>
</tr>
<tr>
<td><strong>FindAll</strong></td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td><strong>ForEach</strong></td>
<td>Performs the specified action on each item in this collection.</td>
</tr>
</tbody>
</table>
| **GetEnumerator**      | Returns an enumerator that enumerates all the items in the bag. If an item is present multiple times in the bag, the representative item is yielded by the enumerator multiple times. The order of enumeration is haphazard and may change. (Overrides `CollectionBase<Of <T>>...:GetEnumerator()`.)
| **GetHashCode**         | Serves as a hash function for a particular type. `GetHashCode()` is suitable for use in hashing algorithms and data structures like a hash table. (Inherited from `Object`.) |
| **GetRepresentativeItem** | Returns the representative item stored in the bag that is equal to the provided item. Also returns the number of copies of the item in the bag. |
| **GetType**            | Gets the `Type` of the current instance. (Inherited from `Object`.)                                    |
| **Intersection**       | Computes the intersection of this bag with another bag. The intersection of two bags is all items that appear in both of the bags. If an item appears X times in one bag, and Y times in the other bag, the intersection contains the item Minimum(X,Y) times. A new bag is created with the intersection of the bags and is returned. This bag and the other |
Computes the intersection of this bag with another bag. The intersection of two bags is all items that appear in both of the bags. If an item appears $X$ times in one bag, and $Y$ times in the other bag, the sum contains the item $\text{Minimum}(X,Y)$ times. This bag receives the intersection of the two bags, the other bag is unchanged.

### IntersectionWith

Determines if this bag is disjoint from another bag. Two bags are disjoint if no item from one set is equal to any item in the other bag.

### IsDisjointFrom

Determines if this bag is equal to another bag.

### IsEqualTo

This bag is equal to `otherBag` if they contain the same number of of copies of equal elements.

### IsProperSubsetOf

Determines if this bag is a proper subset of another bag. Neither bag is modified. This bag is a subset of `otherBag` if every element in this bag is also in `otherBag`, at least the same number of times. Additional, this bag must have strictly fewer items than `otherBag`.

### IsProperSupersetOf

Determines if this bag is a proper superset of another bag. Neither bag is modified. This bag is a proper superset of `otherBag` if every element in `otherBag` is also in this bag, at least the same number of times. Additional, this bag must have strictly more items than `otherBag`.

### IsSubsetOf

Determines if this bag is a subset of another bag. Neither bag is modified. This bag is a subset of `otherBag` if every element in `otherBag` is also in this bag, at least the same number of times.

### IsSupersetOf

Determines if this bag is a superset of another bag. Neither bag is modified. This bag is a superset of `otherBag` if every element in `otherBag` is also in this bag, at least the same number of times.

### NumberOfCopies

Returns the number of copies of item in the bag.

### Remove

Searches the bag for one item equal to `item`, and if found, removes it from the bag. If not found, the bag is unchanged.

Removes all the items in the collection that satisfy
- **RemoveAll**  
  the condition defined by predicate.  
  (Inherited from `CollectionBase<Of <T>>`).

- **RemoveAllCopies**  
  Searches the bag for all items equal to item, and  
  removes all of them from the bag. If not found, the  
  bag is unchanged.

- **RemoveMany**  
  Removes all the items in collection from the bag.  
  Items that are not present in the bag are ignored.

- **Sum**  
  Computes the sum of this bag with another bag.  
  The sum of two bags is all items from both of the bags.  
  If an item appears X times in one bag, and Y times  
  in the other bag, the sum contains the item \((X+Y)\)  
  times. A new bag is created with the sum of the  
  bags and is returned. This bag and the other bag  
  are unchanged.

- **SumWith**  
  Computes the sum of this bag with another bag.  
  The sum of two bags is all items from both of the  
  bags. If an item appears X times in one bag, and Y  
  times in the other bag, the sum contains the item  
  \((X+Y)\) times. This bag receives the sum of the two  
  bags, the other bag is unchanged.

- **SymmetricDifference**  
  Computes the symmetric difference of this bag  
  with another bag. The symmetric difference of  
  two bags is all items that appear in either of the  
  bags, but not both. If an item appears X times in  
  one bag, and Y times in the other bag, the  
  symmetric difference contains the item  
  \(\text{AbsoluteValue}(X - Y)\) times. A new bag is created  
  with the symmetric difference of the bags and is  
  returned. This bag and the other bag are  
  unchanged.

- **SymmetricDifferenceWith**  
  Computes the symmetric difference of this bag  
  with another bag. The symmetric difference of  
  two bags is all items that appear in either of the  
  bags, but not both. If an item appears X times in  
  one bag, and Y times in the other bag, the  
  symmetric difference contains the item  
  \(\text{AbsoluteValue}(X - Y)\) times. This bag receives the
symmetric difference of the two bags; the other bag is unchanged. Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from `CollectionBase<Of <T>>`.)

**ToString**

Shows the string representation of the collection. The string representation contains a list of the items in the collection. Contained collections (except string) are expanded recursively. (Inherited from `CollectionBase<Of <T>>`.)

**TrueForAll**

Determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from `CollectionBase<Of <T>>`.)

**Union**

Computes the union of this bag with another bag. The union of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the union contains the item Maximum(X,Y) times. A new bag is created with the union of the bags and is returned. This bag and the other bag are unchanged.

Computes the union of this bag with another bag. The union of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the union contains the item Maximum(X,Y) times. This bag receives the union of the two bags, the other bag is unchanged.
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a> is reclaimed by garbage collection. (Inherited from <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>. (Inherited from <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.)</td>
</tr>
</tbody>
</table>
### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comparer</strong></td>
<td>Returns the IEqualityComparer(\langle T\rangle) used to compare items in this bag.</td>
</tr>
<tr>
<td><strong>Count</strong></td>
<td>Returns the number of items in the bag. (Overrides CollectionBase(\langle\text{Of} \langle T\rangle\rangle)::Count.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;</code>Of <code>T&gt;</code>::IsReadOnly</td>
<td>(Inherited from <code>CollectionBase&lt;</code>Of <code>T&gt;</code>.) Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.</td>
</tr>
<tr>
<td><code>ICollection::&lt;CopyTo</code></td>
<td>(Inherited from <code>CollectionBase&lt;</code>Of <code>T&gt;</code>.) Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td><code>ICollection::&lt;IsSynchronized</code></td>
<td>(Inherited from <code>CollectionBase&lt;</code>Of <code>T&gt;</code>.) Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td><code>ICollection::&lt;SyncRoot</code></td>
<td>(Inherited from <code>CollectionBase&lt;</code>Of <code>T&gt;</code>.) Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection.</td>
</tr>
<tr>
<td><code>IEnumerable::&lt;GetEnumerator</code></td>
<td>(Inherited from <code>CollectionBase&lt;</code>Of <code>T&gt;</code>.) Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td><code>ICloneable::&lt;Clone</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

Bag<Of <T> >
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>> Constructor

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bag&lt;(Of &lt;T&gt;)&gt;Bag&lt;(Of &lt;T&gt;)&gt;New()</td>
<td>Creates a new Bag.</td>
</tr>
<tr>
<td>Bag&lt;(Of &lt;T&gt;)&gt;Bag&lt;(Of &lt;T&gt;)&gt;New(IEnumerable&lt;(Of &lt;T&gt;)&gt;)</td>
<td>Creates a new Bag. The bag is initialized with all the items in the given collection.</td>
</tr>
<tr>
<td>Bag&lt;(Of &lt;T&gt;)&gt;Bag&lt;(Of &lt;T&gt;)&gt;New(IEqualityComparer&lt;(Of &lt;T&gt;)&gt;)</td>
<td>Creates a new Bag. The Equals and GetHashCode methods of the passed comparison object will be used to compare items in this bag for equality.</td>
</tr>
<tr>
<td>Bag&lt;(Of &lt;T&gt;)&gt;Bag&lt;(Of &lt;T&gt;)&gt;New(IEnumerable&lt;(Of &lt;T&gt;)&gt;, IEqualityComparer&lt;(Of &lt;T&gt;)&gt;)</td>
<td>Creates a new Bag. The Equals and GetHashCode methods of the passed comparison object will be used to compare items in this bag. The bag is initialized with all the items in the given collection.</td>
</tr>
</tbody>
</table>
See Also

Bag(Of T) Class
Bag(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>> Constructor

See Also

Visual Basic (Declaration) ▯ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new Bag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public Bag()

Visual Basic (Declaration)

Public Sub New

Visual C++

public:
Bag ()
Remarks

Items that are null are permitted.
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>> Constructor (IEnumerable<Of <T>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new Bag. The bag is initialized with all the items in the given collection.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public Bag( IEnumerableView<T> collection )
```

Visual Basic (Declaration)

```vbnet
Public Sub New( _
    collection As IEnumerableView(Of T) _
)
```

Visual C++

```cpp
public: 
Bag ( 
    IEnumerableView<T>* collection 
)
```

Parameters

collection
    IEnumerableView(Of T*)
    A collection with items to be placed into the Bag.
Remarks

Items that are null are permitted.
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag(Of <T>) Constructor (IEqualityComparer(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new Bag. The Equals and GetHashCode methods of the passed comparison object will be used to compare items in this bag for equality.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public Bag(
    IEqualityComparer<T> equalityComparer
)

Visual Basic (Declaration)

Public Sub New (_
    equalityComparer As IEqualityComparer(Of T) _
)

Visual C++

public:  
Bag (  
    IEqualityComparer<T>^ equalityComparer
)

Parameters

equalityComparer  
    IEqualityComparer<(Of <T>)>  
    An instance of IEqualityComparer<T> that will be used to compare items.
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag(Of{T})> Constructor (IEnumerable(Of{T}>, IEqualityComparer(Of{T}>)>

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new Bag. The Equals and GetHashCode methods of the passed comparison object will be used to compare items in this bag. The bag is initialized with all the items in the given collection.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public Bag(
    IEnumerable<T> collection,
    IEqualityComparer<T> equalityComparer
)

Visual Basic (Declaration)

Public Sub New ( _
    collection As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
)

Visual C++

public:
Bag ( 
    IEnumerable<T>^ collection,
    IEqualityComparer<T>^ equalityComparer
)

Parameters

collection
    IEnumerable<(Of <T>)>
    A collection with items to be placed into the Bag.

equalityComparer
    IEqualityComparer<(Of <T>)>
    An instance of IEqualityComparer<T> that will be used to compare items.
See Also

Bag<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag(Of T) Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

```csharp
[SerializableAttribute]
public class Bag<T>
```

Visual Basic (Declaration)

```vbnet
<SerializableAttribute> _
Public Class Bag(Of T)
```

Visual C++

```cpp
[SerializableAttribute]
generic<typename T>
public ref class Bag
```
Type Parameters

T

The type exposes the following methods.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Adds a new item to the bag. Since bags can contain duplicate items, the item is added even if the bag already contains an item equal to item. In this case, the count of items for the representative item is increased by one, but the existing representative item is unchanged.</td>
</tr>
<tr>
<td><strong>AddMany</strong></td>
<td>Adds all the items in collection to the bag.</td>
</tr>
<tr>
<td><strong>AddRepresentative</strong></td>
<td>Adds a new item to the bag. Since bags can contain duplicate items, the item is added even if the bag already contains an item equal to item. In this case (unlike Add), the new item becomes the representative item. Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase(Of &lt;T&gt;).)</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Changes the number of copies of an existing item in the bag, or adds the indicated number of copies of the item to the bag.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all items from the bag. (Overrides CollectionBase(Of &lt;T&gt;)::.Clear())</td>
</tr>
<tr>
<td><strong>Clone</strong></td>
<td>Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment. Makes a deep clone of this bag. A new bag is created with a clone of each element of this bag,</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>CloneContents</strong></td>
<td>by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Determines if this bag contains an item equal to item. The bag is not changed.</td>
</tr>
<tr>
<td><strong>ConvertAll(Of TOutput)&gt;</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.</td>
</tr>
<tr>
<td><strong>CountWhere</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td><strong>Difference</strong></td>
<td>Computes the difference of this bag with another bag. The difference of these two bags is all items that appear in this bag, but not in otherBag. If an item appears X times in this bag, and Y times in the other bag, the difference contains the item X - Y times (zero times if Y &gt;= X). A new bag is created with the difference of the bags and is returned. This bag and the other bag are unchanged.</td>
</tr>
<tr>
<td><strong>DifferenceWith</strong></td>
<td>Computes the difference of this bag with another bag. The difference of these two bags is all items that appear in this bag, but not in otherBag. If an item appears X times in this bag, and Y times in the other bag, the difference contains the item X - Y times (zero times if Y &gt;= X). This bag receives the difference of the two bags; the other bag is unchanged.</td>
</tr>
<tr>
<td><strong>Enumerates</strong></td>
<td>Enumerates all the items in the bag, but</td>
</tr>
</tbody>
</table>
**DistinctItems**

Enumerates equal items just once, even if they occur multiple times in the bag.

**Equals**

Determines whether the specified `Object` is equal to the current `Object`.
(Inherited from `Object`.)

**Exists**

Determines if the collection contains any item that satisfies the condition defined by predicate.
(Inherited from `CollectionBase<Of <T>>`.)

**FindAll**

Enumerates the items in the collection that satisfy the condition defined by predicate.
(Inherited from `CollectionBase<Of <T>>`.)

**ForEach**

Performs the specified action on each item in this collection.
(Inherited from `CollectionBase<Of <T>>`.)

**GetEnumerator**

Returns an enumerator that enumerates all the items in the bag. If an item is present multiple times in the bag, the representative item is yielded by the enumerator multiple times. The order of enumeration is haphazard and may change.
(Overrides `CollectionBase<Of <T>>`.::GetEnumerator().)

**GetHashCode**

Serves as a hash function for a particular type. `GetHashCode()` is suitable for use in hashing algorithms and data structures like a hash table.
(Inherited from `Object`.)

**GetRepresentativeItem**

Returns the representative item stored in the bag that is equal to the provided item. Also returns the number of copies of the item in the bag.

**GetType**

Gets the `Type` of the current instance.
(Inherited from `Object`.)

**Intersection**

Computes the intersection of this bag with another bag. The intersection of two bags is all items that appear in both of the bags. If an item appears X times in one bag, and Y times in the other bag, the intersection contains the item Minimum(X,Y) times. A new bag is created with the intersection of the bags and is returned. This bag and the other
Computes the intersection of this bag with another bag. The intersection of two bags is all items that appear in both of the bags. If an item appears $X$ times in one bag, and $Y$ times in the other bag, the sum contains the item $\text{Minimum}(X,Y)$ times. This bag receives the intersection of the two bags, the other bag is unchanged.

Determines if this bag is disjoint from another bag. Two bags are disjoint if no item from one set is equal to any item in the other bag.

Determines if this bag is equal to another bag. This bag is equal to otherBag if they contain the same number of copies of equal elements.

Determines if this bag is a proper subset of another bag. Neither bag is modified. This bag is a subset of otherBag if every element in this bag is also in otherBag, at least the same number of times. Additional, this bag must have strictly fewer items than otherBag.

Determines if this bag is a proper superset of another bag. Neither bag is modified. This bag is a proper superset of otherBag if every element in otherBag is also in this bag, at least the same number of times. Additional, this bag must have strictly more items than otherBag.

Determines if this bag is a subset of another bag. Neither bag is modified. This bag is a subset of otherBag if every element in otherBag is also in this bag, at least the same number of times.

Determines if this bag is a superset of another bag. Neither bag is modified. This bag is a superset of otherBag if every element in otherBag is also in this bag, at least the same number of times.

Returns the number of copies of item in the bag.

Searches the bag for one item equal to item, and if found, removes it from the bag. If not found, the bag is unchanged.

Removes all the items in the collection that satisfy
- **RemoveAll**
  the condition defined by predicate.
  (Inherited from CollectionBase<Of <T>>.)

- **RemoveAllCopies**
  Searches the bag for all items equal to item, and
  removes all of them from the bag. If not found, the
  bag is unchanged.

- **RemoveMany**
  Removes all the items in collection from the bag.
  Items that are not present in the bag are ignored.

- **Sum**
  Computes the sum of this bag with another bag. The
  sum of two bags is all items from both of the bags.
  If an item appears X times in one bag, and Y times
  in the other bag, the sum contains the item (X+Y)
  times. A new bag is created with the sum of the
  bags and is returned. This bag and the other bag
  are unchanged.

- **SumWith**
  Computes the sum of this bag with another bag. The
  sum of two bags is all items from both of the bags.
  If an item appears X times in one bag, and Y times
  in the other bag, the sum contains the item (X+Y)
  times. This bag receives the sum of the two
  bags, the other bag is unchanged.

- **SymmetricDifference**
  Computes the symmetric difference of this bag
  with another bag. The symmetric difference of
  two bags is all items that appear in either of the
  bags, but not both. If an item appears X times in
  one bag, and Y times in the other bag, the
  symmetric difference contains the item
  AbsoluteValue(X - Y) times. A new bag is created
  with the symmetric difference of the bags and is
  returned. This bag and the other bag are
  unchanged.

- **SymmetricDifferenceWith**
  Computes the symmetric difference of this bag
  with another bag. The symmetric difference of
  two bags is all items that appear in either of the
  bags, but not both. If an item appears X times in
  one bag, and Y times in the other bag, the
  symmetric difference contains the item
  AbsoluteValue(X - Y) times. This bag receives the
symmetric difference of the two bags; the other bag is unchanged.

- **ToArray**
  Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.
  (Inherited from `CollectionBase<Of <T>>`.)

- **ToString**
  Shows the string representation of the collection.
  The string representation contains a list of the items in the collection. Contained collections (except string) are expanded recursively.
  (Inherited from `CollectionBase<Of <T>>`.)

- **TrueForAll**
  Determines if all of the items in the collection satisfy the condition defined by predicate.
  (Inherited from `CollectionBase<Of <T>>`.)

- **Union**
  Computes the union of this bag with another bag. The union of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the union contains the item Maximum(X,Y) times. A new bag is created with the union of the bags and is returned. This bag and the other bag are unchanged.

- **UnionWith**
  Computes the union of this bag with another bag. The union of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the union contains the item Maximum(X,Y) times. This bag receives the union of the two bags, the other bag is unchanged.
### Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection::CopyTo</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>IEnumerable::GetEnumerator</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>ICloneable::Clone</td>
<td>Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
</tbody>
</table>
See Also

Bag<(Of <T>))
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Add a new item to the bag. Since bags can contain duplicate items, the item is added even if the bag already contains an item equal to item. In this case, the count of items for the representative item is increased by one, but the existing representative item is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public void Add(
    T item
)

Visual Basic (Declaration)

Public Sub Add (_
    item As T _
)

Visual C++

public:
    virtual void Add ( 
    T item 
) sealed

Parameters

item
    T
    The item to add to the bag.
Remarks

Adding an item takes approximately constant time, regardless of the number of items in the bag.
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>>...::AddMany Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds all the items in collection to the bag.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public void AddMany(
    IEnumerable<T> collection
)
```

### Visual Basic (Declaration)

```vbnet
Public Sub AddMany ( _
    collection As IEnumerable(Of T) _
)
```

### Visual C++

```cpp
public:
    void AddMany ( 
        IEnumerable<T>^ collection
    )
```

## Parameters

- `collection`:
  ```csharp
  IEnumerable<T> collection
  ```
  A collection of items to add to the bag.
Remarks

Adding the collection takes time $O(M \log N)$, where $N$ is the number of items in the bag, and $M$ is the number of items in collection.
See Also

Bag<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>>::AddRepresentative Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a new item to the bag. Since bags can contain duplicate items, the item is added even if the bag already contains an item equal to item. In this case (unlike Add), the new item becomes the representative item.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public void AddRepresentative(
    T item
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub AddRepresentative (_
    item As T _
)
```

**Visual C++**

```cpp
public:
void AddRepresentative (    
    T item
)
```

### Parameters

- **item**
  - T
    - The item to add to the bag.
Remarks

Adding an item takes approximately constant time, regardless of the number of items in the bag.
See Also

Bag(Of<T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>>...::ChangeNumberOfCopies Method

See Also

☐ Visual Basic (Declaration) ☐ Visual Basic (Usage)
☐ C#
☐ Visual C++
☐ J#
☐ JScript
☐ XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Changes the number of copies of an existing item in the bag, or adds the indicated number of copies of the item to the bag.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public void ChangeNumberOfCopies(
    T item,
    int numCopies
)
```

Visual Basic (Declaration)

```vbnet
Public Sub ChangeNumberOfCopies ( _
    item As T, _
    numCopies As Integer _
)
```

Visual C++

```cpp
public:
void ChangeNumberOfCopies ( 
    T item,
    int numCopies
)
```

Parameters

**item**

T

The item to change the number of copies of. This may or may not already be present in the bag.

**numCopies**

Int32

The new number of copies of the item.
Remarks

Changing the number of copies takes approximately constant time, regardless of the number of items in the bag.
See Also

Bag<\(\text{Of } T\)\> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<
(Of <T>)>::Clear Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all items from the bag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public override sealed void Clear()
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Sub Clear
```

**Visual C++**

```cpp
public:
virtual void Clear () override sealed
```
Remarks

Clearing the bag takes a constant amount of time, regardless of the number of items in it.
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>>...::Clone Method

See Also

Visual Basic (Declaration) □ Visual Basic (Usage)
□ C#
□ Visual C++
□ J#
□ JScript
□ XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public Bag<T> Clone()

Visual Basic (Declaration)

Public Function Clone As Bag(Of T)

Visual C++

public:
Bag<T>^ Clone()

Return Value

The cloned bag.
Remarks

Cloning the bag takes time $O(N)$, where $N$ is the number of unique items in the bag.
See Also

Bag<(Of <T>)> Class
Wintellect.Powercollections Namespace

Send comments about this topic to Microsoft.
Bag<(Of <T>)>...::CloneContents Method

See Also

- Visual Basic (Declaration) - Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a deep clone of this bag. A new bag is created with a clone of each element of this bag, by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public Bag<T> CloneContents()

Visual Basic (Declaration)

Public Function CloneContents As Bag(Of T)

Visual C++

public:
Bag<T>^ CloneContents ()

Return Value

The cloned bag.
Remarks

If T is a reference type, it must implement ICloneable. Otherwise, an InvalidOperationException is thrown.

Cloning the bag takes time $O(N \log N)$, where $N$ is the number of items in the bag.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::InvalidOperationException</td>
<td>T is a reference type that does not implement ICloneable.</td>
</tr>
</tbody>
</table>


**See Also**

Bag<(Of <T>)> Class  
Wintellect.PowerCollections Namespace

Send [comments](#) about this topic to Microsoft.
Bag(Of T)::.Contains Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this bag contains an item equal to item. The bag is not changed.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
-Syntax

  **C#**

  ```csharp
  public bool Contains(
      T item
  )
  ```

  **Visual Basic (Declaration)**

  ```vbnet
  Public Function Contains ( _
      item As T _
  ) As Boolean
  ```

  **Visual C++**

  ```cpp
  public:
  virtual bool Contains ( 
      T item
  ) sealed
  ```

  **Parameters**

  item
      T
  The item to search for.

  **Return Value**

  True if the bag contains item. False if the bag does not contain item.
Remarks

Searching the bag for an item takes time $O(\log N)$, where $N$ is the number of items in the bag.
See Also

Bag<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Computes the difference of this bag with another bag. The difference of these two bags is all items that appear in this bag, but not in otherBag. If an item appears X times in this bag, and Y times in the other bag, the difference contains the item X - Y times (zero times if Y >= X). A new bag is created with the difference of the bags and is returned. This bag and the other bag are unchanged.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public Bag<T> Difference(
    Bag<T> otherBag
)
```

#### Visual Basic (Declaration)

```vbnet
Public Function Difference ( _
    otherBag As Bag(Of T) _
) As Bag(Of T)
```

#### Visual C++

```cpp
public:
Bag<T>^ Difference ( _
    Bag<T>^ otherBag
)
```

### Parameters

otherBag

- `Bag<Of <T>>` Bag to difference with.

### Return Value

The difference of the two bags.
Remarks

The difference of two bags is computed in time $O(M + N)$, where $M$ and $N$ are the size of the two bags.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::InvalidOperationException</td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag\<(Of \<T\>)\> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>>...::DifferenceWith Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the difference of this bag with another bag. The difference of these two bags is all items that appear in this bag, but not in otherBag. If an item appears X times in this bag, and Y times in the other bag, the difference contains the item X - Y times (zero times if Y >= X). This bag receives the difference of the two bags; the other bag is unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

public void DifferenceWith(
    Bag<T> otherBag
)

**Visual Basic (Declaration)**

Public Sub DifferenceWith ( _
    otherBag As Bag(Of T) _
)

**Visual C++**

public:
void DifferenceWith ( 
    Bag<T>* otherBag
)

**Parameters**

otherBag
    Bag<Of T>]
    Bag to difference with.
Remarks

The difference of two bags is computed in time $O(M)$, where $M$ is the size of the other bag.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Enumerates all the items in the bag, but enumerates equal items just once, even if they occur multiple times in the bag.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public IEnumerable<T> DistinctItems()
```

**Visual Basic (Declaration)**

```vbnet
Public Function DistinctItems As IEnumerable(Of T)
```

**Visual C++**

```cpp
public: 
IEnumerable<T>^ DistinctItems ()
```

**Return Value**

An IEnumerable<T> that enumerates the unique items.
Remarks

If the bag is changed while items are being enumerated, the enumeration will terminate with an InvalidOperationException.
See Also

Bag<Of<T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Returns an enumerator that enumerates all the items in the bag. If an item is present multiple times in the bag, the representative item is yielded by the enumerator multiple times. The order of enumeration is haphazard and may change.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public override sealed IEnumerator<T> GetEnumerator()
```

### Visual Basic (Declaration)

```vbnet
Public Overrides NotOverridable Function GetEnumerator As IEnumerator
```

### Visual C++

```cpp
public:
virtual IEnumerator<T>^ GetEnumerator() override sealed
```

## Return Value

An enumerator for enumerating all the items in the Bag.
Typically, this method is not called directly. Instead the "foreach" statement is used to enumerate the items, which uses this method implicitly.

If an item is added to or deleted from the bag while it is being enumerated, then the enumeration will end with an InvalidOperationException.

Enumeration all the items in the bag takes time $O(N)$, where $N$ is the number of items in the bag.
See Also

Bag<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Returns the representative item stored in the bag that is equal to the provided item. Also returns the number of copies of the item in the bag.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public int GetRepresentativeItem(
    T item,
    out T representative
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function GetRepresentativeItem ( _
    item As T, _
    <OutAttribute> ByRef representative As T _
) As Integer
```

**Visual C++**

```cpp
public:
    int GetRepresentativeItem ( 
    T item,
    [OutAttribute] T% representative
    )
```

### Parameters

**item**

- **T**
  - Item to find in the bag.

**representative**

- **T%**
  - If one or more items equal to item are present in the bag, returns the representative item. If no items equal to item are stored in the bag, returns item.

### Return Value
The number of items equal to item stored in the bag.
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<(Of <T>)>::Intersection Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the intersection of this bag with another bag. The intersection of two bags is all items that appear in both of the bags. If an item appears X times in one bag, and Y times in the other bag, the intersection contains the item Minimum(X, Y) times. A new bag is created with the intersection of the bags and is returned. This bag and the other bag are unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public Bag<T> Intersection(
    Bag<T> otherBag
)

Visual Basic (Declaration)

Public Function Intersection ( _
    otherBag As Bag(Of T) _
) As Bag(Of T)

Visual C++

public:
Bag<T>^ Intersection ( _
    Bag<T>^ otherBag
)

Parameters

otherBag

Bag<(Of <T>)>

Bag to intersection with.

Return Value

The intersection of the two bags.
Remarks

When equal items appear in both bags, the intersection will include an arbitrary choice of one of the two equal items.

The intersection of two bags is computed in time $O(N)$, where $N$ is the size of the smaller bag.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::InvalidOperationException</td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag(Of T)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>>...::IntersectionWith Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the intersection of this bag with another bag. The intersection of two bags is all items that appear in both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item Minimum(X,Y) times. This bag receives the intersection of the two bags, the other bag is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public void IntersectionWith(
    Bag<T> otherBag
)

Visual Basic (Declaration)

Public Sub IntersectionWith ( _
   otherBag As Bag(Of T) _
)

Visual C++

public:
void IntersectionWith ( 
    Bag<T>^ otherBag
)

Parameters

otherBag

Bag<Of <T>>
Bag to intersection with.
Remarks

When equal items appear in both bags, the intersection will include an arbitrary choice of one of the two equal items.

The intersection of two bags is computed in time $O(N)$, where $N$ is the size of the smaller bag.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of T>...:IsDisjointFrom Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this bag is disjoint from another bag. Two bags are disjoint if no item from one set is equal to any item in the other bag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public bool IsDisjointFrom(
    Bag<T> otherBag
)

Visual Basic (Declaration)

Public Function IsDisjointFrom ( _
    otherBag As Bag(Of T) _
) As Boolean

Visual C++

public:
    bool IsDisjointFrom ( 
        Bag<T>^ otherBag
    )

Parameters

otherBag
    Bag<(Of <T>)>
    Bag to check disjointness with.

Return Value

True if the two bags are disjoint, false otherwise.
Remarks

The answer is computed in time $O(N)$, where $N$ is the size of the smaller set.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::InvalidOperationException</td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this bag is equal to another bag. This bag is equal to otherBag if they contain the same number of copies of equal elements.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public bool IsEqualTo(
    Bag<T> otherBag
)

Visual Basic (Declaration)

Public Function IsEqualTo ( _
    otherBag As Bag(Of T) _
) As Boolean

Visual C++

public:
bool IsEqualTo ( _
    Bag<T>^ otherBag
)

Parameters

otherBag
    Bag<(Of <T>)>
    Bag to compare to

Return Value

True if this bag is equal to otherBag, false otherwise.
Remarks

IsSupersetOf is computed in time $O(N)$, where $N$ is the number of unique items in this bag.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this bag is a proper subset of another bag. Neither bag is modified. This bag is a subset of otherBag if every element in this bag is also in otherBag, at least the same number of times. Additional, this bag must have strictly fewer items than otherBag.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public bool IsProperSubsetOf(
    Bag<T> otherBag
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function IsProperSubsetOf ( _
    otherBag As Bag(Of T) _
) As Boolean
```

**Visual C++**

```cpp
public:
    bool IsProperSubsetOf ( _
        Bag<T>^ otherBag
    )
```

**Parameters**

otherBag

```csharp
Bag<Of <T>>
```

Bag to compare to.

**Return Value**

True if this is a proper subset of otherBag.
Remarks

IsProperSubsetOf is computed in time $O(N)$, where $N$ is the number of unique items in this bag.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..:::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag(Of T)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this bag is a proper superset of another bag. Neither bag is modified. This bag is a proper superset of otherBag if every element in otherBag is also in this bag, at least the same number of times. Additional, this bag must have strictly more items than otherBag.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#  

public bool IsProperSupersetOf(
    Bag<T> otherBag
)

Visual Basic (Declaration)

Public Function IsProperSupersetOf ( _
    otherBag As Bag(Of T) _
) As Boolean

Visual C++

public:  
bool IsProperSupersetOf (  
    Bag<T>* otherBag
)

Parameters

otherBag
    Bag<Of T>  
    Set to compare to.

Return Value

True if this is a proper superset of otherBag.
Remarks

IsProperSupersetOf is computed in time $O(M)$, where $M$ is the number of unique items in otherBag.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::InvalidOperationException</td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this bag is a subset of another bag items in this bag.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public bool IsSubsetOf(
    Bag<T> otherBag
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function IsSubsetOf ( _
    otherBag As Bag(Of T) _
) As Boolean
```

**Visual C++**

```cpp
public:
  bool IsSubsetOf ( 
    Bag<T^> otherBag
  )
```

### Parameters

**otherBag**

`Bag<Of<T>>`  
Bag to compare to.

### Return Value

True if this is a subset of `otherBag`. 
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag(Of <T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this bag is a superset of another bag. Neither bag is modified. This bag is a superset of otherBag if every element in otherBag is also in this bag, at least the same number of times.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

public bool IsSupersetOf(
            Bag<T> otherBag
)

**Visual Basic (Declaration)**

Public Function IsSupersetOf (_
            otherBag As Bag(Of T) _
) As Boolean

**Visual C++**

public:
bool IsSupersetOf ( 
            Bag<T>^ otherBag
)

**Parameters**

otherBag
            Bag<Of <T>>
            Bag to compare to.

**Return Value**

True if this is a superset of otherBag.
Remarks

IsSupersetOf is computed in time $O(M)$, where $M$ is the number of unique items in otherBag.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag(Of $T$) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>>...::NumberOfCopies Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the number of copies of item in the bag.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
class Bag {
    public int NumberOfCopies(
        T item
    )
}
```

**Visual Basic (Declaration)**

```vbnet
Public Function NumberOfCopies ( _
        item As T _
    ) As Integer
```

**Visual C++**

```cpp
public:
    int NumberOfCopies ( 
        T item
    )
```

**Parameters**

*item*  
*T*  
The item to search for in the bag.

**Return Value**

The number of items in the bag that compare equal to *item*. 
Remarks

NumberOfCopies() takes approximately constant time, no matter how many items are stored in the bag.
See Also

Bag<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>>...::Remove Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Searches the bag for one item equal to item, and if found, removes it from the bag. If not found, the bag is unchanged.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public bool Remove(
    T item
)

Visual Basic (Declaration)

Public Function Remove ( _
    item As T _
) As Boolean

Visual C++

public:
virtual bool Remove ( 
    T item 
) sealed

Parameters

item
    T
    The item to remove.

Return Value

True if item was found and removed. False if item was not in the bag.
Remarks

Equality between items is determined by the comparison instance or delegate used to create the bag.

Removing an item from the bag takes approximated constant time, regardless of the number of items in the bag.
See Also

Bag<Of<T>> Class  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<(Of <T>)>...::RemoveAllCopies Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Searches the bag for all items equal to item, and removes all of them from the bag. If not found, the bag is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
- **Syntax**

  **C#**

  ```csharp
  public int RemoveAllCopies(
          T item
  )
  ```

  **Visual Basic (Declaration)**

  ```vbnet
  Public Function RemoveAllCopies (_
          item As T _
  ) As Integer
  ```

  **Visual C++**

  ```cpp
  public:
  int RemoveAllCopies ( 
          T item
  )
  ```

  **Parameters**

  `item`

  `T`  
  The item to remove.

  **Return Value**

  The number of copies of item that were found and removed.
Remarks

Equality between items is determined by the comparer instance used to create the bag.

RemoveAllCopies() takes time $O(M \log N)$, where $N$ is the total number of items in the bag, and $M$ is the number of items equal to item.
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<T>...::RemoveMany Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all the items in collection from the bag. Items that are not present in the bag are ignored.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
- Syntax

C#

```csharp
public int RemoveMany(
    IEnumerable<T> collection
)
```

Visual Basic (Declaration)

```vbnet
Public Function RemoveMany ( _
    collection As IEnumerable(Of T) _
) As Integer
```

Visual C++

```cpp
public: int RemoveMany ( _
    IEnumerable<T>^ collection
)
```

Parameters

- `collection` `IEnumerable<T>`
  A collection of items to remove from the bag.

Return Value

The number of items removed from the bag.
Remarks

Equality between items is determined by the comparer instance used to create the bag.

Removing the collection takes time $O(M)$, where $M$ is the number of items in collection.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag(Of <T>)..::Sum Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the sum of this bag with another bag. The sum of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item (X+Y) times. A new bag is created with the sum of the bags and is returned. This bag and the other bag are unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public Bag<T> Sum(
    Bag<T> otherBag
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function Sum ( _
    otherBag As Bag(Of T) _
) As Bag(Of T)
```

**Visual C++**

```c++
public:
    Bag<T>^ Sum (_
            Bag<T>^ otherBag
    )
```

### Parameters

**otherBag**

A `Bag<T>` `Bag` of type `<T>`. Bag to sum with.

### Return Value

The sum of the two bags.
Remarks

The sum of two bags is computed in time $O(M + N \log M)$, where $M$ is the size of the larger bag, and $N$ is the size of the smaller bag.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>>::SumWith Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the sum of this bag with another bag. The sum of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item (X+Y) times. This bag receives the sum of the two bags, the other bag is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
# Syntax

**C#**

```csharp
public void SumWith(
    Bag<T> otherBag
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub SumWith ( _
    otherBag As Bag(Of T) _
)
```

**Visual C++**

```cpp
public:
void SumWith ( 
    Bag<T>^ otherBag
)
```

**Parameters**

otherBag

Bag<(Of <T>)>

Bag to sum with.
Remarks

The sum of two bags is computed in time $O(M)$, where $M$ is the size of the other bag.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag(Of 'T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag(Of <T>)::.SymmetricDifference Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the symmetric difference of this bag with another bag. The symmetric difference of two bags is all items that appear in either of the bags, but not both. If an item appears X times in one bag, and Y times in the other bag, the symmetric difference contains the item \(\text{AbsoluteValue}(X - Y)\) times. A new bag is created with the symmetric difference of the bags and is returned. This bag and the other bag are unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
# Syntax

## C#

```csharp
public Bag<T> SymmetricDifference(
    Bag<T> otherBag
)
```

## Visual Basic (Declaration)

```vbnet
Public Function SymmetricDifference ( _
    otherBag As Bag(Of T) _
) As Bag(Of T)
```

## Visual C++

```csharp
public:
    Bag<T>^ SymmetricDifference ( 
        Bag<T>^ otherBag
    )
```

## Parameters

otherBag

Bag<Of<T>>

Bag to symmetric difference with.

## Return Value

The symmetric difference of the two bags.
Remarks

The symmetric difference of two bags is computed in time $O(M + N)$, where $M$ is the size of the larger bag, and $N$ is the size of the smaller bag.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag<Of<T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<(Of <T>)>...::SymmetricDifferenceWith Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the symmetric difference of this bag with another bag. The symmetric difference of two bags is all items that appear in either of the bags, but not both. If an item appears X times in one bag, and Y times in the other bag, the symmetric difference contains the item $\text{AbsoluteValue}(X - Y)$ times. This bag receives the symmetric difference of the two bags; the other bag is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public void SymmetricDifferenceWith(
    Bag<T> otherBag
)
```

Visual Basic (Declaration)

```vbnet
Public Sub SymmetricDifferenceWith ( _
    otherBag As Bag(Of T) _
)
```

Visual C++

```cpp
public:
void SymmetricDifferenceWith ( _
    Bag<T>^ otherBag
)
```

Parameters

otherBag

Bag<Of <T>>
Bag to symmetric difference with.
Remarks

The symmetric difference of two bags is computed in time $O(M + N)$, where $M$ is the size of the larger bag, and $N$ is the size of the smaller bag.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>▷...::System.ICloneable.Clone Method
See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private Object ICloneable.Clone()

Visual Basic (Declaration)

Private Function System.ICloneable.Clone As Object Implements IClone

Visual C++

private:
virtual Object^ System.ICloneable.Clone () sealed = ICloneable::Clone

Return Value

The cloned bag.
Remarks

Cloning the bag takes time $O(N)$, where $N$ is the number of items in the bag.
See Also

Bag<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>>::<Union Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the union of this bag with another bag. The union of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the union contains the item Maximum(X,Y) times. A new bag is created with the union of the bags and is returned. This bag and the other bag are unchanged.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public Bag<T> Union(
    Bag<T> otherBag
)
```

Visual Basic (Declaration)

```vbnet
Public Function Union (  
    otherBag As Bag(Of T)  
) As Bag(Of T)
```

Visual C++

```cpp
public:
    Bag<T>^ Union (  
        Bag<T>^ otherBag
    )
```

Parameters

otherBag
    Bag<Of <T>>
    Bag to union with.

Return Value

The union of the two bags.
Remarks

The union of two bags is computed in time $O(M+N)$, where $M$ and $N$ are the size of the two bags.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..:::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag<T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>>...::UnionWith Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the union of this bag with another bag. The union of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the union contains the item Maximum(X,Y) times. This bag receives the union of the two bags, the other bag is unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public void UnionWith(
    Bag<T> otherBag
)

Visual Basic (Declaration)

Public Sub UnionWith ( _
    otherBag As Bag(Of T) _
)

Visual C++

public:
void UnionWith ( 
    Bag<T>^ otherBag
)

Parameters

otherBag
    Bag<(Of <T>)>
    Bag to union with.
Remarks

The union of two bags is computed in time $O(M+N)$, where $M$ and $N$ are the size of the two bags.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::InvalidOperationException</td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Bag<Of<T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<(Of <T>)> Properties

See Also

☑ Include Inherited Members ☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class Bag<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class Bag(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class Bag
Type Parameters

T

The type exposes the following properties.
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comparer</strong></td>
<td>Returns the IEqualityComparer&lt;T&gt; used to compare items in this bag.</td>
</tr>
<tr>
<td><strong>Count</strong></td>
<td>Returns the number of items in the bag. (Overrrides CollectionBase(Of T)&gt;::Count.)</td>
</tr>
</tbody>
</table>
# Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| ICollection`<Of` `<T>`>`::IsReadOnly | (Inherited from `CollectionBase`<Of `<T>`>). Indicates whether the collection is synchronized. (Inherited from `CollectionBase`<Of `<T>`).)
| ICollection`::IsSynchronized` | Indicates the synchronization object for this collection. (Inherited from `CollectionBase`<Of `<T>`).)
| ICollection`::SyncRoot` |
See Also

Bag<\(\text{Of } T\)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag(Of `T`)...::Comparer Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the IEqualityComparer`<T>` used to compare items in this bag.

Namespace: Wintellect.PowerCollections

Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public IEqualityComparer<T> Comparer{ get; }
```

**Visual Basic (Declaration)**

```vbnet
Public ReadOnly Property Comparer As IEqualityComparer(Of T)
```

**Visual C++**

```cpp
public:
property IEqualityComparer<T>^ Comparator {
     IEqualityComparer<T>^ get (); 
}
```

**Field Value**

If the bag was created using a comparer, that comparer is returned. Otherwise the default comparer for T (EqualityComparer<T>.Default) is returned.
See Also

Bag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Bag<Of <T>>...::Count Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the number of items in the bag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed int Count{ get;}

Visual Basic (Declaration)

Public Overrides NotOverridable ReadOnly Property Count As Integer

Visual C++

public:
virtual property int Count {
    int get () override sealed;
}

Field Value

The number of items in the bag.
Remarks

The size of the bag is returned in constant time.
See Also

`Bag(Of T)` Class
`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
BigList<T> provides a list of items, in order, with indices of the items ranging from 0 to one less than the count of items in the collection. BigList<T> is optimized for efficient operations on large (>100 items) lists, especially for insertions, deletions, copies, and concatenations.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

[SerializableAttribute]
public class BigList<T> : ListBase<T>, ICloneable

Visual Basic (Declaration)

<SerializableAttribute>
Public Class BigList(Of T)
    Inherits ListBase(Of T)
    Implements ICloneable

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class BigList : public ListBase<T>,
    ICloneable
- **Type Parameters**

  T
  
  The type of items to store in the BigList.
BigList<T> class is similar in functionality to the standard List<T> class. Both classes provide a collection that stores an set of items in order, with indices of the items ranging from 0 to one less than the count of items in the collection. Both classes provide the ability to add and remove items from any index, and the get or set the item at any index.

BigList<T> differs significantly from List<T> in the performance of various operations, especially when the lists become large (several hundred items or more). With List<T>, inserting or removing elements from anywhere in a large list except the end is very inefficient -- every item after the point of inserting or deletion has to be moved in the list. The BigList<T> class, however, allows for fast insertions and deletions anywhere in the list. Furthermore, BigList<T> allows copies of a list, sub-parts of a list, and concatenations of two lists to be very fast. When a copy is made of part or all of a BigList, two lists shared storage for the parts of the lists that are the same. Only when one of the lists is changed is additional memory allocated to store the distinct parts of the lists.

Of course, there is a small price to pay for this extra flexibility. Although still quite efficient, using an index to get or change one element of a BigList, while still reasonably efficient, is significantly slower than using a plain List. Because of this, if you want to process every element of a BigList, using a foreach loop is a lot more efficient than using a for loop and indexing the list.

In general, use a List when the only operations you are using are Add (to the end), foreach, or indexing, or you are very sure the list will always remain small (less than 100 items). For large (>100 items) lists that do insertions, removals, copies, concatenations, or sub-ranges, BigList will be more efficient than List. In almost all cases, BigList is more efficient and easier to use than LinkedList.
Inheritance Hierarchy

System::Object
Wintellect.PowerCollections::CollectionBase<T>
Wintellect.PowerCollections::ListBase<T>
Wintellect.PowerCollections::BigList<T>
See Also

BigList(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>> Members

See Also  Methods  Constructors  Properties  Explicit Interface Implementations

☐ Include Inherited Members ☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#  

[SerializableAttribute]  
public class BigList<T> 

Visual Basic (Declaration)  

<SerializableAttribute>  
Public Class BigList(Of T) 

Visual C++  

[SerializableAttribute]  
generic<typeName T>  
public ref class BigList
Type Parameters

T

The type exposes the following members.
# Public Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BigList&lt;Of &lt;T&gt;&gt;BigList&lt;Of &lt;T&gt;&gt;New</code></td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Adds an item to the end of the BigList. The indices of all existing items in the Deque are unchanged. (Overrides ListBase&lt;Of &lt;T&gt;&gt;::Add(T).)</td>
</tr>
<tr>
<td><strong>Addition</strong></td>
<td>Concatenates two lists together to create a new list. Both lists being concatenated are unchanged. The resulting list contains all the items in first, followed by all the items in second.</td>
</tr>
<tr>
<td><strong>AddRange</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>AddRangeToFront</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>AddToFront</strong></td>
<td>Adds an item to the beginning of the BigList. The indices of all existing items in the Deque are increased by one, and the new item has index zero.</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this list. The returned IList&lt;T&gt; provides a view of the list that prevents modifications to the list. Use the method to provide access to the list without allowing changes. Since the returned object is just a view, changes to the list will be reflected in the view. (Inherited from ListBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>BinarySearch</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all of the items from the BigList. (Overrides ListBase&lt;Of &lt;T&gt;&gt;::Clear().)</td>
</tr>
<tr>
<td><strong>Clone</strong></td>
<td>Creates a new BigList that is a copy of this list.</td>
</tr>
<tr>
<td><strong>CloneContents</strong></td>
<td>Makes a deep clone of this BigList. A new BigList is created with a clone of each element of this set, by calling ICloneable.Clone on each element. If T is a value type, then this method is the same as Clone.</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Determines if the list contains any item that compares equal to item. The implementation simply checks whether IndexOf(item) returns a non-negative value. (Inherited from ListBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>ConvertAll(Of TDest)&lt;&gt;</code></td>
<td>Convert the list to a new list by applying a delegate to each item in the collection. The resulting list contains the result of applying converter to each item in the list, in order. The current list is unchanged.</td>
</tr>
<tr>
<td><code>CopyTo</code></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><code>CountWhere</code></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from <code>CollectionBase(Of &lt;T&gt;)&lt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>Equals</code></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>Exists</code></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate. (Inherited from <code>CollectionBase(Of &lt;T&gt;)&lt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>Find</code></td>
<td>Finds the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for <code>T</code> (null or all-zero) is returned. (Inherited from <code>ListBase(Of &lt;T&gt;)&lt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>FindAll</code></td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from <code>CollectionBase(Of &lt;T&gt;)&lt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>FindIndex</code></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><code>FindLast</code></td>
<td>Finds the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for <code>T</code> (null or all-zero) is returned. (Inherited from <code>ListBase(Of &lt;T&gt;)&lt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>FindLastIndex</code></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><code>ForEach</code></td>
<td>Performs the specified action on each item in this collection. (Inherited from <code>CollectionBase(Of &lt;T&gt;)&lt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>GetEnumerator</code></td>
<td>Enumerates all of the items in the list, in order. The item at index 0 is enumerated first, then the item at index 1, and so on. Usually, the foreach statement is used to call this method implicitly. (Overrides <code>ListBase&lt;Of &lt;T&gt;)&lt;&gt;::GetEnumerator()</code>.)</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Serves as a hash function for a particular type. <code>GetHashCode()</code> is suitable for use in hashing.</td>
</tr>
</tbody>
</table>
- **GetHashCode**
  algorithms and data structures like a hash table.
  (Inherited from **Object**.)

- **GetRange**
  Creates a new list that contains a subrange of elements from this list. The current list is unchanged.

- **GetType**
  Gets the **Type** of the current instance.
  (Inherited from **Object**.)

- **IndexOf**
  Overloaded.

  Inserts a new item at the given index in the BigList. All items at indexes equal to or greater than index move up one index.

  (Overrides **ListBase**<Of <T>>...::**Insert**(Int32, **T**).)

- **InsertRange**
  Overloaded.

- **LastIndexOf**
  Overloaded.

- **Print**
  Prints out the internal structure of the tree, for debugging purposes.

  Returns a view onto a sub-range of this list. Items are not copied; the returned **IList<T>** is simply a different view onto the same underlying items. Changes to this list are reflected in the view, and vice versa. Insertions and deletions in the view change the size of the view, but insertions and deletions in the underlying list do not.

  (Overrides **ListBase**<Of <T>>...::**Range**(Int32, Int32).)

- **Remove**
  Overloaded.

  Searches the list for the first item that compares equal to item. If one is found, it is removed. Otherwise, the list is unchanged.

  (Inherited from **ListBase**<Of <T>>.)

- **RemoveAll**
  Removes all the items in the collection that satisfy the condition defined by predicate.

  (Inherited from **CollectionBase**<Of <T>>.)

- **RemoveAt**
  Removes the item at the given index in the BigList. All items at indexes greater than index move down one index.

  (Overrides **ListBase**<Of <T>>...::**RemoveAt**(Int32).)

- **RemoveRange**
  Removes a range of items at the given index in the Deque. All items at indexes greater than index move
down count indices in the Deque.

- **Reverse** Overloaded.
- **Sort** Overloaded.
- **ToArray** Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from `CollectionBase<Of <T>>`.)
- **ToString** Shows the string representation of the collection. The string representation contains a list of the items in the collection. Contained collections (except string) are expanded recursively. (Inherited from `CollectionBase<Of <T>>`.)
- **TrueForAll** Determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from `CollectionBase<Of <T>>`.)
- **TryFind** Finds the first item in the list that satisfies the condition defined by predicate. (Inherited from `ListBase<Of <T>>`.)
- **TryFindLast** Finds the last item in the list that satisfies the condition defined by predicate. (Inherited from `ListBase<Of <T>>`.)
- **Validate** Attempts to validate the internal consistency of the tree.
# Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>)</td>
</tr>
</tbody>
</table>
### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td>Gets the number of items stored in the BigList. The indices of the items range from 0 to Count-1. (Overrides <code>ListBase&lt;Of &lt;T&gt;&gt;::Count</code>.)</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Gets or sets an item in the list, by index. (Overrides <code>ListBase&lt;Of &lt;T&gt;&gt;::Item[[Int32]]</code>.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;Of T&gt;</code>::<code>IsReadOnly</code></td>
<td>(Inherited from <code>ICollection&lt;Of T&gt;</code>) Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>ICollection&lt;Of T&gt;</code>)</td>
</tr>
<tr>
<td><code>ICollection::&lt;&gt;CopyTo</code></td>
<td>Indicates whether the collection is synchronized. (Inherited from <code>ICollection&lt;Of T&gt;</code>)</td>
</tr>
<tr>
<td><code>ICollection::&lt;&gt;IsSynchronized</code></td>
<td>Indicates the synchronization object for this collection. (Inherited from <code>ICollection&lt;Of T&gt;</code>)</td>
</tr>
<tr>
<td><code>ICollection::&lt;&gt;SyncRoot</code></td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from <code>ICollection&lt;Of T&gt;</code>)</td>
</tr>
<tr>
<td><code>IEnumerable::&lt;&gt;GetEnumerator</code></td>
<td>Adds an item to the end of the list. This method is equivalent to calling:</td>
</tr>
<tr>
<td><code>IList::&lt;&gt;Add</code></td>
<td></td>
</tr>
</tbody>
</table>

---

*Copy Code*

`Insert(Count, item)`

(Inherited from `ListBase<Of T>`) Removes all the items from the list,
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| `Clear` | Removes all elements, resulting in an empty list. (Inherited from `ListBase<T>`.)
| `Contains` | Determines if the list contains any item that compares equal to value. (Inherited from `ListBase<T>`.)
| `IndexOf` | Find the first occurrence of an item equal to value in the list, and returns the index of that item. (Inherited from `ListBase<T>`.)
| `Insert` | Insert a new item at the given index. (Inherited from `ListBase<T>`.)
| `IsFixedSize` | Returns whether the list is a fixed size. This implementation always returns false. (Inherited from `ListBase<T>`.)
| `IsReadOnly` | Returns whether the list is read only. This implementation returns the value from `ICollection<T>.IsReadOnly`, which is by default, false. (Inherited from `ListBase<T>`.)
| `Item` | Gets or sets the value at a particular index in the list. (Inherited from `ListBase<T>`.)
| `Remove` | Searches the list for the first item that compares equal to value. If one is found, it is removed. Otherwise, the list is unchanged. (Inherited from `ListBase<T>`.)
| `RemoveAt` | Removes the item at the given index. (Inherited from `ListBase<T>`.)
| `Clone` | Creates a new BigList that is a copy of this list.
See Also

BigList<(Of <T>)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<(Of <T>)> Constructor

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BigList(Of&lt;T&gt;)</code></td>
<td>Creates a new <code>BigList</code>. The <code>BigList</code> is initially empty.</td>
</tr>
<tr>
<td><code>BigList(Of&lt;T&gt;)&lt;New&gt;()</code></td>
<td>Creates a new <code>BigList</code> initialized with the items from collection, in order.</td>
</tr>
<tr>
<td><code>BigList(Of&lt;T&gt;)&lt;New(IEnumerable(Of&lt;T&gt;)&gt;)</code></td>
<td>Creates a new <code>BigList</code> that is a copy of list.</td>
</tr>
<tr>
<td><code>BigList(Of&lt;T&gt;)&lt;New(BigList(Of&lt;T&gt;))&gt;</code></td>
<td>Creates a new <code>BigList</code> initialized with a given number of copies of the items from collection, in order.</td>
</tr>
<tr>
<td><code>BigList(Of&lt;T&gt;)&lt;New(BigList(Of&lt;T&gt;), Int32)&gt;</code></td>
<td>Creates a new <code>BigList</code> that is several copies of list.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
BigList(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>> Constructor

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new BigList. The BigList is initially empty.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public BigList()

Visual Basic (Declaration)

Public Sub New

Visual C++

public:
BigList ()
Remarks

Creating a empty BigList takes constant time and consumes a very small amount of memory.
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of T> Constructor (IEnumerable<Of T>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new BigList initialized with the items from collection, in order.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public BigList(
    IEnumerable<T> collection
)

Visual Basic (Declaration)

Public Sub New (_
    collection As IEnumerable(Of T) _
)

Visual C++

public:
BigList (.
    IEnumerable<T>^ collection
)

Parameters

collection
    IEnumerable<(Of <T>)> The collection used to initialize the BigList.
Remarks

Initializing the tree list with the elements of collection takes time $O(N)$, where $N$ is the number of items in collection.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList(Of T) Constructor (BigList(Of T))

See Also

Visual Basic (Declaration) □ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new BigList that is a copy of list.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public BigList(
    BigList<T> list
)

Visual Basic (Declaration)

Public Sub New (_
    list As BigList(Of T) _
)

Visual C++

public:
BigList (
    BigList<T>^ list
)

Parameters

list
    BigList<(Of <T>)>
    The BigList to copy.
Remarks

Copying a BigList takes constant time, and little additional memory, since the storage for the items of the two lists is shared. However, changing either list will take additional time and memory. Portions of the list are copied when they are changed.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>list is null.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>> Constructor (IEnumerable<Of <T>>, Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new BigList initialized with a given number of copies of the items from collection, in order.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public BigList(
    IEnumerable<T> collection,
    int copies
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub New (_
    collection As IEnumerable(Of T), _
    copies As Integer _
)
```

**Visual C++**

```cpp
public:
BigList (
    IEnumerable<T>^ collection,
    int copies
)
```

### Parameters

collection

- `IEnumerable(Of <T>)`
  - The collection used to initialize the BigList.

copies

- `Int32`
  - Number of copies of the collection to use.
Remarks

Initializing the tree list with the elements of collection takes time $O(N + \log K)$, where $N$ is the number of items in collection, and $K$ is the number of copies.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentOutOfRangeException</td>
<td>copies is negative.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>collection is null.</td>
</tr>
</tbody>
</table>


See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<(Of <T>)> Constructor (BigList<(Of <T>)>, Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new BigList that is several copies of list.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public BigList(
    BigList<T> list,
    int copies
)
```

### Visual Basic (Declaration)

```vbnet
Public Sub New (
    list As BigList(Of T),
    copies As Integer
)
```

### Visual C++

```cpp
public:
BigList (  
    BigList<T>^ list,
    int copies
)
```

## Parameters

- **list**
  - `BigList<(Of <T>)>`
  - The BigList to copy.

- **copies**
  - `Int32`
  - Number of copies of the collection to use.
Remarks

Creating K copies of a BigList takes time O(log K), and O(log K) additional memory, since the storage for the items of the two lists is shared. However, changing either list will take additional time and memory. Portions of the list are copied when they are changed.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentNullException</td>
<td>list is null.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>> Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#  
[SerializableAttribute]  
public class BigList<T>

Visual Basic (Declaration)  
<SerializableAttribute>  
Public Class BigList(Of T)

Visual C++  
[SerializableAttribute]  
generic<typename T>  
public ref class BigList
Type Parameters

T

The type exposes the following methods.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Add**             | Adds an item to the end of the BigList. The indices of all existing items in the Deque are unchanged.  
(Overrides ListBase<Of<T>>:::Add(T).) Concatenates two lists together to create a new list. Both lists being concatenated are unchanged. The resulting list contains all the items in first, followed by all the items in second. |
| **Addition**        |                                                                                                                                                                                                            |
| **AddRange**        | Overloaded.                                                                                                                                                                                                |
| **AddRangeToFront** | Overloaded.                                                                                                                                                                                                |
| **AddToFront**      | Adds an item to the beginning of the BigList. The indices of all existing items in the Deque are increased by one, and the new item has index zero.  
Provides a read-only view of this list. The returned IList<T> provides a view of the list that prevents modifications to the list. Use the method to provide access to the list without allowing changes. Since the returned object is just a view, changes to the list will be reflected in the view.  
(Inherited from ListBase<Of<T>>.) |
| **AsReadOnly**      |                                                                                                                                                                                                            |
| **BinarySearch**    | Overloaded.                                                                                                                                                                                                |
| **Clear**           | Removes all of the items from the BigList.  
(Overrides ListBase<Of<T>>:::Clear().)                                                                                                                                                                        |
| **Clone**           | Creates a new BigList that is a copy of this list.                                                                                                                                                         |
| **CloneContents**   | Makes a deep clone of this BigList. A new BigList is created with a clone of each element of this set, by calling ICloneable.Clone on each element. If T is a value type, then this method is the same as Clone.  
Determines if the list contains any item that compares equal to item. The implementation simply checks whether IndexOf(item) returns a non-negative value.  
(Inherited from ListBase<Of<T>>.) |
| **Contains**        |                                                                                                                                                                                                            |
**ConvertAll**<Of TDest>

Convert the list to a new list by applying a delegate to each item in the collection. The resulting list contains the result of applying converter to each item in the list, in order. The current list is unchanged.

**CopyTo**

Overloaded.

Counts the number of items in the collection that satisfy the condition defined by predicate.

(Inherited from CollectionBase<Of T>.)

**CountWhere**

Determines whether the specified Object is equal to the current Object.

(Inherited from Object.)

**Exists**

Determines if the collection contains any item that satisfies the condition defined by predicate.

(Inherited from CollectionBase<Of T>.)

**Find**

Finds the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned.

(Inherited from ListBase<Of T>.)

**FindAll**

Enumerates the items in the collection that satisfy the condition defined by predicate.

(Inherited from CollectionBase<Of T>.)

**FindIndex**

Overloaded.

Finds the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned.

(Inherited from ListBase<Of T>.)

**FindLastIndex**

Overloaded.

Performs the specified action on each item in this collection.

(Inherited from CollectionBase<Of T>.)

**GetEnumerator**

Enumerates all of the items in the list, in order. The item at index 0 is enumerated first, then the item at index 1, and so on. Usually, the foreach statement is used to call this method implicitly.

(Overrides ListBase<Of T>...::GetEnumerator()().

Serves as a hash function for a particular type. GetHashCode() is suitable for use in hashing
- **GetHashCode** algorithms and data structures like a hash table. (Inherited from **Object**.)

- **GetRange** Creates a new list that contains a subrange of elements from this list. The current list is unchanged.

- **GetType** Gets the **Type** of the current instance. (Inherited from **Object**.)

- **IndexOf** Overloaded.

  Inserts a new item at the given index in the BigList. All items at indexes equal to or greater than index move up one index. (Overrides **ListBase**(Of **T**)>::**Insert**(Int32, **T**).)

- **InsertRange** Overloaded.

- **LastIndexOf** Overloaded.

- **Print** Prints out the internal structure of the tree, for debugging purposes.

  Returns a view onto a sub-range of this list. Items are not copied; the returned **IList<T>** is simply a different view onto the same underlying items. Changes to this list are reflected in the view, and vice versa. Insertions and deletions in the view change the size of the view, but insertions and deletions in the underlying list do not. (Overrides **ListBase**(Of **T**)>::**Range**(Int32, Int32).)

- **Range** Searches the list for the first item that compares equal to item. If one is found, it is removed. Otherwise, the list is unchanged. (Inherited from **ListBase**(Of **T**)::.)

- **Remove** Removes all the items in the collection that satisfy the condition defined by predicate. (Inherited from **CollectionBase**(Of **T**)::.)

- **RemoveAll** Removes the item at the given index in the BigList. All items at indexes greater than index move down one index. (Overrrides **ListBase**(Of **T**)>::**RemoveAt**(Int32).)

- **RemoveAt** Removes a range of items at the given index in the Deque. All items at indexes greater than index move
down count indices in the Deque.

- **Reverse** Overloaded.
- **Sort** Overloaded.
- **ToArray** Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from `CollectionBase<Of <T>>`.)
- **ToString** Shows the string representation of the collection. The string representation contains a list of the items in the collection. Contained collections (except string) are expanded recursively. (Inherited from `CollectionBase<Of <T>>`.)
- **TrueForAll** Determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from `CollectionBase<Of <T>>`.)
- **TryFind** Finds the first item in the list that satisfies the condition defined by predicate. (Inherited from `ListBase<Of <T>>`.)
- **TryFindLast** Finds the last item in the list that satisfies the condition defined by predicate. (Inherited from `ListBase<Of <T>>`.)
- **Validate** Attempts to validate the internal consistency of the tree.
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection::CopyTo</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>CollectionBase&lt;</code>Of <code>&lt;T&gt;</code>&gt;).</td>
</tr>
<tr>
<td>IEnumerable::GetEnumerator</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from <code>CollectionBase&lt;</code>Of <code>&lt;T&gt;</code>&gt;). Adds an item to the end of the list. This method is equivalent to calling:</td>
</tr>
<tr>
<td>IList::Add</td>
<td>Insert(Count, item) (Inherited from <code>ListBase&lt;</code>Of <code>&lt;T&gt;</code>&gt;).</td>
</tr>
<tr>
<td>IList::Clear</td>
<td>Removes all the items from the list, resulting in an empty list. (Inherited from <code>ListBase&lt;</code>Of <code>&lt;T&gt;</code>&gt;).</td>
</tr>
<tr>
<td>IList::Contains</td>
<td>Determines if the list contains any item that compares equal to value. (Inherited from <code>ListBase&lt;</code>Of <code>&lt;T&gt;</code>&gt;).</td>
</tr>
<tr>
<td>IList::IndexOf</td>
<td>Find the first occurrence of an item equal to value in the list, and returns the index of that item. (Inherited from <code>ListBase&lt;</code>Of <code>&lt;T&gt;</code>&gt;).</td>
</tr>
<tr>
<td>IList::Insert</td>
<td>Insert a new item at the given index.</td>
</tr>
</tbody>
</table>
IList::Remove

Searches the list for the first item that compares equal to value. If one is found, it is removed. Otherwise, the list is unchanged.

IList::RemoveAt

Removes the item at the given index.

ICloneable::Clone

Creates a new BigList that is a copy of this list.
See Also

BigList<(Of <T)>)
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>>::Add Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds an item to the end of the BigList. The indices of all existing items in the Deque are unchanged.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public override sealed void Add(
    T item
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Sub Add (_
    item As T _
)
```

**Visual C++**

```cpp
public:
virtual void Add ( 
    T item
) override sealed
```

### Parameters

- **item**
  - **T**
  - The item to add.
Remarks

Adding an item takes, on average, constant time.
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>>...::Addition Operator

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Concatenates two lists together to create a new list. Both lists being concatenated are unchanged. The resulting list contains all the items in first, followed by all the items in second.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public static BigList<T> operator +(  
    BigList<T> first,  
    BigList<T> second
)
```

### Visual Basic (Declaration)

```vbnet
Public Shared Operator +(  
    _  
    first As BigList(Of T),  
    _  
    second As BigList(Of T)  
) As BigList(Of T)
```

### Visual C++

```cpp
public:  
static BigList<T>^ operator + (  
    BigList<T>^ first,  
    BigList<T>^ second
)
```

## Parameters

**first**  
**BigList**<Of <T>>

The first list to concatenate.

**second**  
**BigList**<Of <T>>

The second list to concatenate.
Remarks

This method takes, on average, constant time, regardless of the size of either list. Although conceptually all of the items in both lists are copied, storage is shared until changes are made to the shared sections.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>first or second is null.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of <T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<(Of <T>>...::AddRange Method

See Also  Members

This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AddRange(IEnumerable&lt;'T&gt;)</code></td>
<td>Adds a collection of items to the end of BigList. The indices of all existing items are unchanged. The last item in the added collection becomes the last item in the BigList.</td>
</tr>
<tr>
<td><code>AddRange(BigList&lt;'T&gt;)</code></td>
<td>Adds a BigList of items to the end of BigList. The indices of all existing items are unchanged. The last item in list becomes the last item in this list. The added list list is unchanged.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
BigList(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>>::AddRange Method (IEnumerable<Of <T>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a collection of items to the end of BigList. The indices of all existing items are unchanged. The last item in the added collection becomes the last item in the BigList.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public void AddRange(
    IEnumerable<T> collection
)

Visual Basic (Declaration)

Public Sub AddRange (_
    collection As IEnumerable(Of T) _
)

Visual C++

public:
void AddRange (
    IEnumerable<T>^ collection
)

Parameters

collection
    IEnumerable<(Of <T>)> The collection of items to add.
Remarks

This method takes time $O(M + \log N)$, where $M$ is the number of items in the collection, and $N$ is the size of the BigList.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...::ArgumentNullException</td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

BigList<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList(Of T)>::.AddRange Method (BigList(Of T)>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a BigList of items to the end of BigList. The indices of all existing items are unchanged. The last item in list becomes the last item in this list. The added list list is unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public void AddRange(
    BigList<T> list
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub AddRange ( _
    list As BigList(Of T) _
)
```

**Visual C++**

```cpp
public:
void AddRange ( 
    BigList<T>^ list
)
```

Parameters

```
list
    BigList<(Of <T>)>
    The list of items to add.
```
Remarks

This method takes, on average, constant time, regardless of the size of either list. Although conceptually all of the items in list are copied, storage is shared between the two lists until changes are made to the shared sections.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>list is null.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<(Of <T>)>...::AddRangeToFront Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AddRangeToFirst(IEnumerable&lt;T&gt;...)</code></td>
<td>Adds a collection of items to the front of BigList. The indices of all existing items in the are increased by the number of items in collection. The first item in the added collection becomes the first item in the BigList.</td>
</tr>
<tr>
<td><code>AddRangeToFirst(BigList&lt;(...)&gt;)</code></td>
<td>Adds a BigList of items to the front of BigList. The indices of all existing items are increased by the number of items in list. The first item in list becomes the first item in this list. The added list list is unchanged.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of<T>) Class
BigList(Of<T>) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Adds a collection of items to the front of BigList. The indices of all existing items in the are increased by the number of items in collection. The first item in the added collection becomes the first item in the BigList.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public void AddRangeToFront(
    IEnumerable<T> collection
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub AddRangeToFront ( _
    collection As IEnumerable(Of T) _
)
```

**Visual C++**

```cpp
public:
void AddRangeToFront ( 
    IEnumerable<T>& collection
)
```

**Parameters**

`collection`  
`IEnumerable<T>`  
The collection of items to add.
Remarks

This method takes time $O(M + \log N)$, where $M$ is the number of items in the collection, and $N$ is the size of the BigList.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::ArgumentNullException</td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList(Of T)>:::AddRangeToFront Method (BigList(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a BigList of items to the front of BigList. The indices of all existing items are increased by the number of items in list. The first item in list becomes the first item in this list. The added list list is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public void AddRangeToFront(
    BigList<T> list
)
```

#### Visual Basic (Declaration)

```vbnet
Public Sub AddRangeToFront ( _
    list As BigList(Of T) _
)
```

#### Visual C++

```cpp
public:
void AddRangeToFront ( 
    BigList<T>^ list
)
```

### Parameters

- **list**
  - `BigList<Of <T>>`
  - The list of items to add.
Remarks

This method takes, on average, constant time, regardless of the size of either list. Although conceptually all of the items in list are copied, storage is shared between the two lists until changes are made to the shared sections.
<table>
<thead>
<tr>
<th><strong>Exception</strong></th>
<th><strong>Condition</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>list is null.</td>
</tr>
</tbody>
</table>
See Also

**BigList(Of T) Class**

**Wintellect.PowerCollections Namespace**

Send comments about this topic to Microsoft.
BigList(Of T)>...:::AddToFront Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds an item to the beginning of the BigList. The indices of all existing items in the Deque are increased by one, and the new item has index zero.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public void AddToFront(
    T item
)

Visual Basic (Declaration)

Public Sub AddToFront ( _
    item As T _
)

Visual C++

public:
void AddToFront ( 
    T item
)

Parameters

item
    T
    The item to add.
Adding an item takes, on average, constant time.
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>>...::BinarySearch Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>- BinarySearch(T)</code></td>
<td>Searches a sorted list for an item via binary search. The list must be sorted in the order defined by the default ordering of the item type; otherwise, incorrect results will be returned.</td>
</tr>
<tr>
<td><code>- BinarySearch(T, IComparer&lt;Of &lt;T&gt;&gt;)</code></td>
<td>Searches a sorted list for an item via binary search. The list must be sorted by the ordering defined by the passed IComparer&lt;T&gt; interface; otherwise, incorrect results will be returned.</td>
</tr>
<tr>
<td><code>- BinarySearch(T, Comparison&lt;Of &lt;T&gt;&gt;)</code></td>
<td>Searches a sorted list for an item via binary search. The list must be sorted by the ordering defined by the passed Comparison&lt;T&gt; delegate; otherwise, incorrect results will be returned.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T)  Class
BigList(Of T)  Members
Wintellect.PowerCollections  Namespace

Send comments about this topic to Microsoft.
BigList(Of <T>)::.BinarySearch Method (T)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Searches a sorted list for an item via binary search. The list must be sorted in the order defined by the default ordering of the item type; otherwise, incorrect results will be returned.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public int BinarySearch(T item)
```

**Visual Basic (Declaration)**

```vbnet
Public Function BinarySearch ( _
    item As T _
) As Integer
```

**Visual C++**

```cpp
public:
    int BinarySearch ( T item
    )
```

### Parameters

`item`  
`T`  
The item to search for.

### Return Value

Returns the index of the first occurrence of item in the list. If the item does not occur in the list, the bitwise complement of the first item larger than item in the list is returned. If no item is larger than item, the bitwise complement of Count is returned.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>The type T does not implement either the</td>
</tr>
<tr>
<td></td>
<td>IComparable or IComparable&lt;T&gt; interfaces.</td>
</tr>
</tbody>
</table>
See Also

[BigList](Of [T]) Class
[WinTellecPowerCollections] Namespace

Send comments about this topic to Microsoft.
BigList<(Of <T>))>.::BinarySearch Method (T, IComparer<(Of <T>))>

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Searches a sorted list for an item via binary search. The list must be sorted by the ordering defined by the passed IComparer<T> interface; otherwise, incorrect results will be returned.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public int BinarySearch(
    T item,
    IComparer<T> comparer
)

Visual Basic (Declaration)

Public Function BinarySearch ( _
    item As T, _
    comparer As IComparer(Of T) _
) As Integer

Visual C++

public:
    int BinarySearch ( 
    T item, 
    IComparer<T>^ comparer
    )

Parameters

dependent

item
    T
    The item to search for.

comparer
    IComparer(Of <T>)
    The IComparer<T> interface used to sort the list.

Return Value

Returns the index of the first occurrence of item in the list. If the item does not occur in the list, the bitwise complement of the first item larger than item in the
list is returned. If no item is larger than item, the bitwise complement of Count is returned.
See Also

BigList<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<(Of <T>)>...::BinarySearch Method (T, Comparison<(Of <T>)>)

See Also

- Visual Basic (Declaration) - Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Searches a sorted list for an item via binary search. The list must be sorted by the ordering defined by the passed Comparison<T> delegate; otherwise, incorrect results will be returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public int BinarySearch(
    T item,
    Comparison<T> comparison
)

Visual Basic (Declaration)

Public Function BinarySearch ( _
    item As T, _
    comparison As Comparison(Of T) _
) As Integer

Visual C++

public:
    int BinarySearch ( 
    T item, 
    Comparison<T>^ comparison
    )

Parameters

item
    T
    The item to search for.

comparison
    Comparison<(Of <T>)>
    The comparison delegate used to sort the list.

Return Value

Returns the index of the first occurrence of item in the list. If the item does not occur in the list, the bitwise complement of the first item larger than item in the
list is returned. If no item is larger than item, the bitwise complement of Count is returned.
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>>...::Clear Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all of the items from the BigList.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed void Clear()

Visual Basic (Declaration)

Public Overrides NotOverridable Sub Clear

Visual C++

public:
virtual void Clear () override sealed
Remarks

Clearing a BigList takes constant time.
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<(Of <T>)>...::Clone Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new BigList that is a copy of this list.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public BigList<T> Clone()

Visual Basic (Declaration)

Public Function Clone As BigList(Of T)

Visual C++

public:
BigList<T>^ Clone ()

Return Value

A copy of the current list
Remarks

Copying a BigList takes constant time, and little additional memory, since the storage for the items of the two lists is shared. However, changing either list will take additional time and memory. Portions of the list are copied when they are changed.
See Also

BigList<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList(Of T)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a deep clone of this BigList. A new BigList is created with a clone of each element of this set, by calling ICloneable.Clone on each element. If T is a value type, then this method is the same as Clone.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```
public BigList<T> CloneContents()
```

**Visual Basic (Declaration)**

```
Public Function CloneContents As BigList(Of T)
```

**Visual C++**

```
public: BigList<T>^ CloneContents ()
```

**Return Value**

The cloned set.
Remarks

If T is a reference type, it must implement ICloneable. Otherwise, an InvalidOperationException is thrown.

If T is a reference type, cloning the list takes time approximate O(N), where N is the number of items in the list.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::InvalidOperationException</td>
<td>T is a reference type that does not implement ICloneable.</td>
</tr>
</tbody>
</table>
See Also

**BigList(Of T)>** Class
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
BigList(Of T)>...::ConvertAll(Of TDest)> Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Convert the list to a new list by applying a delegate to each item in the collection. The resulting list contains the result of applying converter to each item in the list, in order. The current list is unchanged.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public BigList<TDest> ConvertAll<TDest>(
    Converter<T, TDest> converter
)
```

Visual Basic (Declaration)

```vbnet
Public Function ConvertAll(Of TDest) ( _
    converter As Converter(Of T, TDest) _
) As BigList(Of TDest)
```

Visual C++

```cpp
public:
    template<typename TDest>
    BigList<TDest>^ ConvertAll ( 
        Converter<T, TDest>^ converter
    )
```

Parameters

converter

```cpp
Converter<(Of <T, TDest>)>
```
A delegate to the method to call, passing each item in .

Return Value

The resulting BigList from applying converter to each item in this list.
Type Parameters

TDest

The type each item is being converted to.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...::ArgumentNullException</td>
<td>converter is null.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<(Of <T>)>...::CopyTo Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `CopyTo<T>[]()` | Copies all the items in the list, in order, to array, starting at index 0.  
(Inherited from `ListBase<Of<T>>`.) |
| `CopyTo<T>[](), Int32` | Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.  
(Inherited from `CollectionBase<Of<T>>`.) |
| `CopyTo(Int32, array<T>[](), Int32, Int32)` | Copies a range of elements from the list to array, starting at arrayIndex.  
(Inherited from `ListBase<Of<T>>`.) |
See Also

BigList(Of T) Class
BigList(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<(Of <T>)>...:::FindIndex Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FindIndex(Predicate&lt;Of &lt;T&gt;&gt;)</code></td>
<td>Finds the index of the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned. (Inherited from <code>ListBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>FindIndex(Int32, Predicate&lt;Of &lt;T&gt;&gt;)</code></td>
<td>Finds the index of the first item, in the range of items extending from index to the end, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned. (Inherited from <code>ListBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>FindIndex(Int32, Int32, Predicate&lt;Of &lt;T&gt;&gt;)</code></td>
<td>Finds the index of the first item, in the range of count items starting from index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned. (Inherited from <code>ListBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
BigList(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList(Of T)>...::FindLastIndex Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FindLastIndex(Predicate&lt;Of T&gt;)</td>
<td>Finds the index of the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned. (Inherited from ListBase&lt;Of T&gt;.)</td>
</tr>
<tr>
<td>FindLastIndex(Int32, Predicate&lt;Of T&gt;)</td>
<td>Finds the index of the last item, in the range of items extending from the beginning of the list to index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned. (Inherited from ListBase&lt;Of T&gt;.)</td>
</tr>
<tr>
<td>FindLastIndex(Int32, Int32, Predicate&lt;Of T&gt;)</td>
<td>Finds the index of the last item, in the range of count items ending at index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned. (Inherited from ListBase&lt;Of T&gt;.)</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
BigList(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>>.::GetEnumerator Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerates all of the items in the list, in order. The item at index 0 is enumerated first, then the item at index 1, and so on. Usually, the foreach statement is used to call this method implicitly.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed IEnumerator<T> GetEnumerator()

Visual Basic (Declaration)

Public Overrides NotOverridable Function GetEnumerator As IEnumerator

Visual C++

public:
virtual IEnumerator<T>^ GetEnumerator() override sealed

Return Value

An IEnumerator<T> that enumerates all the items in the list.
Remarks

Enumerating all of the items in the list take time $O(N)$, where $N$ is the number of items in the list. Using GetEnumerator() or foreach is much more efficient than accessing all items by index.
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<(Of <T>)>::GetRange Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new list that contains a subrange of elements from this list. The current list is unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public BigList<T> GetRange(int index, int count)
```

Visual Basic (Declaration)

```visualbasic
Public Function GetRange(_
    index As Integer, _
    count As Integer _) As BigList(Of T)
```

Visual C++

```cpp
public:
    BigList<T>^ GetRange ( int index, int count)
```

Parameters

index

- **Int32**
  - The starting index of the sub-range.

count

- **Int32**
  - The number of items in the sub-range. If this is zero, the returned list is empty.

Return Value

A new list with the count items that start at index.
Remarks

This method takes $O(\log N)$, where $N$ is the size of the current list. Although the sub-range is conceptually copied, storage is shared between the two lists until a change is made to the shared items.
Remarks

If a view of a sub-range is desired, instead of a copy, use the more efficient `Range(Int32, Int32)` method, which provides a view onto a sub-range of items.
See Also

BigList<(Of <T>)) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList(Of T)>...::IndexOf Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IndexOf(T)</code></td>
<td>Finds the index of the first item in the list that is equal to item. (Inherited from <code>ListBase&lt;Of&lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IndexOf(T, Int32)</code></td>
<td>Finds the index of the first item, in the range of items extending from index to the end, that is equal to item. (Inherited from <code>ListBase&lt;Of&lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IndexOf(T, Int32, Int32)</code></td>
<td>Finds the index of the first item, in the range of count items starting from index, that is equal to item. (Inherited from <code>ListBase&lt;Of&lt;T&gt;&gt;</code>.)</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T)> Class
BigList(Of T)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>>...::Insert Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Inserts a new item at the given index in the BigList. All items at indexes equal to or greater than index move up one index.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed void Insert(
    int index,
    T item
)

Visual Basic (Declaration)

Public Overrides NotOverridable Sub Insert ( _
    index As Integer, _
    item As T _
)

Visual C++

public:
    virtual void Insert ( 
        int index, 
        T item 
    ) override sealed

Parameters

index

Int32
    The index to insert the item at. After the insertion, the inserted item is located at this index. The first item has index 0.

item

T
    The item to insert at the given index.
Remarks

The amount of time to insert an item is $O(\log N)$, no matter where in the list the insertion occurs. Inserting an item at the beginning or end of the list is $O(N)$. 
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::ArgumentOutOfRangeException</code></td>
<td>index is less than zero or greater than Count.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of<T>>...::InsertRange Method
See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InsertRange(Int32, IEnumerable&lt;Of &lt;T&gt;&gt;)</code></td>
<td>Inserts a collection of items at the given index in the BigList. All items at indexes equal to or greater than index increase their indices by the number of items inserted.</td>
</tr>
<tr>
<td><code>InsertRange(Int32, BigList&lt;Of &lt;T&gt;&gt;)</code></td>
<td>Inserts a BigList of items at the given index in the BigList. All items at indexes equal to or greater than index increase their indices by the number of items inserted.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
BigList(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Inserts a collection of items at the given index in the BigList. All items at indexes equal to or greater than index increase their indices by the number of items inserted.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

**C#**

```csharp
public void InsertRange(
    int index,
    IEnumerable<T> collection
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub InsertRange (_
    index As Integer, _
    collection As IEnumerable(Of T) _
)
```

**Visual C++**

```cpp
public:
void InsertRange ( _
    int index,
    IEnumerable<T>^ collection
)
```

## Parameters

**index**

*Int32*

The index to insert the collection at. After the insertion, the first item of the inserted collection is located at this index. The first item has index 0.

**collection**

*IEnumerable<(Of <T>)>*

The collection of items to insert at the given index.
Remarks

The amount of time to insert an arbitrary collection in the BigList is $O(M + \log N)$, where $M$ is the number of items inserted, and $N$ is the number of items in the list.
- **Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentOutOfRangeException</code></td>
<td>index is less than zero or greater than Count.</td>
</tr>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Inserts a BigList of items at the given index in the BigList. All items at indexes equal to or greater than index increase their indices by the number of items inserted.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public void InsertRange(
    int index,
    BigList<T> list
)
```

Visual Basic (Declaration)

```vbnet
Public Sub InsertRange ( _
    index As Integer, _
    list As BigList(Of T) _
)
```

Visual C++

```cpp
public:
void InsertRange ( 
    int index,
    BigList<T>^ list
)
```

Parameters

index

`Int32`

The index to insert the collection at. After the insertion, the first item of the inserted collection is located at this index. The first item has index 0.

list

`BigList<(Of <T>)>`

The BigList of items to insert at the given index.
Remarks

The amount of time to insert another BigList is $O(\log N)$, where $N$ is the number of items in the list, regardless of the number of items in the inserted list. Storage is shared between the two lists until one of them is changed.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentOutOfRangeException</code></td>
<td>index is less than zero or greater than Count.</td>
</tr>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>list is null.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<(Of <T>)>...::LastIndexOf Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LastIndexOf(T)</code></td>
<td>Finds the index of the last item in the list that is equal to item.</td>
</tr>
<tr>
<td>(Inherited from <code>ListBase&lt;Of&lt;T&gt;&gt;</code>)</td>
<td></td>
</tr>
<tr>
<td><code>LastIndexOf(T, Int32)</code></td>
<td>Finds the index of the last item, in the range of items extending from the beginning of the list to index, that is equal to item.</td>
</tr>
<tr>
<td>(Inherited from <code>ListBase&lt;Of&lt;T&gt;&gt;</code>)</td>
<td></td>
</tr>
<tr>
<td><code>LastIndexOf(T, Int32, Int32)</code></td>
<td>Finds the index of the last item, in the range of count items ending at index, that is equal to item.</td>
</tr>
<tr>
<td>(Inherited from <code>ListBase&lt;Of&lt;T&gt;&gt;</code>)</td>
<td></td>
</tr>
</tbody>
</table>
See Also

`BigList(Of T)` Class
`BigList(Of T)` Members
`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
Prints out the internal structure of the tree, for debugging purposes.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public void Print()

Visual Basic (Declaration)

Public Sub Print

Visual C++

public:
void Print ()
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Returns a view onto a sub-range of this list. Items are not copied; the returned IList<T> is simply a different view onto the same underlying items. Changes to this list are reflected in the view, and vice versa. Insertions and deletions in the view change the size of the view, but insertions and deletions in the underlying list do not.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public override sealed IEnumerable<T> Range(
    int index,
    int count
)
```

Visual Basic (Declaration)

```vbnet
Public Overrides NotOverridable Function Range ( _
    index As Integer, _
    count As Integer _
) As IEnumerable(Of T)
```

Visual C++

```cpp
public:
virtual IEnumerable<T>^ Range (
    int index,
    int count
) override sealed
```

Parameters

index
  Int32
  The starting index of the view.

count
  Int32
  The number of items in the view.

Return Value

A list that is a view onto the given sub-list.
Remarks

If a copy of the sub-range is desired, use the `GetRange(Int32, Int32)` method instead.

This method can be used to apply an algorithm to a portion of a list. For example:

```csharp
Copy Code

Algorithms.ReverseInPlace(list.Range(3, 6))
```

will reverse the 6 items beginning at index 3.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>index or count is negative.</td>
</tr>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>index + count is greater than the size of this list.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
RemoveAt Method

Removes the item at the given index in the BigList. All items at indexes greater than index move down one index.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed void RemoveAt(
    int index
)

Visual Basic (Declaration)

Public Overrides NotOverridable Sub RemoveAt ( _
    index As Integer _
)

Visual C++

public:
virtual void RemoveAt (  
    int index
) override sealed

Parameters

index

Int32
The index in the list to remove the item at. The first item in the list has index 0.
Remarks

The amount of time to delete an item in the BigList is $O(\log N)$, where $N$ is the number of items in the list.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>index is less than zero or greater than or equal to Count.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T> >::<RemoveRange Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes a range of items at the given index in the Deque. All items at indexes greater than index move down count indices in the Deque.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public void RemoveRange(
    int index,
    int count
)
```

### Visual Basic (Declaration)

```vbnet
Public Sub RemoveRange (_
    index As Integer, _
    count As Integer _
)
```

### Visual C++

```cpp
public:
void RemoveRange ( 
    int index,
    int count
)
```

## Parameters

### index

- **Int32**
  - The index in the list to remove the range at. The first item in the list has index 0.

### count

- **Int32**
  - The number of items to remove.
Remarks

The amount of time to delete count items in the Deque is proportional to the distance of index from the closest end of the Deque, plus count: $O(\text{count} + \text{Min(index, Count - 1 - index)})$. 
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>index is less than zero or greater than or equal to Count, or count is less than zero or too large.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>>...::Reverse Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reverse()()</td>
<td>Reverses the current list in place.</td>
</tr>
<tr>
<td>Reverse(Int32, Int32)</td>
<td>Reverses the items in the range of count items starting from start, in place.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
BigList(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>>...::Reverse Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Reverses the current list in place.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#
public void Reverse()

Visual Basic (Declaration)
Public Sub Reverse

Visual C++
public:
 void Reverse ()
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList(Of T)

Reverse Method (Int32, Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Reverses the items in the range of count items starting from start, in place.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public void Reverse(
    int start,
    int count
)

Visual Basic (Declaration)

Public Sub Reverse ( _
    start As Integer, _
    count As Integer _
)

Visual C++

public:
void Reverse (_
    int start,
    int count _
)

Parameters

start
   Int32
   The starting index of the range to reverse.

count
   Int32
   The number of items in range to reverse.
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<(Of <T>)>::Sort Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="SortMethod" alt="Sort()" /></td>
<td>Sorts the list in place.</td>
</tr>
<tr>
<td><img src="ComparerMethod" alt="Sort(IComparer&lt;T&gt;)" /></td>
<td>Sorts the list in place. A supplied IComparer&lt;T&gt; is used to compare the items in the list.</td>
</tr>
<tr>
<td><img src="ComparisonMethod" alt="Sort(Comparison&lt;T&gt;)" /></td>
<td>Sorts the list in place. A supplied Comparison&lt;T&gt; delegate is used to compare the items in the list.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
BigList(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList(Of <T>):::Sort Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Sorts the list in place.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public void Sort()

Visual Basic (Declaration)

Public Sub Sort

Visual C++

public:
void Sort ()
Remarks

The Quicksort algorithm is used to sort the items. In virtually all cases, this takes time $O(N \log N)$, where $N$ is the number of items in the list.

Values are compared by using the IComparable or IComparable$\langle T \rangle$ interface implementation on the type $T$. 
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::InvalidOperationException</td>
<td>The type T does not implement either the IComparable or IComparable&lt;T&gt; interfaces.</td>
</tr>
</tbody>
</table>
See Also

BigList<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList(Of T)>...::Sort Method (IComparer(Of T)>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Sorts the list in place. A supplied IComparer<T> is used to compare the items in the list.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public void Sort(
    IComparer<T> comparer
)
```

Visual Basic (Declaration)

```vbnet
Public Sub Sort ( _
    comparer As IComparer(Of T) _
)
```

Visual C++

```cpp
public:
void Sort ( 
    IComparer<T>^ comparer
)
```

Parameters

comparer

```csharp
IComparer(Of T)->
```
The comparer instance used to compare items in the collection. Only the Compare method is used.
Remarks

The Quicksort algorithms is used to sort the items. In virtually all cases, this takes time $O(N \log N)$, where $N$ is the number of items in the list.
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList(Of T)>...::Sort Method (Comparison(Of T)>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Sorts the list in place. A supplied Comparison<T> delegate is used to compare the items in the list.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public void Sort(
    Comparison<T> comparison
)

Visual Basic (Declaration)

Public Sub Sort (_
    comparison As Comparison(Of T) _
)

Visual C++

public:
void Sort ( 
    Comparison<T>^ comparison
)

Parameters

comparison
    Comparison(Of T>)
    The comparison delegate used to compare items in the collection.
Remarks

The Quicksort algorithms is used to sort the items. In virtually all cases, this takes time $O(N \log N)$, where $N$ is the number of items in the list.
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList(Of <T>)::.::System.ICloneable.Clone Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new BigList that is a copy of this list.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

private `Object` `ICloneable.Clone()`

#### Visual Basic (Declaration)

Private Function System.ICloneable.Clone As `Object` Implements `ICloneable`

#### Visual C++

private:
virtual `Object^` System.ICloneable.Clone () sealed = `ICloneable::Clone`

### Return Value

A copy of the current list
Remarks

Copying a BigList takes constant time, and little additional memory, since the storage for the items of the two lists is shared. However, changing either list will take additional time and memory. Portions of the list are copied when they are changed.
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<(Of <T>)>...:::Validate Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Attempts to validate the internal consistency of the tree.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public void Validate()
```

**Visual Basic (Declaration)**

```vbnet
Public Sub Validate
```

**Visual C++**

```cpp
public:
void Validate ()
```
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList(Of <T>)> Properties

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class BigList<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class BigList(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class BigList
Type Parameters

T

The type exposes the following properties.
### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| ![Count] | Gets the number of items stored in the BigList. The indices of the items range from 0 to Count-1.  
(Overrrides `ListBase<Of<T>>::Count`) |
| ![Item] | Gets or sets an item in the list, by index.  
(Overrrides `ListBase<Of<T>>::Item[Int32]`) |
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;Of &lt;T&gt;&gt;::IsReadOnly</code></td>
<td>Indicates whether the collection is</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection::IsSynchronized</code></td>
<td>Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection::SyncRoot</code></td>
<td>Returns whether the list is a fixed size. This implementation always returns false.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>ListBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IList::IsFixedSize</code></td>
<td>Returns whether the list is read only. This implementation returns the value from ICollection&lt;SomeType&gt;.IsReadOnly, which is by default, false.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>ListBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IList::IsReadOnly</code></td>
<td>Gets or sets the value at a particular index in the list.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>ListBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
</tbody>
</table>
See Also

BigList<(Of <T>)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>>...::Count Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets the number of items stored in the BigList. The indices of the items range from 0 to Count-1.

Namespace: Wintellect.PowerCollections
Assembly:  PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed int Count { get; }

Visual Basic (Declaration)

Public Overrides NotOverridable ReadOnly Property Count As Integer

Visual C++

public:
virtual property int Count {
    int get () override sealed;
}

Field Value

The number of items in the BigList.
Remarks

Getting the number of items in the BigList takes constant time.
- See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
BigList<Of <T>>::Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets or sets an item in the list, by index.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public override sealed T this[
                   int index
    ]{ get; set;}
```

#### Visual Basic (Declaration)

```vbnet
Public Overrides NotOverridable Default Property Item ( _
                   index As Integer _
               ) As T
```

#### Visual C++

```cpp
public:
virtual property T default[int index] { 
        T get (int index) override sealed;
        void set (int index, T value) override sealed;
    }
```

### Parameters

#### index

- **Int32**
  - The index of the item to get or set. The first item in the list has index 0, the last item has index Count-1.

### Return Value

- The value of the item at the given index.
Remarks

Getting or setting an item takes time $O(\log N)$, where $N$ is the number of items in the list.

To process each of the items in the list, using GetEnumerator() or a foreach loop is more efficient than accessing each of the elements by index.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>index is less than zero or greater than or equal to Count.</td>
</tr>
</tbody>
</table>
See Also

BigList(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
The `BinaryPredicate<Of <T>>` delegate type encapsulates a method that takes two items of the same type, and returns a boolean value representing some relationship between them. For example, checking whether two items are equal or equivalent is one kind of binary predicate.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public delegate bool BinaryPredicate<T>(
    T item1,
    T item2
)
```

### Visual Basic (Declaration)

```vbnet
Public Delegate Function BinaryPredicate(Of T) ( _
    item1 As T, _
    item2 As T _
) As Boolean
```

### Visual C++

```cpp
generic<typename T>
public delegate bool BinaryPredicate (  
    T item1,
    T item2
)
```

### Parameters

- **item1**
  - `T`
  - The first item.

- **item2**
  - `T`
  - The second item.

### Return Value

Whether `item1` and `item2` satisfy the relationship that the `BinaryPredicate` defines.
Type Parameters

T
See Also

Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase<Of <T>> Class

See Also  Members

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

CollectionBase is a base class that can be used to more easily implement the generic ICollection<T> and non-generic ICollection interfaces.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
[SerializableAttribute]
public abstract class CollectionBase<T> : ICollection<T>, IEnumerable<T>, ICollection, IEnumerable
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute> _
Public MustInherit Class CollectionBase(Of T) _
    Implements ICollection(Of T), IEnumerable(Of T), _
    ICollection, IEnumerable
```

**Visual C++**

```cpp
[SerializableAttribute]
generic<typename T>
public ref class CollectionBase abstract : ICollection<T>, IEnumerable<T>, ICollection, IEnumerable
```
**Type Parameters**

T

The item type of the collection.
Remarks

To use CollectionBase as a base class, the derived class must override the Count, GetEnumerator, Add, Clear, and Remove methods.

ICollection&lt;T&gt;.Contains need not be implemented by the derived class, but it should be strongly considered, because the CollectionBase implementation may not be very efficient.
**Inheritance Hierarchy**

- `System::::Object`
- `Wintellect.PowerCollections::::CollectionBase<Of <T>>`
  - `Wintellect.PowerCollections::::MultiDictionaryBase<Of <TKey, TValue>>`
  - `Wintellect.PowerCollections::::ListBase<Of <T>>`
  - `Wintellect.PowerCollections::::DictionaryBase<Of <TKey, TValue>>`
  - `Wintellect.PowerCollections::::OrderedBag<Of <T>>`
  - `Wintellect.PowerCollections::::OrderedBag<Of <T>>::View`
  - `Wintellect.PowerCollections::::Bag<Of <T>>`
  - `Wintellect.PowerCollections::::OrderedSet<Of <T>>`
  - `Wintellect.PowerCollections::::OrderedSet<Of <T>>::View`
  - `Wintellect.PowerCollections::::Set<Of <T>>`
See Also

CollectionBase<Of T> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase(Of T) Members

See Also  Methods  Constructors  Properties  Explicit Interface Implementations

- [ ] Include Inherited Members
- [ ] Include Protected Members
- [ ] .NET Compact Framework Members Only
- [ ] XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

```csharp
[SerializableAttribute]
public abstract class CollectionBase<T>
```

Visual Basic (Declaration)

```vbnet
<SerializableAttribute>
Public MustInherit Class CollectionBase(Of T)
```

Visual C++

```cpp
[SerializableAttribute]
generic<typename T>
public ref class CollectionBase abstract
```
Type Parameters

T

The type exposes the following members.
## Protected Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CollectionBase&lt;Of T&gt;CollectionBase&lt;Of T&gt;</code></td>
<td>New</td>
</tr>
</tbody>
</table>
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Add</code></td>
<td>Must be overridden to allow adding items to this collection. Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view.</td>
</tr>
<tr>
<td><code>AsReadOnly</code></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view.</td>
</tr>
<tr>
<td><code>Clear</code></td>
<td>Must be overridden to allow clearing this collection.</td>
</tr>
<tr>
<td><code>Contains</code></td>
<td>Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable&lt;T&gt;.Equals or Object.Equals.</td>
</tr>
<tr>
<td><code>ConvertAll(Of&lt;TOutput&gt;)</code></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.</td>
</tr>
<tr>
<td><code>CopyTo</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.</td>
</tr>
<tr>
<td><code>CountWhere</code></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><code>Exists</code></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate.</td>
</tr>
<tr>
<td><code>FindAll</code></td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td><code>ForEach</code></td>
<td>Performs the specified action on each item in this collection.</td>
</tr>
<tr>
<td><code>GetEnumerator</code></td>
<td>Must be overridden to enumerate all the members of the</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td></td>
<td><strong>GetHashCode()</strong> is suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Remove</td>
<td>Must be overridden to allow removing items from this collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td>ToArray</td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.</td>
</tr>
<tr>
<td></td>
<td>Shows the string representation of the collection. The string representation contains a list of the items in the collection.</td>
</tr>
<tr>
<td>ToString</td>
<td>Contained collections (except string) are expanded recursively.</td>
</tr>
<tr>
<td></td>
<td>(Overrides <strong>Object..::ToString()</strong>.)</td>
</tr>
<tr>
<td>TrueForAll</td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate.</td>
</tr>
</tbody>
</table>
### Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{Finalize}</code></td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><code>{MemberwiseClone}</code></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Must be overridden to provide the number of items in the collection.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;Of </code>T&gt;<code>&gt;::IsReadOnly</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.</td>
</tr>
<tr>
<td><code>ICollection::CopyTo</code></td>
<td>Indicates whether the collection is synchronized. Indicates the synchronization object for this collection. Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection.</td>
</tr>
<tr>
<td><code>ICollection::IsSynchronized</code></td>
<td></td>
</tr>
<tr>
<td><code>ICollection::SyncRoot</code></td>
<td></td>
</tr>
<tr>
<td><code>IEnumerable::GetEnumerator</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

CollectionBase<(Of <T>)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase(Of T) Constructor

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

protected CollectionBase()

Visual Basic (Declaration)

Protected Sub New

Visual C++

protected:
CollectionBase ()
See Also

CollectionBase<Of T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase(Of T) Methods

See Also

☐ □

☐ Include Inherited Members☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#  
[SerializableAttribute]
public abstract class CollectionBase<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public MustInherit Class CollectionBase(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class CollectionBase abstract
Type Parameters

T

The type exposes the following methods.
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Must be overridden to allow adding items to this collection. Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view.</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Must be overridden to allow clearing this collection. Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable&lt;T&gt;.Equals or Object.Equals.</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable&lt;T&gt;.Equals or Object.Equals.</td>
</tr>
<tr>
<td><strong>ConvertAll(Of &lt;TOutput&gt;)</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.</td>
</tr>
<tr>
<td><strong>CountWhere</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate. Enumerates the items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Performs the specified action on each item in this collection. Must be overridden to enumerate all the members of the</td>
</tr>
<tr>
<td><strong>FindAll</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ForEach</strong></td>
<td></td>
</tr>
<tr>
<td><strong>GetEnumerator</strong></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. GetHashCode() is suitable for use in hashing algorithms and data structures like a hash table. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Remove</td>
<td>Must be overridden to allow removing items from this collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td>ToArray</td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. Shows the string representation of the collection. The string representation contains a list of the items in the collection.</td>
</tr>
<tr>
<td>ToString</td>
<td>Shows the string representation of the array. The string representation contains a list of the items in the array. Contained collections (except string) are expanded recursively. (Overrides Object::ToString().)</td>
</tr>
<tr>
<td>TrueForAll</td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate.</td>
</tr>
</tbody>
</table>
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
### Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection</code> :: <code>CopyTo</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from <code>GetEnumerator</code> to get all the items and copy them to the provided array. Provides an <code>IEnumerator</code> that can be used to iterate all the members of the collection. This implementation uses the <code>IEnumerator&lt;T&gt;</code> that was overridden by the derived classes to enumerate the members of the collection.</td>
</tr>
<tr>
<td><code>IEnumerable</code> :: <code>GetEnumerator</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

CollectionBase<(Of <T>)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase<Of <T>>...::Add Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Must be overridden to allow adding items to this collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual void Add(
    T item
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Sub Add ( _
    item As T _
)
```

**Visual C++**

```cpp
public:
virtual void Add ( 
    T item
)
```

**Parameters**

`item`  
`T`  
Item to be added to the collection.
Remarks

This method is not abstract, although derived classes should always override it. It is not abstract because some derived classes may wish to reimplement `Add` with a different return type (typically bool). In C#, this can be accomplished with code like the following:

```csharp
public class MyCollection<T> : CollectionBase<T>, ICollection<T>
{
    public new bool Add(T item)
    {
        /* Add the item */
    }

    void ICollection<T>.Add(T item) {
        Add(item);
    }
}
```
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::NotImplementedException</code></td>
<td>Always throws this exception to indicated that the method must be overridden or re-implemented in the derived class.</td>
</tr>
</tbody>
</table>
See Also

CollectionBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Provides a read-only view of this collection. The returned ICollection<T> provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual ICollection<T> AsReadOnly()
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function AsReadOnly As ICollection(Of T)
```

**Visual C++**

```cpp
public:
virtual ICollection<T>^ AsReadOnly ()
```

**Return Value**

An ICollection<T> that provides read-only access to the collection.
See Also

CollectionBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase<Of <T>>...::Clear Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Must be overridden to allow clearing this collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public abstract void Clear()
```

**Visual Basic (Declaration)**

```vbnet
Public MustOverride Sub Clear
```

**Visual C++**

```c++
public:
virtual void Clear () abstract
```
See Also

CollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable<T>.Equals or Object.Equals.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual bool Contains(
    T item
)

Visual Basic (Declaration)

Public Overridable Function Contains (_
    item As T _
) As Boolean

Visual C++

public:
    virtual bool Contains ( 
    T item
    )

Parameters

item
    T
    The item to check for in the collection.

Return Value

True if the collection contains item, false otherwise.
Remarks

You should strongly consider overriding this method to provide a more efficient implementation, or if the default equality comparison is inappropriate.
See Also

`CollectionBase<Of<T>>` Class

`Wintellect.PowerCollections` Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public virtual IEnumerable<TOutput> ConvertAll<TOutput>(
    Converter<T, TOutput> converter
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function ConvertAll(Of TOutput) ( _
    converter As Converter(Of T, TOutput) _
) As IEnumerable(Of TOutput)
```

**Visual C++**

```csharp
public:
    template<typename TOutput>
    virtual IEnumerable<TOutput>^ ConvertAll ( 
        Converter<T, TOutput>^ converter
    )
```

### Parameters

**converter**

- `Converter<(Of <T, TOutput>)>`
  - A delegate to the method to call, passing each item in this collection.

### Return Value

- An `IEnumerable<TOutput>` that enumerates the resulting collection from applying `converter` to each item in this collection in order.
Type Parameters

TOutput

The type each item is being converted to.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>converter is null.</td>
</tr>
</tbody>
</table>
See Also

CollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase(Of T)::.CopyTo Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public virtual void CopyTo(
    T[] array,
    int arrayIndex
)
```

#### Visual Basic (Declaration)

```vbnet
Public Overridable Sub CopyTo ( _
    array As T(), _
    arrayIndex As Integer _
)
```

#### Visual C++

```cpp
public:
virtual void CopyTo ( 
    array<T>^ array,
    int arrayIndex
)
```

### Parameters

- **array**
  - `array<T>[]()`
  - Array to copy to.

- **arrayIndex**
  - `Int32`
  - Starting index in array to copy to.
See Also

**CollectionBase(Of T)** Class
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
Counts the number of items in the collection that satisfy the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public virtual int CountWhere(
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function CountWhere ( _
    predicate As Predicate(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:
virtual int CountWhere (
    Predicate<T>^ predicate
)
```

**Parameters**

`predicate`
- `Predicate<(Of <T>)>`
  - A delegate that defines the condition to check for.

**Return Value**

The number of items in the collection that satisfy predicate.
See Also

CollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase(Of &lt;T&gt;).::Exists Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if the collection contains any item that satisfies the condition defined by predicate.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
- Syntax

**C#**

```csharp
public virtual bool Exists(
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function Exists ( _
predicate As Predicate(Of T) _
) As Boolean
```

**Visual C++**

```csharp
public:
    virtual bool Exists (_
        Predicate<T>^ predicate
    )
```

**Parameters**

predicate

- **Predicate<(Of <T>)>**
  A delegate that defines the condition to check for.

**Return Value**

True if the collection contains one or more items that satisfy the condition defined by predicate. False if the collection does not contain an item that satisfies predicate.
See Also

CollectionBase<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Enumerates the items in the collection that satisfy the condition defined by predicate.

**Namespace**: Wintellect.PowerCollections  
**Assembly**: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public virtual IEnumerable<T> FindAll(
    Predicate<T> predicate
)
```

Visual Basic (Declaration)

```vbnet
Public Overridable Function FindAll ( _
    predicate As Predicate(Of T) _
) As IEnumerable(Of T)
```

Visual C++

```csharp
public:
    virtual IEnumerable<T>^ FindAll ( 
    Predicate<T>^ predicate 
)
```

Parameters

predicate

`Predicate(Of <T>)`
A delegate that defines the condition to check for.

Return Value

An IEnumerable<T> that enumerates the items that satisfy the condition.
See Also

**CollectionBase(Of T)** Class

**Wintellect.PowerCollections** Namespace

Send comments about this topic to Microsoft.
CollectionBase(Of T)::.ForEach Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Performs the specified action on each item in this collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual void ForEach(
    Action<T> action
)

Visual Basic (Declaration)

Public Overridable Sub ForEach ( _
    action As Action(Of T) _
)

Visual C++

public:
    virtual void ForEach (
    Action<T>^ action
)

Parameters

action

    Action(Of <T>)>
    An Action delegate which is invoked for each item in this collection.
See Also

**CollectionBase(Of T) Class**  
**Wintellect.PowerCollections Namespace**

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
CollectionBase(Of T)>..::.GetEnumerator Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Must be overridden to enumerate all the members of the collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public abstract IEnumerator<T> GetEnumerator()

Visual Basic (Declaration)

Public MustOverride Function GetEnumerator As IEnumerator(Of T)

Visual C++

public:
  virtual IEnumerator<T>^ GetEnumerator () abstract

Return Value

A generic IEnumerator<T> that can be used to enumerate all the items in the collection.
See Also

CollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase(Of T)::.Remove Method

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Must be overridden to allow removing items from this collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

public abstract bool Remove(
    T item
)

**Visual Basic (Declaration)**

Public MustOverride Function Remove ( _
    item As T _
) As Boolean

**Visual C++**

public:
    virtual bool Remove ( 
        T item
    ) abstract

### Parameters

item
    T

### Return Value

True if item existed in the collection and was removed. False if item did not exist in the collection.
See Also

CollectionBase<({T}> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase<Of <T>>::.RemoveAll Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all the items in the collection that satisfy the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public virtual ICollection<T> RemoveAll(
    Predicate<T> predicate
)
```

#### Visual Basic (Declaration)

```vbnet
Public Overridable Function RemoveAll ( _
    predicate As Predicate(Of T) _
) As ICollection(Of T)
```

#### Visual C++

```cpp
public:
    virtual ICollection<T>^ RemoveAll ( 
        Predicate<T>^ predicate
    )
```

### Parameters

- **predicate**
  - `Predicate(Of T)`
  - A delegate that defines the condition to check for.

### Return Value

- Returns a collection of the items that were removed, in sorted order.
See Also

CollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase(Of T)::.System.Collections.ICollection.CopyTo Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private void ICollection.CopyTo(
    Array array,
    int index
)

Visual Basic (Declaration)

Private Sub System.Collections.ICollection.CopyTo ( _
    array As Array, _
    index As Integer _
) Implements ICollection.CopyTo

Visual C++

private:
virtual void System.Collections.ICollection.CopyTo ( _
    Array^ array,
    int index
) sealed = ICollection::CopyTo

Parameters

array
    Array
    Array to copy to.

index
    Int32
    Starting index in array to copy to.
See Also

CollectionBase<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase<Of <T>>...::System.Collections.IEnumerable.GetEnumerator
Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator<T> that was overridden by the derived classes to enumerate the members of the collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private IEnumerator IEnumerable.GetEnumerator()

Visual Basic (Declaration)

Private Function System.Collections.IEnumerable.GetEnumerator As IEnum

Visual C++

private:
virtual IEnumerator^ System.Collections.IEnumerable.GetEnumerator ()

Return Value

An IEnumerator that can be used to iterate the collection.
See Also

CollectionBase<Of<T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase(Of T)>..::.ToArray Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual T[] ToArray()
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function ToArray As T()
```

**Visual C++**

```cpp
public:
virtual array<T>^ ToArray ()
```

**Return Value**

An array containing all the elements in the collection, in order.
See Also

CollectionBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase(Of T)::.ToString Method

See Also

- Visual Basic (Declaration) - Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Shows the string representation of the collection. The string representation contains a list of the items in the collection. Contained collections (except string) are expanded recursively.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override string ToString()

Visual Basic (Declaration)

Public Overrides Function ToString As String

Visual C++

public:
virtual String^ ToString () override

Return Value

The string representation of the collection.
See Also

CollectionBase(Of T) Class
Winellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase<Of <T>>...::TrueForAll Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if all of the items in the collection satisfy the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual bool TrueForAll(
    Predicate<T> predicate
)

Visual Basic (Declaration)

Public Overridable Function TrueForAll ( _
    predicate As Predicate(Of T) _
) As Boolean

Visual C++

public:
    virtual bool TrueForAll ( _
        Predicate<T>^ predicate
    )

Parameters

predicate
    Predicate<(Of <T>)>
    A delegate that defines the condition to check for.

Return Value

True if all of the items in the collection satisfy the condition defined by
predicate, or if the collection is empty. False if one or more items in the
collection do not satisfy predicate.
See Also

CollectionBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase(Of T) Properties

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

**C#**

[SerializableAttribute]
public abstract class CollectionBase<T>

**Visual Basic (Declaration)**

<SerializableAttribute> _
Public MustInherit Class CollectionBase(Of T)

**Visual C++**

[SerializableAttribute]
generic<typename T>
public ref class CollectionBase abstract
Type Parameters

T

The type exposes the following properties.
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Must be overridden to provide the number of items in the collection.</td>
</tr>
</tbody>
</table>
### Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;T&gt;</code>::&lt;IsReadOnly&gt;</td>
<td>Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td><code>ICollection::&lt;IsSynchronized&gt;</code></td>
<td>Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td><code>ICollection::&lt;SyncRoot&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

CollectionBase<Of <T>>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase<
(OF <T>))>.::Count Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Must be overridden to provide the number of items in the collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public abstract int Count { get; }
```

**Visual Basic (Declaration)**

```vbnet
Public MustOverride ReadOnly Property Count As Integer
```

**Visual C++**

```cpp
public:
virtual property int Count {
    int get () abstract;
}
```

**Field Value**

The number of items in the collection.
See Also

CollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private bool ICollection<T>.IsReadOnly{ get; }

Visual Basic (Declaration)

Private ReadOnly Property System.Collections.Generic.ICollection<T>.IsReadOnly

Visual C++

private:
virtual property bool System.Collections.Generic.ICollection<T>.IsReadOnly = bool get () sealed = ICollection<T>::IsReadOnly::get;

}
See Also

CollectionBase<(Of <T>)) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase<Of <T>>...::System.Collections.ICollection.IsSynchronized

**Property**

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Indicates whether the collection is synchronized.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private bool ICollection.IsSynchronized{ get; }

Visual Basic (Declaration)

Private ReadOnly Property System.Collections.ICollection.IsSynchronized

Visual C++

private:
virtual property bool System.Collections.ICollection.IsSynchronized
    bool get () sealed = ICollection::IsSynchronized::get;

Field Value

Always returns false, indicating that the collection is not synchronized.
See Also

CollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
CollectionBase<Of <T>>...::System.Collections.ICollection.SyncRoot

Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Indicates the synchronization object for this collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
private Object ICollection.SyncRoot { get; }
```

**Visual Basic (Declaration)**

```vbnet
Private ReadOnly Property System.Collections.ICollection.SyncRoot As Object
```

**Visual C++**

```cpp
private:
virtual property Object^ System.Collections.ICollection.SyncRoot { Object^ get () sealed = ICollection::SyncRoot::get; }
```

### Field Value

Always returns this.
See Also

CollectionBase<Of T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
The Deque class implements a type of list known as a Double Ended Queue. A Deque is quite similar to a List, in that items have indices (starting at 0), and the item at any index can be efficiently retrieved. The difference between a List and a Deque lies in the efficiency of inserting elements at the beginning. In a List, items can be efficiently added to the end, but inserting an item at the beginning of the List is slow, taking time proportional to the size of the List. In a Deque, items can be added to the beginning or end equally efficiently, regardless of the number of items in the Deque. As a trade-off for this increased flexibility, Deque is somewhat slower than List (but still constant time) when being indexed to get or retrieve elements.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
[SerializableAttribute]
public class Deque<T> : ListBase<T>, ICloneable
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute>
Public Class Deque(Of T)
    Inherits ListBase(Of T)
    Implements ICloneable
```

**Visual C++**

```cpp
[SerializableAttribute]
generic<typename T>
public ref class Deque : public ListBase<T>,
    ICloneable
```
Type Parameters

T

The type of items stored in the Deque.
Remarks

The Deque class can also be used as a more flexible alternative to the Queue and Stack classes. Deque is as efficient as Queue and Stack for adding or removing items, but is more flexible: it allows access to all items in the queue, and allows adding or removing from either end.

Deque is implemented as a ring buffer, which is grown as necessary. The size of the buffer is doubled whenever the existing capacity is too small to hold all the elements.
Inheritance Hierarchy

System..:::Object
Wintellect.PowerCollections..:::CollectionBase<(Of <T>)>
Wintellect.PowerCollections..:::ListBase<(Of <T>)>
Wintellect.PowerCollections..:::Deque<(Of <T>)>
See Also

Deque<Of <T>> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T) Members

See Also  Methods  Properties  Constructors  Explicit Interface Implementations

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

**C#**

```csharp
[SerializableAttribute]
public class Deque<T>
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute>_Public Class Deque(Of T)
```

**Visual C++**

```cpp
[SerializableAttribute]
generic<typename T>
public ref class Deque
```
**Type Parameters**

T

The type exposes the following members.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Deque&lt;Of&lt;T&gt;&gt;</code> <code>Deque&lt;Of&lt;T&gt;&gt;</code> <code>New</code></td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Adds an item to the back of the Deque. The indices of all existing items in the Deque are unchanged. This method is equivalent to AddToBack(item). Adds a collection of items to the back of the Deque. The indices of all existing items in the Deque are unchanged. This method is equivalent to AddManyToBack(item). The last item in the added collection becomes the last item in the Deque.</td>
</tr>
<tr>
<td><strong>AddManyToBack</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AddManyToFront</strong></td>
<td>Adds a collection of items to the front of the Deque. The indices of all existing items in the Deque are increased by the number of items inserted. The first item in the added collection becomes the first item in the Deque.</td>
</tr>
<tr>
<td><strong>AddToBack</strong></td>
<td>Adds an item to the back of the Deque. The indices of all existing items in the Deque are unchanged. This method is equivalent to Insert(Count, item) but is a little more efficient. Adds an item to the back of the Deque. The indices of all existing items in the Deque are unchanged. This method is equivalent to Insert(Count, item) but is a little more efficient.</td>
</tr>
<tr>
<td><strong>AddToFront</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this list. The returned IList&lt;T&gt; provides a view of the list that prevents modifications to the list. Use the method to provide access to the list without allowing changes. Since the returned object is just a view, changes to the list will be reflected in the view. (Inherited from ListBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all items from the Deque. (Overrrides ListBase&lt;Of&lt;T&gt;&gt;::Clear().)</td>
</tr>
<tr>
<td><strong>Clone</strong></td>
<td>Creates a new Deque that is a copy of this one. Makes a deep clone of this Deque. A new Deque is created with a clone of each element of this set, by calling ICloneable.Clone on each element. If T is a value type,</td>
</tr>
<tr>
<td><strong>CloneContents</strong></td>
<td></td>
</tr>
</tbody>
</table>
then each element is copied as if by simple assignment. Determines if the list contains any item that compares equal to item. The implementation simply checks whether IndexOf(item) returns a non-negative value. (Inherited from ListBase<Of <T>>.)

- **Contains**
  Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from CollectionBase<Of <T>>.)

- **ConvertAll<Of <TOutput>>**
  Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from CollectionBase<Of <T>>.)

- **CopyTo**
  Determines whether the specified **Object** is equal to the current **Object**. (Inherited from **Object**.)

- **CountWhere**
  Determines if the collection contains any item that satisfies the condition defined by predicate. (Inherited from CollectionBase<Of <T>>.)

- **Equals**
  Finds the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned. (Inherited from ListBase<Of <T>>.)

- **Find**
 Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from CollectionBase<Of <T>>.)

- **FindAll**
  Overloaded.

- **FindIndex**
  Finds the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned. (Inherited from ListBase<Of <T>>.)

- **FindLast**
  Overloaded.

- **FindLastIndex**
  Performs the specified action on each item in this collection. (Inherited from CollectionBase<Of <T>>.)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetAtBack</td>
<td>Retrieves the item currently at the back of the Deque. The Deque is unchanged. This method is equivalent to <code>deque[deque.Count - 1]</code> (except that a different exception is thrown).</td>
</tr>
<tr>
<td>GetAtFront</td>
<td>Retrieves the item currently at the front of the Deque. The Deque is unchanged. This method is equivalent to <code>deque[0]</code> (except that a different exception is thrown).</td>
</tr>
<tr>
<td>GetEnumerator</td>
<td>Enumerates all of the items in the list, in order. The item at index 0 is enumerated first, then the item at index 1, and so on. If the items are added to or removed from the Deque during enumeration, the enumeration ends with an InvalidOperationException. (Overrides <code>ListBase&lt;Of &lt;T&gt;&gt;&gt;::GetEnumerator()</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. GetHashCode() is suitable for use in hashing algorithms and data structures like a hash table. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Insert</td>
<td>Inserts a new item at the given index in the Deque. All items at indexes equal to or greater than index move up one index in the Deque.</td>
</tr>
<tr>
<td>InsertRange</td>
<td>Inserts a collection of items at the given index in the Deque. All items at indexes equal to or greater than index increase their indices in the Deque by the number of items inserted.</td>
</tr>
<tr>
<td>LastIndexOf</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>Range</td>
<td>Returns a view onto a sub-range of this list. Items are not copied; the returned <code>IList&lt;T&gt;</code> is simply a different view onto the same underlying items. Changes to this list are reflected in the view, and vice versa. Insertions and deletions in the view change the size of the view, but insertions and deletions in the underlying list do not. (Inherited from <code>ListBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td></td>
<td>Searches the list for the first item that compares equal to item. If one is found, it is removed. Otherwise, the list is</td>
</tr>
</tbody>
</table>
- **Remove** unchanged.
  (Inherited from ListBase<Of <T>>.)

- **RemoveAll** Removes all the items in the collection that satisfy the condition defined by predicate.
  (Inherited from CollectionBase<Of <T>>.)

- **RemoveAt** Removes the item at the given index in the Deque. All items at indexes greater than index move down one index in the Deque.
  (Overrides ListBase<Of <T>>...: RemoveAt(Int32).)

- **RemoveFromBack** Removes an item from the back of the Deque. The indices of all existing items in the Deque are unchanged. This method is equivalent to RemoveAt(Count-1) but is a little more efficient.

- **RemoveFromFront** Removes an item from the front of the Deque. The indices of all existing items in the Deque are decreased by 1. This method is equivalent to RemoveAt(0) but is a little more efficient.

- **RemoveRange** Removes a range of items at the given index in the Deque. All items at indexes greater than index move down count indices in the Deque.

- **ToArray** Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.
  (Inherited from CollectionBase<Of <T>>.)

- **ToString** Shows the string representation of the collection. The string representation contains a list of the items in the collection. Contained collections (except string) are expanded recursively.
  (Inherited from CollectionBase<Of <T>>.)

- **TrimToSize** Trims the amount of memory used by the Deque by changing the Capacity to be equal to Count. If no more items will be added to the Deque, calling TrimToSize will reduce the amount of memory used by the Deque.

- **TrueForAll** Determines if all of the items in the collection satisfy the condition defined by predicate.
  (Inherited from CollectionBase<Of <T>>.)

  Finds the first item in the list that satisfies the condition
- **TryFind**
  defined by predicate.
  (Inherited from ListBase<Of<T>>.)

- **TryFindLast**
  Finds the last item in the list that satisfies the condition
defined by predicate.
  (Inherited from ListBase<Of<T>>.)
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>Gets or sets the capacity of the Deque. The Capacity is the number of items that this Deque can hold without expanding its internal buffer. Since Deque will automatically expand its buffer when necessary, in almost all cases it is unnecessary to worry about the capacity. However, if it is known that a Deque will contain exactly 1000 items eventually, it can slightly improve efficiency to set the capacity to 1000 up front, so that the Deque does not have to expand automatically.</td>
</tr>
<tr>
<td>Count</td>
<td>Gets the number of items currently stored in the Deque. The last item in the Deque has index Count-1. (Overrides ListBase&lt;Of &lt;T&gt;&gt;::Count.)</td>
</tr>
<tr>
<td>Item</td>
<td>Gets or sets an item at a particular index in the Deque. (Overrides ListBase&lt;Of &lt;T&gt;&gt;::_:Item[Int32]).</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;(Of &lt;T&gt;),::IsReadOnly</code></td>
<td>(Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;),</code>).  Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;),</code>).)</td>
</tr>
<tr>
<td><code>ICollection::CopyTo</code></td>
<td>Indicates whether the collection is synchronized. (Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;),</code>).)</td>
</tr>
<tr>
<td><code>ICollection::IsSynchronized</code></td>
<td>Indicates the synchronization object for this collection. (Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;),</code>).)</td>
</tr>
<tr>
<td><code>ICollection::SyncRoot</code></td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;),</code>).)</td>
</tr>
<tr>
<td><code>IEnumerable::GetEnumerator</code></td>
<td>Adds an item to the end of the list. This method is equivalent to calling: (Inherited from <code>ListBase&lt;(Of &lt;T&gt;),</code>).)</td>
</tr>
</tbody>
</table>
| `IList::Add` | Insert(Count, item) (Inherited from `ListBase<(Of <T>),`).) Removes all the items from the list,
**IList::Clear**
resulting in an empty list.
(Inherited from `ListBase<OF <T>>`.)

**IList::Contains**
Determines if the list contains any item that compares equal to value.
(Inherited from `ListBase<OF <T>>`.)

**IList::IndexOf**
Find the first occurrence of an item equal to value in the list, and returns the index of that item.
(Inherited from `ListBase<OF <T>>`.)

**IList::Insert**
Insert a new item at the given index.
(Inherited from `ListBase<OF <T>>`.)

**IList::IsFixedSize**
Returns whether the list is a fixed size. This implementation always returns false.
(Inherited from `ListBase<OF <T>>`.)

**IList::IsReadOnly**
Returns whether the list is read only. This implementation returns the value from `ICollection<T>.IsReadOnly`, which is by default, false.
(Inherited from `ListBase<OF <T>>`.)

**IList::Item**
Gets or sets the value at a particular index in the list.
(Inherited from `ListBase<OF <T>>`.)

**IList::Remove**
Searches the list for the first item that compares equal to value. If one is found, it is removed. Otherwise, the list is unchanged.
(Inherited from `ListBase<OF <T>>`.)

**IList::RemoveAt**
Removes the item at the given index.
(Inherited from `ListBase<OF <T>>`.)

**ICloneable::Clone**
Creates a new `Deque` that is a copy of this one.
See Also

Deque<Of <T>>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<Of <T>> Constructor

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Deque&lt;(Of &lt;T&gt;)&gt;Deque&lt;(Of &lt;T&gt;)&gt;New()</code></td>
<td>Create a new Deque that is initially empty.</td>
</tr>
<tr>
<td><code>Deque&lt;(Of &lt;T&gt;)&gt;Deque&lt;(Of &lt;T&gt;)&gt;New(IEnumerable&lt;(Of &lt;T&gt;)&gt;)</code></td>
<td>Create a new Deque initialized with the items from the passed collection, in order.</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Deque(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<(Of <T>)> Constructor

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Create a new Deque that is initially empty.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public Deque()

Visual Basic (Declaration)

Public Sub New

Visual C++

public:
Deque ()
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Dequeue<(Of <T>)> Constructor (IEnumerable<(Of <T>))>

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Create a new Deque initialized with the items from the passed collection, in order.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public Deque(
        IEnumerable&lt;T&gt; collection
)

Visual Basic (Declaration)

Public Sub New (_
        collection As IEnumerable(Of T) _
)

Visual C++

public:
    Deque (
        IEnumerable&lt;T&gt;^ collection
    )

Parameters

collection
    IEnumerable&lt;(Of &lt;T&gt;)&gt;
    A collection of items to initialize the Deque with.
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T) Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

**C#**

[SerializableAttribute]
public class Deque<T>

**Visual Basic (Declaration)**

<SerializableAttribute> _
Public Class Deque(Of T)

**Visual C++**

[SerializableAttribute]
generic<typename T>
public ref class Deque
Type Parameters

T

The type exposes the following methods.
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Adds an item to the back of the Deque. The indices of all existing items in the Deque are unchanged. This method is equivalent to AddToBack(item).</td>
</tr>
<tr>
<td><strong>AddManyToBack</strong></td>
<td>Adds a collection of items to the back of the Deque. The indices of all existing items in the Deque are unchanged. The last item in the added collection becomes the last item in the Deque.</td>
</tr>
<tr>
<td><strong>AddManyToFront</strong></td>
<td>Adds a collection of items to the front of the Deque. The indices of all existing items in the Deque are increased by the number of items inserted. The first item in the added collection becomes the first item in the Deque.</td>
</tr>
<tr>
<td><strong>AddToBack</strong></td>
<td>Adds an item to the back of the Deque. The indices of all existing items in the Deque are unchanged. This method is equivalent to Insert(Count, item) but is a little more efficient.</td>
</tr>
<tr>
<td><strong>AddToFront</strong></td>
<td>Adds an item to the front of the Deque. The indices of all existing items in the Deque are increased by 1. This method is equivalent to Insert(0, item) but is a little more efficient.</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this list. The returned IList&lt;T&gt; provides a view of the list that prevents modifications to the list. Use the method to provide access to the list without allowing changes. Since the returned object is just a view, changes to the list will be reflected in the view.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all items from the Deque. (Overides ListBase&lt;Of&lt;T&gt;&gt;::.Clear().)</td>
</tr>
<tr>
<td><strong>Clone</strong></td>
<td>Creates a new Deque that is a copy of this one.</td>
</tr>
<tr>
<td><strong>CloneContents</strong></td>
<td>Makes a deep clone of this Deque. A new Deque is created with a clone of each element of this set, by calling ICloneable.Clone on each element. If T is a value type,</td>
</tr>
</tbody>
</table>
then each element is copied as if by simple assignment. Determines if the list contains any item that compares equal to item. The implementation simply checks whether IndexOf(item) returns a non-negative value. (Inherited from ListBase<Of<T>>.)

- **Contains**
  - ConvertAll<Of<TOutput>>
    - Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from CollectionBase<Of<T>>.)

- **CopyTo**
  - Overloaded.

- **CountWhere**
  - Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from CollectionBase<Of<T>>.)

- **Equals**
  - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)

- **Exists**
  - Determines if the collection contains any item that satisfies the condition defined by predicate. (Inherited from CollectionBase<Of<T>>.)

- **Find**
  - Finds the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned. (Inherited from ListBase<Of<T>>.)

- **FindAll**
  -Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from CollectionBase<Of<T>>.)

- **FindIndex**
  - Overloaded.

- **FindLast**
  - Finds the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned. (Inherited from ListBase<Of<T>>.)

- **FindLastIndex**
  - Overloaded.

- **ForEach**
  - Performs the specified action on each item in this collection. (Inherited from CollectionBase<Of<T>>.)
**GetAtBack**
Retrieves the item currently at the back of the Deque. The Deque is unchanged. This method is equivalent to deque[deque.Count - 1] (except that a different exception is thrown).

**GetAtFront**
Retrieves the item currently at the front of the Deque. The Deque is unchanged. This method is equivalent to deque[0] (except that a different exception is thrown).

**GetEnumerator**
Enumerates all of the items in the list, in order. The item at index 0 is enumerated first, then the item at index 1, and so on. If the items are added to or removed from the Deque during enumeration, the enumeration ends with an InvalidOperationException.
(Overrides ListBase<Of <T>>...::GetEnumerator().)

**GetHashCode**
Serves as a hash function for a particular type. GetHashCode() is suitable for use in hashing algorithms and data structures like a hash table.
(Inherited from Object.)

**GetType**
Gets the Type of the current instance.
(Inherited from Object.)

**IndexOf**
Overloaded.

**Insert**
Inserts a new item at the given index in the Deque. All items at indexes equal to or greater than index move up one index in the Deque.

**InsertRange**
Inserts a collection of items at the given index in the Deque. All items at indexes equal to or greater than index increase their indices in the Deque by the number of items inserted.

**LastIndexOf**
Overloaded.

**Range**
Returns a view onto a sub-range of this list. Items are not copied; the returned IList<T> is simply a different view onto the same underlying items. Changes to this list are reflected in the view, and vice versa. Insertions and deletions in the view change the size of the view, but insertions and deletions in the underlying list do not.
(Inherited from ListBase<Of <T>>.)

Searches the list for the first item that compares equal to item. If one is found, it is removed. Otherwise, the list is
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Inheritance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Remove</strong></td>
<td>Removes all the items in the collection that satisfy the condition defined by predicate.</td>
<td>(Inherited from ListBase&lt;Of&lt;T&gt;&gt;).</td>
</tr>
<tr>
<td><strong>RemoveAll</strong></td>
<td>Removes the item at the given index in the Deque. All items at indexes greater than index move down one index in the Deque.</td>
<td>(Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;).</td>
</tr>
<tr>
<td><strong>RemoveAt</strong></td>
<td>Removes an item from the back of the Deque. The indices of all existing items in the Deque are unchanged. This method is equivalent to RemoveAt(Count-1) but is a little more efficient.</td>
<td>(Overrides ListBase&lt;Of&lt;T&gt;&gt;::RemoveAt(Int32).)</td>
</tr>
<tr>
<td><strong>RemoveFromBack</strong></td>
<td>Removes an item from the front of the Deque. The indices of all existing items in the Deque are decreased by 1. This method is equivalent to RemoveAt(0) but is a little more efficient.</td>
<td></td>
</tr>
<tr>
<td><strong>RemoveFromFront</strong></td>
<td>Removes a range of items at the given index in the Deque. All items at indexes greater than index move down count indices in the Deque.</td>
<td></td>
</tr>
<tr>
<td><strong>RemoveRange</strong></td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.</td>
<td>(Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;).</td>
</tr>
<tr>
<td><strong>ToArray</strong></td>
<td>Shows the string representation of the collection. The string representation contains a list of the items in the collection. Contained collections (except string) are expanded recursively.</td>
<td>(Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;).</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Trims the amount of memory used by the Deque by changing the Capacity to be equal to Count. If no more items will be added to the Deque, calling TrimToSize will reduce the amount of memory used by the Deque.</td>
<td></td>
</tr>
<tr>
<td><strong>TrueForAll</strong></td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate.</td>
<td>(Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;).</td>
</tr>
</tbody>
</table>
**TryFind**

defined by predicate.
(Inherited from `ListBase<Of T>`.)

**TryFindLast**

Finds the last item in the list that satisfies the condition defined by predicate.
(Inherited from `ListBase<Of T>`.)
# Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection::CopyTo</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.) Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IEnumerable::GetEnumerator</code></td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IList::Add</code></td>
<td>Adds an item to the end of the list. This method is equivalent to calling: <code>Insert(Count, item)</code> (Inherited from <code>ListBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IList::Clear</code></td>
<td>Removes all the items from the list, resulting in an empty list. (Inherited from <code>ListBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IList::Contains</code></td>
<td>Determines if the list contains any item that compares equal to value. (Inherited from <code>ListBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IList::IndexOf</code></td>
<td>Find the first occurrence of an item equal to value in the list, and returns the index of that item. (Inherited from <code>ListBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IList::Insert</code></td>
<td>Insert a new item at the given index.</td>
</tr>
</tbody>
</table>
IList::Remove

Searches the list for the first item that compares equal to value. If one is found, it is removed. Otherwise, the list is unchanged.

(IInherited from ListBase<Of<T>>.)

IList::RemoveAt

Removes the item at the given index.

(IInherited from ListBase<Of<T>>.)

ICloneable::Clone

Creates a new Deque that is a copy of this one.
See Also

Deque<Of<$T$>>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T)>...::Add Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Add an item to the back of the Deque. The indices of all existing items in the Deque are unchanged. This method is equivalent to AddToBack(item).

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public void Add(
    T item
)

Visual Basic (Declaration)

Public Sub Add (_
    item As T _
)

Visual C++

public:
virtual void Add (  
    T item
) sealed

Parameters

item
    T
    The item to add.
Remarks

Adding an item to the back of the Deque takes a small constant amount of time, regardless of how many items are in the Deque.
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<Of <T>>...::AddManyToBack Method

See Also

Visual Basic (Declaration) ▶ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a collection of items to the back of the Deque. The indices of all existing items in the Deque are unchanged. The last item in the added collection becomes the last item in the Deque.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public void AddManyToBack(
    IEnumerable<T> collection
)

Visual Basic (Declaration)

Public Sub AddManyToBack ( _
    collection As IEnumerable(Of T) _
)

Visual C++

public:
void AddManyToBack ( 
    IEnumerable<T^ collection
)

Parameters

collection
    IEnumerable(Of <T>)
    The collection of item to add.
Remarks

This method takes time $O(M)$, where $M$ is the number of items in the collection.
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<(Of <T>)>...::AddManyToFront Method

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a collection of items to the front of the Deque. The indices of all existing items in the Deque are increased by the number of items inserted. The first item in the added collection becomes the first item in the Deque.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public void AddManyToFront(
    IEnumerable<T> collection
)

Visual Basic (Declaration)

Public Sub AddManyToFront ( _
    collection As IEnumerable(Of T) _
)

Visual C++

public:
void AddManyToFront ( 
    IEnumerable<T>^ collection
)

Parameters

collection
    IEnumerable<(Of <T>)> The collection of items to add.
Remarks

This method takes time $O(M)$, where $M$ is the number of items in the collection.
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque< Of T >>:: AddToBack Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds an item to the back of the Deque. The indices of all existing items in the Deque are unchanged. This method is equivalent to Insert(Count, item) but is a little more efficient.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public void AddToBack(
    T item
)

Visual Basic (Declaration)

Public Sub AddToBack (_
    item As T _
)

Visual C++

public:
void AddToBack (  
    T item
)

Parameters

item
    T
    The item to add.
Remarks

Adding an item to the back of the Deque takes a small constant amount of time, regardless of how many items are in the Deque.
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<Of <T>>...::AddToFront Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds an item to the front of the Deque. The indices of all existing items in the Deque are increased by 1. This method is equivalent to Insert(0, item) but is a little more efficient.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public void AddToFront(
    T item
)

Visual Basic (Declaration)

Public Sub AddToFront ( _
    item As T _
)

Visual C++

public:
void AddToFront (  
    T item
)

Parameters

item  
T  
The item to add.
Remarks

Adding an item to the front of the Deque takes a small constant amount of time, regardless of how many items are in the Deque.
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T)::.Clear Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all items from the Deque.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public override sealed void Clear()
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Sub Clear
```

**Visual C++**

```cpp
public:
virtual void Clear () override sealed
```
Remarks

Clearing the Deque takes a small constant amount of time, regardless of how many items are currently in the Deque.
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T)>...::Clone Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new Deque that is a copy of this one.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public Deque<T> Clone()

Visual Basic (Declaration)

Public Function Clone As Deque(Of T)

Visual C++

public:
Deque<T>^ Clone()

ReturnValue

A copy of the current deque.
Remarks

Copying a Deque takes $O(N)$ time, where $N$ is the number of items in this Deque.
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T)>...:::CloneContents Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a deep clone of this Deque. A new Deque is created with a clone of each element of this set, by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public Deque<T> CloneContents()

Visual Basic (Declaration)

Public Function CloneContents As Deque(Of T)

Visual C++

public:
Deque<T>^ CloneContents ()

Return Value

The cloned Deque.
Remarks

If T is a reference type, it must implement ICloneable. Otherwise, an InvalidOperationException is thrown.

Cloning the Deque takes time O(N), where N is the number of items in the Deque.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>T is a reference type that does not implement <code>ICloneable</code>.</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<Of <T>><::CopyTo Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CopyTo(array&lt;T&gt;[])</code></td>
<td>Copies all the items in the list, in order, to array, starting at index 0. (Inherited from <code>ListBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>CopyTo(array&lt;T&gt;[](), Int32)</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>CopyTo(Int32, array&lt;T&gt;[](), Int32, Int32)</code></td>
<td>Copies a range of elements from the list to array, starting at arrayIndex. (Inherited from <code>ListBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Deque(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<Of <T>>...::FindIndex Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FindIndex(Predicate(Of&lt;T&gt;))</td>
<td>Finds the index of the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned. (Inherited from ListBase(Of&lt;T&gt;).)</td>
</tr>
<tr>
<td>FindIndex(Int32, Predicate(Of&lt;T&gt;))</td>
<td>Finds the index of the first item, in the range of items extending from index to the end, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned. (Inherited from ListBase(Of&lt;T&gt;).)</td>
</tr>
<tr>
<td>FindIndex(Int32, Int32, Predicate(Of&lt;T&gt;))</td>
<td>Finds the index of the first item, in the range of count items starting from index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned. (Inherited from ListBase(Of&lt;T&gt;).)</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Deque(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<(Of <T>)>::FindLastIndex Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FindLastIndex(Predicate&lt;Of&lt;T&gt;&gt;)</strong></td>
<td>Finds the index of the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned. (Inherited from ListBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>FindLastIndex(Int32, Predicate&lt;Of&lt;T&gt;&gt;)</strong></td>
<td>Finds the index of the last item, in the range of items extending from the beginning of the list to index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned. (Inherited from ListBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>FindLastIndex(Int32, Int32, Predicate&lt;Of&lt;T&gt;&gt;)</strong></td>
<td>Finds the index of the last item, in the range of count items ending at index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned. (Inherited from ListBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Deque(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<(Of <T>)>...::GetAtBack Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Retreives the item currently at the back of the Deque. The Deque is unchanged. This method is equivalent to `deque[deque.Count - 1]` (except that a different exception is thrown).

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public T GetAtBack()
```

**Visual Basic (Declaration)**

```vbnet
Public Function GetAtBack As T
```

**Visual C++**

```cpp
public:
T GetAtBack ()
```

**Return Value**

The item at the back of the Deque.
Remarks

Retrieving the item at the back of the Deque takes a small constant amount of time, regardless of how many items are in the Deque.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::InvalidOperationException</code></td>
<td>The Deque is empty.</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<(Of <T>)>...::GetAtFront Method

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Retreives the item currently at the front of the Deque. The Deque is unchanged. This method is equivalent to deque[0] (except that a different exception is thrown).

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public T GetAtFront()

Visual Basic (Declaration)

Public Function GetAtFront As T

Visual C++

public:
T GetAtFront ()

Return Value

The item at the front of the Deque.
Remarks

Retreiving the item at the front of the Deque takes a small constant amount of time, regardless of how many items are in the Deque.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::InvalidOperationException</code></td>
<td>The Deque is empty.</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T)>...::GetEnumerator Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerates all of the items in the list, in order. The item at index 0 is enumerated first, then the item at index 1, and so on. If the items are added to or removed from the Deque during enumeration, the enumeration ends with an InvalidOperationException.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public override sealed IEnumerator<T> GetEnumerator()
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Function GetEnumerator As IEnumerable
```

**Visual C++**

```cpp
public:
virtual IEnumerable<T>^ GetEnumerator () override sealed
```

### Return Value

An `IEnumerator<T>` that enumerates all the items in the list.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..:::InvalidOperationException</code></td>
<td>The Deque has an item added or deleted during the enumeration.</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<Of <T>>...::IndexOf Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IndexOf(T)</code></td>
<td>Finds the index of the first item in the list that is equal to item.</td>
</tr>
<tr>
<td>(Inherited from</td>
<td>ListBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><code>IndexOf(T, Int32)</code></td>
<td>Finds the index of the first item, in the range of items extending from</td>
</tr>
<tr>
<td></td>
<td>index to the end, that is equal to item.</td>
</tr>
<tr>
<td>(Inherited from</td>
<td>ListBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><code>IndexOf(T, Int32, Int32)</code></td>
<td>Finds the index of the first item, in the range of count items</td>
</tr>
<tr>
<td></td>
<td>starting from index, that is equal to item.</td>
</tr>
<tr>
<td>(Inherited from</td>
<td>ListBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
</tbody>
</table>
See Also

Deque<Of<T>> Class
Deque<Of<T>> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<Of <T>>...::Insert Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Inserts a new item at the given index in the Deque. All items at indexes equal to or greater than index move up one index in the Deque.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public void Insert(
    int index,
    T item
)
```

### Visual Basic (Declaration)

```vbnet
Public Sub Insert (
    index As Integer, _
    item As T _
)
```

### Visual C++

```cpp
public:
    virtual void Insert (
        int index,
        T item
    ) sealed
```

## Parameters

### index

*Int32*

The index in the Deque to insert the item at. After the insertion, the inserted item is located at this index. The front item in the Deque has index 0.

### item

*T*

The item to insert at the given index.
Remarks

The amount of time to insert an item in the Deque is proportional to the distance of index from the closest end of the Deque: $O(\text{Min}(\text{index, Count - index}))$. Thus, inserting an item at the front or end of the Deque is always fast; the middle of the Deque is the slowest place to insert.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>index is less than zero or greater than Count.</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T)\>\:\:InsertRange Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Inserts a collection of items at the given index in the Deque. All items at indexes equal to or greater than index increase their indices in the Deque by the number of items inserted.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public void InsertRange(
    int index,
    IEnumerable<T> collection
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub InsertRange (_
    index As Integer, _
    collection As IEnumerable(Of T) _
)
```

**Visual C++**

```cpp
public:
void InsertRange ( 
    int index, 
    IEnumerable<T>^ collection
)
```

**Parameters**

**index**

*Int32*

The index in the Deque to insert the collection at. After the insertion, the first item of the inserted collection is located at this index. The front item in the Deque has index 0.

**collection**

*IEnumerable<Of <T>>*

The collection of items to insert at the given index.
Remarks

The amount of time to insert a collection in the Deque is proportional to the distance of index from the closest end of the Deque, plus the number of items inserted (M): $O(M + \text{Min}(\text{index}, \text{Count} - \text{index}))$. 
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>index is less than zero or greater than Count.</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T)...:LastIndexOf Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![LastIndexOf(T)]</td>
<td>Finds the index of the last item in the list that is equal to item.</td>
</tr>
<tr>
<td>![ListBase&lt;(Of &lt;T&gt;)]</td>
<td>(Inherited from ListBase&lt;(Of &lt;T&gt;)&gt;)</td>
</tr>
<tr>
<td>![LastIndexOf(T, Int32)]</td>
<td>Finds the index of the last item, in the range of items extending from the beginning of the list to index, that is</td>
</tr>
<tr>
<td>![ListBase&lt;(Of &lt;T&gt;)]</td>
<td>equal to item.</td>
</tr>
<tr>
<td>![LastIndexOf(T, Int32, Int32)]</td>
<td>(Inherited from ListBase&lt;(Of &lt;T&gt;)&gt;)</td>
</tr>
<tr>
<td>![ListBase&lt;(Of &lt;T&gt;)]</td>
<td>Finds the index of the last item, in the range of count items ending at index, that is equal to item.</td>
</tr>
<tr>
<td>![ListBase&lt;(Of &lt;T&gt;)]</td>
<td>(Inherited from ListBase&lt;(Of &lt;T&gt;)&gt;)</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Deque(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<\(\text{Of } \langle T \rangle \rangle \ldots \text{:RemoveAt Method}

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes the item at the given index in the Deque. All items at indexes greater than index move down one index in the Deque.

**Namespace:** Wintellect.PowerCollections  
**Assembly:**  PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public override sealed void RemoveAt(int index)
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Sub RemoveAt ( _
       index As Integer _
    )
```

**Visual C++**

```cpp
public:
virtual void RemoveAt ( _
    int index
 ) override sealed
```

**Parameters**

**index**

`Int32`

The index in the list to remove the item at. The first item in the list has index 0.
Remarks

The amount of time to delete an item in the Deque is proportional to the distance of index from the closest end of the Deque: $O(\text{Min}(\text{index}, \text{Count} - 1 - \text{index}))$. Thus, deleting an item at the front or end of the Deque is always fast; the middle of the Deque is the slowest place to delete.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>index is less than zero or greater than or equal to Count.</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<Of <T>>...::RemoveFromBack Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes an item from the back of the Deque. The indices of all existing items in the Deque are unchanged. This method is equivalent to RemoveAt(Count-1) but is a little more efficient.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public T RemoveFromBack()

Visual Basic (Declaration)

Public Function RemoveFromBack As T

Visual C++

public:
T RemoveFromBack ()
Remarks

Removing an item from the back of the Deque takes a small constant amount of time, regardless of how many items are in the Deque.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...::InvalidOperationException</td>
<td>The Deque is empty.</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<(Of <T>)>::RemoveFromFront Method

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes an item from the front of the Deque. The indices of all existing items in the Deque are decreased by 1. This method is equivalent to RemoveAt(0) but is a little more efficient.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public T RemoveFromFront()
```

**Visual Basic (Declaration)**

```vbnet
Public Function RemoveFromFront As T
```

**Visual C++**

```cpp
public:
T RemoveFromFront ()
```

**Return Value**

The item that was removed.
Remarks

Removing an item from the front of the Deque takes a small constant amount of time, regardless of how many items are in the Deque.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...::InvalidOperationException</td>
<td>The Deque is empty.</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T)>...:RemoveRange Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes a range of items at the given index in the Deque. All items at indexes greater than index move down count indices in the Deque.

Namespace: Wintellect.PowerCollections

Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public void RemoveRange(
    int index,
    int count
)

Visual Basic (Declaration)

Public Sub RemoveRange ( _
    index As Integer, _
    count As Integer _
)

Visual C++

public:
void RemoveRange ( 
    int index,
    int count
)

Parameters

index

Int32
The index in the list to remove the range at. The first item in the list has index 0.

count

Int32
The number of items to remove.
Remarks

The amount of time to delete count items in the Deque is proportional to the distance to the closest end of the Deque: $O(\text{Min}(\text{index}, \text{Count} - \text{index} - \text{count}))$. 
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>Index is less than zero or greater than or equal to Count, or count is less than zero or too large.</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T)>.::.System.ICloneable.Clone Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new Deque that is a copy of this one.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
- **Syntax**

  **C#**

  ```csharp
  private Object ICloneable.Clone()
  ```

  **Visual Basic (Declaration)**

  ```vbnet
  Private Function System.ICloneable.Clone As Object Implements ICloneable
  ```

  **Visual C++**

  ```cpp
  private:
  virtual Object ^ System.ICloneable.Clone () sealed = ICloneable::Clone
  ```

  **Return Value**

  A copy of the current deque.
Remarks

Copying a Deque takes $O(N)$ time, where $N$ is the number of items in this Deque.
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T)

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Trims the amount of memory used by the Deque by changing the Capacity to be equal to Count. If no more items will be added to the Deque, calling TrimToSize will reduce the amount of memory used by the Deque.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public void TrimToSize()
```

**Visual Basic (Declaration)**

```vbnet
Public Sub TrimToSize
```

**Visual C++**

```cpp
public:
void TrimToSize ()
```
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T) Properties

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class Deque<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class Deque(Of T)

Visual C++

[SerializableAttribute]
generic< typename T>
public ref class Deque
Type Parameters

T

The type exposes the following properties.
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Capacity</strong></td>
<td>Gets or sets the capacity of the Deque. The Capacity is the number of items that this Deque can hold without expanding its internal buffer. Since Deque will automatically expand its buffer when necessary, in almost all cases it is unnecessary to worry about the capacity. However, if it is known that a Deque will contain exactly 1000 items eventually, it can slightly improve efficiency to set the capacity to 1000 up front, so that the Deque does not have to expand automatically.</td>
</tr>
<tr>
<td><strong>Count</strong></td>
<td>Gets the number of items currently stored in the Deque. The last item in the Deque has index Count-1. (Overrides <code>ListBase&lt;Of&lt;T&gt;&gt;::Count</code>.)</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Gets or sets an item at a particular index in the Deque. (Overrides <code>ListBase&lt;Of&lt;T&gt;&gt;::Item[Int32]</code>.)</td>
</tr>
</tbody>
</table>
### Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection(&lt;\text{T}&gt;)\östl.\öIsReadOnly</td>
<td>(Inherited from CollectionBase(&lt;\text{T}&gt;)\östl.) Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td>ICollection\östl.\öIsSynchronized</td>
<td>(Inherited from CollectionBase(&lt;\text{T}&gt;)\östl.) Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td>ICollection\östl.\öSyncRoot</td>
<td>(Inherited from CollectionBase(&lt;\text{T}&gt;)\östl.) Returns whether the list is a fixed size. This implementation always returns false.</td>
</tr>
<tr>
<td>IList\östl.\öIsFixedSize</td>
<td>(Inherited from ListBase(&lt;\text{T}&gt;)\östl.) Returns whether the list is read only. This implementation returns the value from ICollection(&lt;\text{T}&gt;).IsReadOnly, which is by default, false.</td>
</tr>
<tr>
<td>IList\östl.\öIsReadOnly</td>
<td>(Inherited from ListBase(&lt;\text{T}&gt;)\östl.) Gets or sets the value at a particular index in the list.</td>
</tr>
<tr>
<td>IList\östl.\öItem</td>
<td>(Inherited from ListBase(&lt;\text{T}&gt;)\östl.)</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T)
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T)>...:::Capacity Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets or sets the capacity of the Deque. The Capacity is the number of items that this Deque can hold without expanding its internal buffer. Since Deque will automatically expand its buffer when necessary, in almost all cases it is unnecessary to worry about the capacity. However, if it is known that a Deque will contain exactly 1000 items eventually, it can slightly improve efficiency to set the capacity to 1000 up front, so that the Deque does not have to expand automatically.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#
public int Capacity{ get; set;}

Visual Basic (Declaration)
Public Property Capacity As Integer

Visual C++
public:
property int Capacity {
    int get ();
    void set (int value);
}

Field Value

The number of items that this Deque can hold without expanding its internal buffer.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>The capacity is being set to less than Count, or to too large a value.</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque<Of <T>>::Count Property

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets the number of items currently stored in the Deque. The last item in the Deque has index Count-1.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public override sealed int Count{ get; }
```

Visual Basic (Declaration)

```vbnet
Public Overrides NotOverridable ReadOnly Property Count As Integer
```

Visual C++

```cpp
public:
virtual property int Count {
        int get () override sealed;
}
```

Field Value

The number of items stored in this Deque.
Remarks

Getting the count of items in the Deque takes a small constant amount of time.
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Deque(Of T)>...:::Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

 Gets or sets an item at a particular index in the Deque.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public override sealed T this[int index]
{	get;	set;
}
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Default Property Item ( _
    index As Integer _
) As T
```

**Visual C++**

```cpp
public:
    virtual property T default[int index] {
        T get (int index) override sealed;
        void set (int index, T value) override sealed;
    }
```

### Parameters

**index**

**Int32**

The index of the item to retrieve or change. The front item has index 0, and the back item has index Count-1.

### Return Value

The value at the indicated index.
Remarks

Getting or setting the item at a particular index takes a small constant amount of time, no matter what index is used.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentOutOfRangeException</code></td>
<td>The index is less than zero or greater than or equal to Count.</td>
</tr>
</tbody>
</table>
See Also

Deque(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<Of TKey, TValue> Class

See Also  Members

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

DictionaryBase is a base class that can be used to more easily implement the generic IDictionary<T> and non-generic IDictionary interfaces.

Namespace: Wintellect.PowerCollections
Assembly:  PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
[SerializableAttribute]
public abstract class DictionaryBase<TKey, TValue> : CollectionBase
    ICollection<KeyValuePair<TKey, TValue>>, IEnumerable<KeyValuePair<TKey, TValue>>,
    IDictionary<TKey, TValue>, ICollection, IEnumerable
```

Visual Basic (Declaration)

```vbnet
<SerializableAttribute>
Public MustInherit Class DictionaryBase(Of TKey, TValue) Inherits CollectionBase(Of KeyValuePair(Of TKey, TValue))
    Implements IDictionary(Of TKey, TValue), ICollection(Of KeyValuePair(Of TKey, TValue)), IEnumerable<KeyValuePair(Of TKey, TValue), IDictionary, ICollection, IEnumerable
```

Visual C++

```cpp
[SerializableAttribute]
template<typename TKey, typename TValue>
public ref class DictionaryBase abstract : public CollectionBase
    IDictionary<TKey, TValue>, ICollection<KeyValuePair<TKey, TValue>>, IEnumerable<KeyValuePair<TKey, TValue>>, IDictionary, ICollection, IEnumerable
```
**Type Parameters**

TKey
   The key type of the dictionary.
TValue
   The value type of the dictionary.
Remarks

To use DictionaryBase as a base class, the derived class must override Count, GetEnumerator, TryGetValue, Clear, Remove, and the indexer set accessor.
Inheritance Hierarchy

System..:::Object
   Wintellect.PowerCollections..:::CollectionBase<Of <KeyValuePair<Of<TKey, TValue>>>>
   Wintellect.PowerCollections..:::DictionaryBase<Of <TKey, TValue>>
   Wintellect.PowerCollections..:::OrderedDictionary<Of <TKey, TValue>>
   Wintellect.PowerCollections..:::OrderedDictionary<Of <TKey, TValue>>..:::View
See Also

DictionaryBase<Of <TKey, TValue>> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase(Of TKey, TValue) Members

See Also  Methods  Properties  Constructors  Explicit Interface Implementations

☐ Include Inherited Members  ☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class DictionaryBase<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute>  _
Public MustInherit Class DictionaryBase(Of TKey, TValue)

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>  
public ref class DictionaryBase abstract
Type Parameters

TKey
TValue

The type exposes the following members.
## Protected Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DictionaryBase&lt;OF TKey, TValue&gt;</code>&gt;New</td>
<td></td>
</tr>
</tbody>
</table>
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Overloaded. Provides a read-only view of this dictionary. The returned IDictionary&lt;TKey,TValue&gt; provides a view of the dictionary that prevents modifications to the dictionary. Use the method to provide access to the dictionary without allowing changes. Since the returned object is just a view, changes to the dictionary will be reflected in the view.</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this dictionary. The returned IDictionary&lt;TKey,TValue&gt; provides a view of the dictionary that prevents modifications to the dictionary. Use the method to provide access to the dictionary without allowing changes. Since the returned object is just a view, changes to the dictionary will be reflected in the view.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Clears the dictionary. This method must be overridden in the derived class. (Overrides CollectionBase&lt;(Of (KeyValuePair(Of TKey,TValue&gt;))&gt;&gt;).::Clear().)</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Determines if a dictionary contains a given KeyValuePair. This implementation checks to see if the dictionary contains the given key, and if the value associated with the key is equal to (via object.Equals) the value. (Overrides CollectionBase&lt;(Of (KeyValuePair(Of TKey,TValue&gt;))&gt;&gt;).::Contains(T).)</td>
</tr>
<tr>
<td><strong>ContainsKey</strong></td>
<td>Determines whether a given key is found in the dictionary. Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from CollectionBase&lt;(Of (T)&gt;).)</td>
</tr>
<tr>
<td><strong>ConvertAll(Of TOutput)&gt;</strong></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;(Of (T)&gt;).)</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;(Of (T)&gt;).)</td>
</tr>
<tr>
<td><strong>CountWhere</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from CollectionBase&lt;(Of (T)&gt;).) Determines whether the specified Object is equal to the</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td><code>Equals</code></td>
<td>current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td></td>
<td>Determines if the collection contains any item that</td>
</tr>
<tr>
<td></td>
<td>satisfies the condition defined by predicate.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>CollectionBase(Of&lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><code>Exists</code></td>
<td>Enumerates the items in the collection that satisfy</td>
</tr>
<tr>
<td></td>
<td>the condition defined by predicate.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>CollectionBase(Of&lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><code>FindAll</code></td>
<td>Performs the specified action on each item in this</td>
</tr>
<tr>
<td></td>
<td>collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>CollectionBase(Of&lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td></td>
<td>Must be overridden to enumerate all the members of</td>
</tr>
<tr>
<td></td>
<td>the collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>CollectionBase(Of&lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><code>GetEnumerator</code></td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td></td>
<td><code>GetHashCode()</code> is suitable for use in hashing</td>
</tr>
<tr>
<td></td>
<td>algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>Remove</code></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><code>RemoveAll</code></td>
<td>Removes all the items in the collection that satisfy</td>
</tr>
<tr>
<td></td>
<td>the condition defined by predicate.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>CollectionBase(Of&lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><code>ToArray</code></td>
<td>Creates an array of the correct size, and copies all</td>
</tr>
<tr>
<td></td>
<td>the items in the collection into the array, by calling</td>
</tr>
<tr>
<td></td>
<td>CopyTo.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>CollectionBase(Of&lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Shows the string representation of the dictionary.</td>
</tr>
<tr>
<td></td>
<td>The string representation contains a list of the</td>
</tr>
<tr>
<td></td>
<td>mappings in the dictionary.</td>
</tr>
<tr>
<td></td>
<td>(Overrides <code>CollectionBase(Of&lt;KeyValuePair(Of&lt;TKey, TValue&gt;)&gt;&gt;)</code>&gt;...:<code>ToString()</code>.)</td>
</tr>
<tr>
<td><code>TryGetValue</code></td>
<td>Determines if this dictionary contains a key equal to</td>
</tr>
<tr>
<td></td>
<td>key. If so, the value associated with that key is</td>
</tr>
<tr>
<td></td>
<td>returned through the value parameter. This method</td>
</tr>
<tr>
<td></td>
<td>must be overridden by the</td>
</tr>
</tbody>
</table>
derived class.
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>)</td>
</tr>
</tbody>
</table>
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚒ <strong>Count</strong></td>
<td>Must be overridden to provide the number of items in the collection. (Inherited from <code>CollectionBase&lt;Of&lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td>⚒ <strong>Item</strong></td>
<td>The indexer of the dictionary. This is used to store keys and values and retrieve values from the dictionary. The setter accessor must be overridden in the derived class.</td>
</tr>
<tr>
<td>⚒ <strong>Keys</strong></td>
<td>Returns a collection of the keys in this dictionary.</td>
</tr>
<tr>
<td>⚒ <strong>Values</strong></td>
<td>Returns a collection of the values in this dictionary. The ordering of values in this collection is the same as that in the Keys collection.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection(&lt;(\text{Of}\ &lt;\text{T}&gt;)&gt;;;::;\text{IsReadOnly}</td>
<td>(Inherited from CollectionBase(&lt;(\text{Of}\ &lt;\text{T}&gt;)&gt;).) Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase(&lt;(\text{Of}\ &lt;\text{T}&gt;)&gt;).)</td>
</tr>
<tr>
<td>ICollection;;::;\text{CopyTo}</td>
<td>Indicates whether the collection is synchronized. (Inherited from CollectionBase(&lt;(\text{Of}\ &lt;\text{T}&gt;)&gt;).)</td>
</tr>
<tr>
<td>ICollection;;::;\text{IsSynchronized}</td>
<td>Indicates the synchronization object for this collection. (Inherited from CollectionBase(&lt;(\text{Of}\ &lt;\text{T}&gt;)&gt;).)</td>
</tr>
<tr>
<td>ICollection;;::;\text{SyncRoot}</td>
<td>Adds a key-value pair to the collection. If key or value are not of the expected types, an ArgumentException is thrown. If both key and value are of the expected types, the (overridden) Add method is called with the key and value to add.</td>
</tr>
<tr>
<td>IDictionary;;::;\text{Add}</td>
<td>Clears this dictionary. Calls the (overridden) Clear method.</td>
</tr>
<tr>
<td>IDictionary;;::;\text{Clear}</td>
<td>Determines if this dictionary contains a key equal to key. The dictionary is not changed. Calls the (overridden) ContainsKey method. If key is not of the correct TKey for the dictionary, false is returned.</td>
</tr>
<tr>
<td>IDictionary;;::;\text{Contains}</td>
<td>Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries</td>
</tr>
</tbody>
</table>
are enumerated in the same orders as the (overridden) GetEnumerator method.

**IDictionary...::IsFixedSize**

Returns whether this dictionary is fixed size. This implemented always returns false.

**IDictionary...::IsReadOnly**

Returns if this dictionary is read-only. This implementation always returns false.

**IDictionary...::Item**

Gets or sets the value associated with a given key. When getting a value, if this key is not found in the collection, then null is returned. When setting a value, the value replaces any existing value in the dictionary. If either the key or value are not of the correct type for this dictionary, an ArgumentException is thrown.

**IDictionary...::Keys**

Returns a collection of all the keys in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method. Removes the key (and associated value) from the collection that is equal to the passed in key. If no key in the dictionary is equal to the passed key, the dictionary is unchanged. Calls the (overridden) Remove method. If key is not of the correct TKey for the dictionary, the dictionary is unchanged.

**IDictionary...::Remove**

Returns a collection of all the values in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method.

**IEnumerable...::GetEnumerator**

Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method.
See Also

DictionaryBase(Of TKey, TValue)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<(Of <TKey, TValue>)> Constructor

See Also

Visual Basic (Declaration) ▶ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

protected DictionaryBase()

Visual Basic (Declaration)

Protected Sub New

Visual C++

protected:
DictionaryBase ()
See Also

DictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<(Of <TKey, TValue>)> Methods

See Also

☐ Include Inherited Members ☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class DictionaryBase<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute> _
Public MustInherit Class DictionaryBase(Of TKey, TValue)

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class DictionaryBase abstract
Type Parameters

TKey
TValue

The type exposes the following methods.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Overloaded. Provides a read-only view of this dictionary. The returned IDictionary&lt;TKey,TValue&gt; provides a view of the dictionary that prevents modifications to the dictionary. Use the method to provide access to the dictionary without allowing changes. Since the returned object is just a view, changes to the dictionary will be reflected in the view.</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this dictionary. The returned IDictionary&lt;TKey,TValue&gt; provides a view of the dictionary that prevents modifications to the dictionary. Use the method to provide access to the dictionary without allowing changes. Since the returned object is just a view, changes to the dictionary will be reflected in the view.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Clears the dictionary. This method must be overridden in the derived class. (Overrides CollectionBase&lt;(Of &lt;KeyValuePair(Of &lt;TKey, TValue&gt;&gt;)&gt;)...::Clear()().)</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Determines if a dictionary contains a given KeyValuePair. This implementation checks to see if the dictionary contains the given key, and if the value associated with the key is equal to (via object.Equals) the value. (Overrides CollectionBase&lt;(Of &lt;KeyValuePair(Of &lt;TKey, TValue&gt;&gt;)&gt;)...::Contains(T).)</td>
</tr>
<tr>
<td><strong>ContainsKey</strong></td>
<td>Determines whether a given key is found in the dictionary.</td>
</tr>
<tr>
<td><strong>ConvertAll(Of TOutput)&gt;</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from CollectionBase&lt;(Of &lt;T&gt;)).</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;(Of &lt;T&gt;)).</td>
</tr>
<tr>
<td><strong>CountWhere</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from CollectionBase&lt;(Of &lt;T&gt;)).</td>
</tr>
<tr>
<td><strong>Determines whether the specified Object is equal to the</strong></td>
<td></td>
</tr>
</tbody>
</table>
- **Equals**
  current **Object**.
  (Inherited from **Object**.)
  Determines if the collection contains any item that satisfies the condition defined by predicate.
  (Inherited from **CollectionBase(Of <T>)**.)

- **Exists**
  Enumerates the items in the collection that satisfy the condition defined by predicate.
  (Inherited from **CollectionBase(Of <T>)**.)

- **FindAll**
  Performs the specified action on each item in this collection.
  (Inherited from **CollectionBase(Of <T>)**.)
  Must be overridden to enumerate all the members of the collection.
  (Inherited from **CollectionBase(Of <T>)**.)

- **GetEnumerator**
  Serves as a hash function for a particular type.
  (Inherited from **CollectionBase(Of <T>)**.)

- **GetHashCode**
  **GetHashCode()** is suitable for use in hashing algorithms and data structures like a hash table.
  (Inherited from **Object**.)

- **GetType**
  Gets the **Type** of the current instance.
  (Inherited from **Object**.)

- **Remove**
  Overloaded.
  Removes all the items in the collection that satisfy the condition defined by predicate.
  (Inherited from **CollectionBase(Of <T>)**.)

- **RemoveAll**
  Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.
  (Inherited from **CollectionBase(Of <T>)**.)

- **ToArray**
  Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.
  (Overrides **CollectionBase(Of <KeyValuePair(Of TKey, TValue>)>>)>...:**ToString()**.)

- **ToString**
  Determines if all of the items in the collection satisfy the condition defined by predicate.
  (Inherited from **CollectionBase(Of <T>)**.)

- **TryGetValue**
  Determines if this dictionary contains a key equal to key. If so, the value associated with that key is returned through the value parameter. This method must be overridden by the
derived class.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection::CopyTo</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>CollectionBase&lt;T&gt;</code>. )</td>
</tr>
<tr>
<td><code>IDictionary::Add</code></td>
<td>Adds a key-value pair to the collection. If key or value are not of the expected types, an ArgumentException is thrown. If both key and value are of the expected types, the (overridden) Add method is called with the key and value to add.</td>
</tr>
<tr>
<td><code>IDictionary::Clear</code></td>
<td>Clears this dictionary. Calls the (overridden) Clear method.</td>
</tr>
<tr>
<td><code>IDictionary::Contains</code></td>
<td>Determines if this dictionary contains a key equal to key. The dictionary is not changed. Calls the (overridden) ContainsKey method. If key is not of the correct TKey for the dictionary, false is returned.</td>
</tr>
<tr>
<td><code>IDictionary::GetEnumerator</code></td>
<td>Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method. Removes the key (and associated value) from the collection that is equal to the passed in key. If no key in the dictionary is equal to the passed key, the dictionary is unchanged. Calls the (overridden) Remove method. If key is not of the correct TKey for the dictionary, the dictionary is unchanged.</td>
</tr>
<tr>
<td><code>IDictionary::Remove</code></td>
<td></td>
</tr>
</tbody>
</table>
Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method.
See Also

DictionaryBase<(Of<TKey, TValue>)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<,(Of <TKey, TValue>)>...::Add Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add(KeyValuePair(Of &lt;TKey, TValue&gt;))</td>
<td>Adds a key-value pair to the collection. This implementation calls the Add method with the Key and Value from the item. ( Overrides <code>CollectionBase&lt;Of &lt;KeyValuePair&lt;Of &lt;TKey, TValue&gt;&gt;&gt;&gt;::Add(T)</code> )</td>
</tr>
<tr>
<td>Add(TKey, TValue)</td>
<td>Adds a new key-value pair to the dictionary.</td>
</tr>
</tbody>
</table>
See Also

DictionaryBase(Of TKey, TValue)> Class
DictionaryBase(Of TKey, TValue)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<Of <TKey, TValue>>::Add Method (KeyValuePair<Of <TKey, TValue>>)

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a key-value pair to the collection. This implementation calls the Add method with the Key and Value from the item.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public override void Add(
    KeyValuePair<TKey, TValue> item
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides Sub Add (_
    item As KeyValuePair(Of TKey, TValue) _
)
```

**Visual C++**

```cpp
public:
    virtual void Add (
        KeyValuePair<TKey, TValue> item
    ) override
```

### Parameters

`item`  
`KeyValuePair(Of<TKey, TValue>)`  
A `KeyValuePair` contains the Key and Value to add.
See Also

DictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<
(Of <TKey, TValue>)>

Add Method (TKey, TValue)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a new key-value pair to the dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public virtual void Add(
    TKey key,
    TValue value
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Sub Add ( _
    key As TKey, _
    value As TValue _
)
```

**Visual C++**

```cpp
public:
    virtual void Add ( 
        TKey key,
        TValue value
    )
```

### Parameters

**key**

TKey

Key to add.

**value**

TValue

Value to associated with the key.
Remarks

The default implementation of this method checks to see if the key already exists using `ContainsKey`, then calls the indexer setter if the key doesn't already exist.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...::ArgumentException</td>
<td>key is already present in the dictionary</td>
</tr>
</tbody>
</table>
See Also

DictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<Of <TKey, TValue>> AsReadOnly Method

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Provides a read-only view of this dictionary. The returned IDictionary<TKey,TValue> provides a view of the dictionary that prevents modifications to the dictionary. Use the method to provide access to the dictionary without allowing changes. Since the returned object is just a view, changes to the dictionary will be reflected in the view.

Namespace: Wintellect.PowerCollections  
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public virtual IDictionary<TKey, TValue> AsReadOnly()
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function AsReadOnly As IDictionary(Of TKey, TValue)
```

**Visual C++**

```cpp
public:
virtual IDictionary<TKey, TValue>^ AsReadOnly ()
```

### Return Value

An IDictionary<TKey, TValue> that provides read-only access to the dictionary.
See Also

DictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<(Of <TKey, TValue>)>::Clear Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Clears the dictionary. This method must be overridden in the derived class.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public abstract void Clear()
```

**Visual Basic (Declaration)**

```vbnet
Public MustOverride Sub Clear
```

**Visual C++**

```cpp
public:
    virtual void Clear () abstract override
```
See Also

DictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<Of <TKey, TValue>>::Contains Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if a dictionary contains a given KeyValuePair. This implementation checks to see if the dictionary contains the given key, and if the value associated with the key is equal to (via object.Equals) the value.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public override bool Contains(
    KeyValuePair<TKey, TValue> item
)
```

Visual Basic (Declaration)

```vbnet
Public Overrides Function Contains ( _
    item As KeyValuePair(Of TKey, TValue) _
) As Boolean
```

Visual C++

```cpp
public:
    virtual bool Contains ( 
        KeyValuePair<TKey, TValue> item
    ) override
```

Parameters

item

    `KeyValuePair<(Of <TKey, TValue>)>`,
    A KeyValuePair containing the Key and Value to check for.

Return Value
See Also

DictionaryBase<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines whether a given key is found in the dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public virtual bool ContainsKey(TKey key)
```

Visual Basic (Declaration)

```vbnet
Public Overridable Function ContainsKey (key As TKey) As Boolean
```

Visual C++

```cpp
public:
    virtual bool ContainsKey ( TKey key
```

Parameters

key

TKey

Key to look for in the dictionary.

Return Value

True if the key is present in the dictionary.
Remarks

The default implementation simply calls TryGetValue and returns what it returns.
See Also

DictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<Of <TKey, TValue>>::Remove Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove(KeyValuePair(Of TKey, TValue))</td>
<td>Determines if a dictionary contains a given KeyValuePair, and if so, removes it. This implementation checks to see if the dictionary contains the given key, and if the value associated with the key is equal to (via object.Equals) the value. If so, the key-value pair is removed. (Overrides CollectionBase(Of KeyValuePair(Of TKey, TValue))::.Remove(T).)</td>
</tr>
<tr>
<td>Remove(TKey)</td>
<td>Removes a key from the dictionary. This method must be overridden in the derived class.</td>
</tr>
</tbody>
</table>
See Also

DictionaryBase(Of TKey, TValue)> Class
DictionaryBase(Of TKey, TValue)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<Of <TKey, TValue>>...::Remove Method (KeyValuePair<Of <TKey, TValue>>)

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if a dictionary contains a given KeyValuePair, and if so, removes it. This implementation checks to see if the dictionary contains the given key, and if the value associated with the key is equal to (via object.Equals) the value. If so, the key-value pair is removed.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public override bool Remove(
    KeyValuePair<TKey, TValue> item
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides Function Remove ( _
    item As KeyValuePair(Of TKey, TValue) _
) As Boolean
```

**Visual C++**

```cpp
public:
    virtual bool Remove ( _
        KeyValuePair<TKey, TValue> item
    ) override
```

**Parameters**

- **item**
  ```cpp
  KeyValuePair(Of TKey, TValue)
  ```
  A `KeyValuePair` containing the Key and Value to check for.

**Return Value**

- True if the item was found and removed. False otherwise.
See Also

DictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<(Of <TKey, TValue)>>: Remove Method (TKey)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes a key from the dictionary. This method must be overridden in the derived class.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public abstract bool Remove( TKey key )
```

**Visual Basic (Declaration)**

```vbnet
Public MustOverride Function Remove ( _
    key As TKey _
) As Boolean
```

**Visual C++**

```cpp
public:
    virtual bool Remove ( _
        TKey key
    ) abstract
```

**Parameters**

key

    TKey
    Key to remove from the dictionary.

**Return Value**

True if the key was found, false otherwise.
See Also

DictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<Of <TKey, TValue>>::.System.Collections.IDictionary.Add

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a key-value pair to the collection. If key or value are not of the expected types, an ArgumentException is thrown. If both key and value are of the expected types, the (overridden) Add method is called with the key and value to add.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
# Syntax

**C#**

```csharp
private void IDictionary.Add(
    Object key,
    Object value
)
```

**Visual Basic (Declaration)**

```vbnet
Private Sub System.Collections.IDictionary.Add (
    key As Object, _
    value As Object _
) Implements IDictionary.Add
```

**Visual C++**

```cpp
private:
virtual void System.Collections.IDictionary.Add (  
    Object^ key,  
    Object^ value
) sealed = IDictionary::Add
```

## Parameters

**key**

- **Object**
  - Key to add to the dictionary.

**value**

- **Object**
  - Value to add to the dictionary.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentException</code></td>
<td>key or value are not of the expected type for this dictionary.</td>
</tr>
</tbody>
</table>
See Also

DictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<
(Of <TKey, TValue>)>...::System.Collections.IDictionary.Clear

**Method**

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Clears this dictionary. Calls the (overridden) Clear method.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

private void IDictionary.Clear()

**Visual Basic (Declaration)**

Private Sub System.Collections.IDictionary.Clear Implements IDictionary

**Visual C++**

private:
virtual void System.Collections.IDictionary.Clear() sealed = IDictionary
See Also

DictionaryBase<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<
(Of <TKey,
TValue>)>::System.Collections.IDictionary.Contains Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this dictionary contains a key equal to key. The dictionary is not changed. Calls the (overridden) ContainsKey method. If key is not of the correct TKey for the dictionary, false is returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
# Syntax

**C#**

```csharp
private bool IDictionary.Contains(
    Object key
)
```

**Visual Basic (Declaration)**

```vbnet
Private Function System.Collections.IDictionary.Contains ( _
    key As Object _
) As Boolean Implements IDictionary.Contains
```

**Visual C++**

```cpp
private:
virtual bool System.Collections.IDictionary.Contains ( _
    Object^ key
)
sealed = IDictionary::Contains
```

## Parameters

**key**

`Object`

The key to search for.

## Return Value

True if the dictionary contains key. False if the dictionary does not contain key.
See Also

**DictionaryBase(Of TKey, TValue)>** Class
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
DictionaryBase<
(Of <TKey,
TValue>)>...::System.Collections.IDictionary.GetEnumerator Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
private IDictionaryEnumerator IDictionary.GetEnumerator()
```

**Visual Basic (Declaration)**

```vbnet
Private Function System.Collections.IDictionary.GetEnumerator As IDictionaryEnumerator
```

**Visual C++**

```cpp
private:
virtual IDictionaryEnumerator^ System.Collections.IDictionary.GetEnumerator();
```

### Return Value

An enumerator for enumerating all the elements in the OrderedDictionary.
See Also

DictionaryBase<Of<TKey, TValue>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<Of TKey, TValue>::System.Collections.IDictionary.Remove Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes the key (and associated value) from the collection that is equal to the passed in key. If no key in the dictionary is equal to the passed key, the dictionary is unchanged. Calls the (overridden) Remove method. If key is not of the correct TKey for the dictionary, the dictionary is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
private void IDictionary.Remove(
    Object key
)
```

**Visual Basic (Declaration)**

```vbnet
Private Sub System.Collections.IDictionary.Remove ( _
    key As Object _
) Implements IDictionary.Remove
```

**Visual C++**

```cpp
private:
virtual void System.Collections.IDictionary.Remove ( _
    Object^ key
) sealed = IDictionary::Remove
```

**Parameters**

`key`  
*Object*  
The key to remove.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::ArgumentException</code></td>
<td>key could not be converted to TKey.</td>
</tr>
</tbody>
</table>
See Also

DictionaryBase<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<Of <TKey,
 TValue>>:..:.System.Collections.IEnumerable.GetEnumerator Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
private IEnumerable IEnumerable.GetEnumerator()
```

**Visual Basic (Declaration)**

Private Function System.Collections.IEnumerable.GetEnumerator As IEr

**Visual C++**

```cpp
private:
virtual IEnumerator^ System.Collections.IEnumerable.GetEnumerator ()
```

**Return Value**

An enumerator for enumerating all the elements in the OrderedDictionary.
See Also

DictionaryBase(Of TKey, TValue)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<
(OfYear Key, TValue)>
::ToString Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public override string ToString()
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides Function ToString As String
```

**Visual C++**

```cpp
public:
    virtual String^ ToString () override
```

**ReturnValue**

The string representation of the dictionary.
See Also

DictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase(Of TKey, TValue)\:::TryGetValue Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this dictionary contains a key equal to `key`. If so, the value associated with that key is returned through the value parameter. This method must be overridden by the derived class.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public abstract bool TryGetValue(
    TKey key,
    out TValue value
)

Visual Basic (Declaration)

Public MustOverride Function TryGetValue ( _
    key As TKey, _
    <OutAttribute> ByRef value As TValue _
) As Boolean

Visual C++

public:
virtual bool TryGetValue ( 
    TKey key,
    [OutAttribute] TValue% value
) abstract

Parameters

key
    TKey
    The key to search for.

value
    TValue%
    Returns the value associated with key, if true was returned.

Return Value

True if the dictionary contains key. False if the dictionary does not contain key.
See Also

DictionaryBase<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<Of <TKey, TValue>> Properties

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class DictionaryBase<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute> _
Public MustInherit Class DictionaryBase(Of TKey, TValue)

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class DictionaryBase abstract
Type Parameters

TKey
TValue

The type exposes the following properties.
# Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✅ <strong>Count</strong></td>
<td>Must be overridden to provide the number of items in the collection. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td>✅ <strong>Item</strong></td>
<td>The indexer of the dictionary. This is used to store keys and values and retrieve values from the dictionary. The setter accessor must be overridden in the derived class.</td>
</tr>
<tr>
<td>✅ <strong>Keys</strong></td>
<td>Returns a collection of the keys in this dictionary.</td>
</tr>
<tr>
<td>✅ <strong>Values</strong></td>
<td>Returns a collection of the values in this dictionary. The ordering of values in this collection is the same as that in the Keys collection.</td>
</tr>
</tbody>
</table>
# Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection&lt;&lt;T&gt;&gt;::IsReadOnly</td>
<td>(Inherited from CollectionBase&lt;&lt;T&gt;&gt;.) Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td>ICollection::IsSynchronized</td>
<td>(Inherited from CollectionBase&lt;&lt;T&gt;&gt;.) Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td>ICollection::SyncRoot</td>
<td>(Inherited from CollectionBase&lt;&lt;T&gt;&gt;.) Returns whether this dictionary is fixed size. This implemented always returns false.</td>
</tr>
<tr>
<td>IDictionary::IsFixedSize</td>
<td>Returns if this dictionary is read-only. This implementation always returns false. Gets or sets the value associated with a given key. When getting a value, if this key is not found in the collection, then null is returned. When setting a value, the value replaces any existing value in the dictionary. If either the key or value are not of the correct type for this dictionary, an ArgumentException is thrown.</td>
</tr>
<tr>
<td>IDictionary::Item</td>
<td>Returns a collection of all the keys in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method.</td>
</tr>
<tr>
<td>IDictionary::Keys</td>
<td>Returns a collection of all the values in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method.</td>
</tr>
</tbody>
</table>
See Also

DictionaryBase<(<TKey, TValue>)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase(Of TKey, TValue)>...::Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

The indexer of the dictionary. This is used to store keys and values and retrieve values from the dictionary. The setter accessor must be overridden in the derived class.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual TValue this[
    TKey key
]{ get; set;}

Visual Basic (Declaration)

Public Overridable Default Property Item ( _
            key As TKey _
) As TValue

Visual C++

public:
virtual property TValue default[TKey key] { 
    TValue get (TKey key);
    void set (TKey key, TValue value);
}

Parameters

key
    TKey
    Key to find in the dictionary.

Return Value

The value associated with the key.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.Collections.Generic.KeyNotFoundException</code></td>
<td>Thrown from the get accessor if the key was not found in the dictionary.</td>
</tr>
</tbody>
</table>
See Also

DictionaryBase<Of<TKey, TValue>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<(Of <TKey, TValue>)>...::Keys Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a collection of the keys in this dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual ICollection<TKey> Keys { get; }

Visual Basic (Declaration)

Public Overridable ReadOnly Property Keys As ICollection(Of TKey)

Visual C++

public:
virtual property ICollection<TKey>^ Keys {
    ICollection<TKey>^ get ();
}

Field Value

A read-only collection of the keys in this dictionary.
See Also

DictionaryBase<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<Of<TKey, TValue>>...::System.Collections.IDictionary.IsFixedSize Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns whether this dictionary is fixed size. This implemented always returns false.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private bool IDictionary.IsFixedSize{ get; }

Visual Basic (Declaration)

Private ReadOnly Property System.Collections.IDictionary.IsFixedSize

Visual C++

private:
virtual property bool System.Collections.IDictionary.IsFixedSize {
    bool get () sealed = IDictionary::IsFixedSize::get;
}

Field Value

Always returns false.
See Also

DictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<Of <TKey, TValue>>...: System.Collections.IDictionary.IsReadOnly Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns if this dictionary is read-only. This implementation always returns false.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
private bool IDictionary.IsReadOnly { get; }
```

**Visual Basic (Declaration)**

```vbnet
Private Readonly Property System.Collections.IDictionary.IsReadOnly
```

**Visual C++**

```cpp
private:
virtual property bool System.Collections.IDictionary.IsReadOnly {
    bool get () sealed = IDictionary::IsReadOnly::get;
}
```

**Field Value**

Always returns false.
See Also

DictionaryBase<(Of <TKey, TValue>)) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Gets or sets the value associated with a given key. When getting a value, if this key is not found in the collection, then null is returned. When setting a value, the value replaces any existing value in the dictionary. If either the key or value are not of the correct type for this dictionary, an ArgumentException is thrown.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

**C#**

```csharp
private Object IDictionary.Item[
  Object key
]{ get; set;}
```

**Visual Basic (Declaration)**

```vbnet
Private Property System.Collections.IDictionary.Item ( _
  key As Object _
) As Object Implements IDictionary.Item
```

**Visual C++**

```c++
private:
  virtual Object^ System.Collections.IDictionary.Item[Object/
    Object^ get (Object^ key) sealed = IDictionary::Item::get;
  void set (Object^ key, Object^ value) sealed = IDictionary::

```

### Parameters

**key**

Object

### Field Value

The value associated with the key, or null if the key was not present.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentException</td>
<td>key could not be converted to TKey, or value could not be converted to TValue.</td>
</tr>
</tbody>
</table>
See Also

DictionaryBase<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<
(Of <TKey, TValue>)>...::System.Collections.IDictionary.Keys
Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a collection of all the keys in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private ICollection IDictionary.Keys { get; }

Visual Basic (Declaration)

Private ReadOnly Property System.Collections.IDictionary.Keys As ICollection

Visual C++

private:
virtual property ICollection^ System.Collections.IDictionary.Keys { 
    ICollection^ get () sealed = IDictionary::Keys::get; } 

Field Value

The collection of keys.
See Also

DictionaryBase<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Returns a collection of all the values in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

private ICollection IDictionary.Values{ get; }

**Visual Basic (Declaration)**

Private ReadOnly Property System.Collections.IDictionary.Values As }

**Visual C++**

private:
virtual property ICollection^ System.Collections.IDictionary.Values
ICollection^ get () sealed = IDictionary::Values::get;

**Field Value**

The collection of values.
See Also

DictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
DictionaryBase<Of <TKey, TValue>>::.Values Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a collection of the values in this dictionary. The ordering of values in this collection is the same as that in the Keys collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public virtual ICollection<TValue> Values{ get; }
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable ReadOnly Property Values As ICollection(Of TValue)
```

**Visual C++**

```cpp
public:
virtual property ICollection<TValue>^ Values {
ICollection<TValue>^ get ()
}
```

### Field Value

A read-only collection of the values in this dictionary.
See Also

DictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase< (Of <T> ) > Class

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

ListBase is an abstract class that can be used as a base class for a read-write collection that needs to implement the generic IList<T> and non-generic IList collections. The derived class needs to override the following methods: Count, Clear, Insert, RemoveAt, and the indexer. The implementation of all the other methods in IList<T> and IList are handled by ListBase.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

[SerializableAttribute]
public abstract class ListBase<T> : CollectionBase<T>, IList<T>, ICollection<T>, IEnumerable<T>, IList, ICollection, IEnumerable

Visual Basic (Declaration)

<SerializableAttribute> _
Public MustInherit Class ListBase(Of T) _
    Inherits CollectionBase(Of T) _
    Implements IList(Of T), ICollection(Of T), _
    IEnumerable(Of T), IList, ICollection, IEnumerable

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class ListBase abstract : public CollectionBase<T>, IList<T>, ICollection<T>, IEnumerable<T>, IList, ICollection, IEnumerable
Type Parameters

T
Inheritance Hierarchy

System...::Object
Wintellect.PowerCollections...::CollectionBase<Of <T>>
Wintellect.PowerCollections...::ListBase<Of <T>>
Wintellect.PowerCollections...::BigList<Of <T>>
Wintellect.PowerCollections...::Deque<Of <T>>
See Also

ListBase<(Of <T>)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<(Of <T>)> Members

See Also  Methods  Properties  Constructors  Explicit Interface Implementations

☐  Include Inherited Members  ☐  Include Protected Members
☐  .NET Compact Framework Members Only
☐  XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class ListBase<T>

Visual Basic (Declaration)

<SerializableAttribute>
Public MustInherit Class ListBase(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class ListBase abstract
Type Parameters

T

The type exposes the following members.
## Protected Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListBase&lt;T&gt;</td>
<td>New</td>
</tr>
</tbody>
</table>
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds an item to the end of the list. This method is equivalent to calling: &lt;br&gt;&lt;br&gt;<code>Insert(Count, item)</code>&lt;br&gt;(Overrides <code>CollectionBase&lt;Of &lt;T&gt;&gt;...::Add(T).</code>) Provides a read-only view of this list. The returned IList&lt;T&gt; provides a view of the list that prevents modifications to the list. Use the method to provide access to the list without allowing changes. Since the returned object is just a view, changes to the list will be reflected in the view. This method must be overridden by the derived class to empty the list of all items. (Overrides <code>CollectionBase&lt;Of &lt;T&gt;&gt;...::Clear()</code>) Determines if the list contains any item that compares equal to item. The implementation simply checks whether <code>IndexOf(item)</code> returns a non-negative value. (Overides <code>CollectionBase&lt;Of &lt;T&gt;&gt;...::Contains(T).</code>) Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>).</td>
</tr>
</tbody>
</table>
- **Exists**
  the condition defined by predicate.
  (Inherited from `CollectionBase<Of <T>>`.)

- **Find**
  Finds the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned.

- **FindAll**
  Enumerates the items in the collection that satisfy the condition defined by predicate.
  (Inherited from `CollectionBase<Of <T>>`.)

- **FindIndex**
  Overloaded.
  Finds the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned.

- **FindLast**
  Overloaded.

- **ForEach**
  Performs the specified action on each item in this collection.
  (Inherited from `CollectionBase<Of <T>>`.)
  Enumerates all of the items in the list, in order. The item at index 0 is enumerated first, then the item at index 1, and so on.

- **GetEnumerator**
  (Overrides `CollectionBase<Of <T>>::GetEnumerator`.)
  Serves as a hash function for a particular type.

- **GetHashCode**
  `GetHashCode()` is suitable for use in hashing algorithms and data structures like a hash table.
  (Inherited from `Object`.)

- **GetType**
  Gets the `Type` of the current instance.
  (Inherited from `Object`.)

- **IndexOf**
  Overloaded.

- **Insert**
  This method must be overridden by the derived class to insert a new item at the given index.

- **LastIndexOf**
  Overloaded.

- **Range**
  Returns a view onto a sub-range of this list. Items are not copied; the returned `IList<T>` is simply a different view onto the same underlying items. Changes to this list are reflected in the view, and vice versa. Insertions and deletions in the view change the size of the view, but insertions and deletions in the underlying list do not.
- **Remove**
  Searches the list for the first item that compares equal to item. If one is found, it is removed. Otherwise, the list is unchanged.
  (Overrides `CollectionBase<Of <T>>::Remove(T)`.)

- **RemoveAll**
  Removes all the items in the collection that satisfy the condition defined by predicate.
  (Inherited from `CollectionBase<Of <T>>`.)

- **RemoveAt**
  This method must be overridden by the derived class to remove the item at the given index.

- **ToArray**
  Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.
  (Inherited from `CollectionBase<Of <T>>`.)

- **ToString**
  Shows the string representation of the collection. The string representation contains a list of the items in the collection.
  (Inherited from `CollectionBase<Of <T>>`.)

- **TrueForAll**
  Determines if all of the items in the collection satisfy the condition defined by predicate.
  (Inherited from `CollectionBase<Of <T>>`.)

- **TryFind**
  Finds the first item in the list that satisfies the condition defined by predicate.

- **TryFindLast**
  Finds the last item in the list that satisfies the condition defined by predicate.
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>The property must be overridden by the derived class to return the number of items in the list. (Overrides <code>CollectionBase&lt;Of &lt;T&gt;&gt;::Count</code>.)</td>
</tr>
<tr>
<td>Item</td>
<td>The indexer must be overridden by the derived class to get and set values of the list at a particular index.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| ICollection\<(Of `T">\,:::IsReadOnly | (Inherited from `CollectionBase\<(Of `T">\).)
  Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.
| ICollection\,:::CopyTo | Indicates whether the collection is synchronized.
| ICollection\,:::IsSynchronized | Indicates the synchronization object for this collection.
| ICollection\,:::SyncRoot | Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator\<`T> that was overridden by the derived classes to enumerate the members of the collection.
| IEnumerable\,:::GetEnumerator | Adds an item to the end of the list. This method is equivalent to calling:  

  ```csharp
  Insert(Count, item)
  ```
| IList\,:::Add | Removes all the items from the list, resulting in an empty list.
<p>| IList,:::Clear |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Contains</code></td>
<td>Determines if the list contains any item that compares equal to <code>value</code>.</td>
</tr>
<tr>
<td><code>IndexOf</code></td>
<td>Find the first occurrence of an item equal to <code>value</code> in the list, and returns the index of that item.</td>
</tr>
<tr>
<td><code>Insert</code></td>
<td>Insert a new item at the given index.</td>
</tr>
<tr>
<td><code>IsFixedSize</code></td>
<td>Returns whether the list is a fixed size. This implementation always returns false.</td>
</tr>
<tr>
<td><code>IsReadOnly</code></td>
<td>Returns whether the list is read only. This implementation returns the value from <code>ICollection&lt;T&gt;.IsReadOnly</code>, which is by default, false.</td>
</tr>
<tr>
<td><code>Item</code></td>
<td>Gets or sets the value at a particular index in the list.</td>
</tr>
<tr>
<td><code>Remove</code></td>
<td>Searches the list for the first item that compares equal to <code>value</code>. If one is found, it is removed. Otherwise, the list is unchanged.</td>
</tr>
<tr>
<td><code>RemoveAt</code></td>
<td>Removes the item at the given index.</td>
</tr>
</tbody>
</table>
See Also

ListBase(Of T)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<(Of <T>)> Constructor

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JSExpand
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

protected ListBase()

**Visual Basic (Declaration)**

Protected Sub New

**Visual C++**

protected:
ListBase ();
See Also

ListBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>> Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class ListBase<T>

Visual Basic (Declaration)

<SerializableAttribute>
Public MustInherit Class ListBase(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class ListBase abstract
Type Parameters

T

The type exposes the following methods.
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Adds an item to the end of the list. This method is equivalent to calling:</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this list. The returned IList&lt;T&gt; provides a view of the list that prevents modifications to the list. Use the method to provide access to the list without allowing changes. Since the returned object is just a view, changes to the list will be reflected in the view. This method must be overridden by the derived class to empty the list of all items.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Determines if the list contains any item that compares equal to item. The implementation simply checks whether IndexOf(item) returns a non-negative value.</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>CountWhere</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
</tbody>
</table>

(Overrides `CollectionBase<Of <T>>...::Add(T).`)

(Overrides `CollectionBase<Of <T>>...::Clear()`.)

(Overrides `CollectionBase<Of <T>>...::Contains(T).`)

(Inherited from `CollectionBase<Of <T>>`).

(Inherited from Object.)

Determine if the collection contains any item that satisfi...
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| Exists                         | the condition defined by predicate. (Inherited from `CollectionBase<Of <T>>`.)
| Find                           | Finds the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned. |
| FindAll                        | Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from `CollectionBase<Of <T>>`.)
| FindIndex                      | Overloaded.                                                                 |
| FindLast                       | Finds the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned. |
| FindLastIndex                  | Overloaded.                                                                 |
| ForEach                        | Performs the specified action on each item in this collection. (Inherited from `CollectionBase<Of <T>>`). |
| GetEnumerator                 | Enumerates all of the items in the list, in order. The item at index 0 is enumerated first, then the item at index 1, and so on. (Overrides `CollectionBase<Of <T>>::GetEnumerator()`.) |
| GetHashCode                   | Serves as a hash function for a particular type. `GetHashCode()` is suitable for use in hashing algorithms and data structures like a hash table. (Inherited from `Object`.) |
| GetType                        | Gets the Type of the current instance. (Inherited from `Object`.)            |
| IndexOf                        | Overloaded.                                                                 |
| Insert                         | This method must be overridden by the derived class to insert a new item at the given index. |
| LastIndexOf                   | Overloaded.                                                                 |
| Range                          | Returns a view onto a sub-range of this list. Items are not copied; the returned `IList<T>` is simply a different view onto the same underlying items. Changes to this list are reflected in the view, and vice versa. Insertions and deletions in the view change the size of the view, but insertions and deletions in the underlying list do not. |
- **Remove**
  Searches the list for the first item that compares equal to item. If one is found, it is removed. Otherwise, the list is unchanged.
  (Overrides `CollectionBase<Of <T>>::Remove(T)`.)

- **RemoveAll**
  Removes all the items in the collection that satisfy the condition defined by predicate.
  (Inherited from `CollectionBase<Of <T>>`.)

- **RemoveAt**
  This method must be overridden by the derived class to remove the item at the given index.

- **ToArray**
  Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.
  (Inherited from `CollectionBase<Of <T>>`.)
  Shows the string representation of the collection. The string representation contains a list of the items in the collection.

- **ToString**
  Contained collections (except string) are expanded recursively.
  (Inherited from `CollectionBase<Of <T>>`.)

- **TrueForAll**
  Determines if all of the items in the collection satisfy the condition defined by predicate.
  (Inherited from `CollectionBase<Of <T>>`.)

- **TryFind**
  Finds the first item in the list that satisfies the condition defined by predicate.

- **TryFindLast**
  Finds the last item in the list that satisfies the condition defined by predicate.
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection::CopyTo</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>)</td>
</tr>
<tr>
<td><code>IEnumerable::GetEnumerator</code></td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>)</td>
</tr>
<tr>
<td><code>IList::Add</code></td>
<td>Adds an item to the end of the list. This method is equivalent to calling: <code>Insert(Count, item)</code></td>
</tr>
<tr>
<td><code>IList::Clear</code></td>
<td>Removes all the items from the list, resulting in an empty list.</td>
</tr>
<tr>
<td><code>IList::Contains</code></td>
<td>Determines if the list contains any item that compares equal to value.</td>
</tr>
<tr>
<td><code>IList::IndexOf</code></td>
<td>Find the first occurrence of an item equal to value in the list, and returns the index of that item.</td>
</tr>
<tr>
<td><code>IList::Insert</code></td>
<td>Insert a new item at the given index.</td>
</tr>
<tr>
<td><code>IList::Remove</code></td>
<td>Searches the list for the first item that compares equal to value. If one is found, it is removed. Otherwise, the list is unchanged.</td>
</tr>
</tbody>
</table>
IList::RemoveAt

Removes the item at the given index.
See Also

ListBase<Of <T>>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::Add Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds an item to the end of the list. This method is equivalent to calling:

Code

Insert(Count, item)

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public override void Add(
    T item
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides Sub Add ( 
    item As T 
)
```

**Visual C++**

```cpp
public:
virtual void Add ( 
    T item
) override
```

**Parameters**

**item**

```
T
The item to add to the list.
```
See Also

ListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase(Of T)> AsReadOnly Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Provides a read-only view of this list. The returned IList<T> provides a view of the list that prevents modifications to the list. Use the method to provide access to the list without allowing changes. Since the returned object is just a view, changes to the list will be reflected in the view.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual IList<T> AsReadOnly()

Visual Basic (Declaration)

Public Overridable Function AsReadOnly As IList(Of T)

Visual C++

public:
virtual IList<T>^ AsReadOnly ()

Return Value

An IList<T> that provides read-only access to the list.
See Also

[ListBase<OF <T>> Class]
[Wintellect.PowerCollections Namespace]

Send comments about this topic to Microsoft.
ListBase<(Of <T>)>::Clear Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

This method must be overridden by the derived class to empty the list of all items.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public abstract void Clear()

Visual Basic (Declaration)

Public MustOverride Sub Clear

Visual C++

public:
virtual void Clear () abstract override
See Also

[ListBase<Of <T>>] Class
[Wintellect.PowerCollections] Namespace

Send comments about this topic to Microsoft.
ListBase<(Of <T>))...:::Contains Method

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

Determines if the list contains any item that compares equal to item. The implementation simply checks whether IndexOf(item) returns a non-negative value.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public override bool Contains(T item)
```

Visual Basic (Declaration)

```vbnet
Public Overrides Function Contains (item As T) As Boolean
```

Visual C++

```cpp
public:
virtual bool Contains(T item)
override
```

Parameters

item

T

The item to search for.

Return Value

True if the list contains an item that compares equal to item.
Equality in the list is determined by the default sense of equality for T. If T implements IComparable<T>, the Equals method of that interface is used to determine equality. Otherwise, Object.Equals is used to determine equality.
See Also

- ListBase<(Of `<T>`)> Class
- Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<(Of <T>)>...::CopyTo Method
See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyTo(array&lt;T&gt;[], ())</td>
<td>Copies all the items in the list, in order, to array, starting at index 0.</td>
</tr>
<tr>
<td>CopyTo(array&lt;T&gt;[], Int32)</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>CopyTo(Int32, array&lt;T&gt;[], Int32, Int32)</td>
<td>Copies a range of elements from the list to array, starting at arrayIndex.</td>
</tr>
</tbody>
</table>
See Also

- ListBase<Of T> Class
- ListBase<Of T> Members
- Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::CopyTo Method (array<T>[])()

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies all the items in the list, in order, to array, starting at index 0.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual void CopyTo(
    T[] array
)

Visual Basic (Declaration)

Public Overridable Sub CopyTo (_
    array As T() _
)

Visual C++

public:
    virtual void CopyTo ( _
        array<T>^ array
    )

Parameters

array
    array<T>[](0)
    The array to copy to. This array must have a size that is greater than or equal to Count.
See Also

[ListBase(Of T)](Of T) Class
[Wintellect.PowerCollections] Namespace

Send [comments](comments) about this topic to Microsoft.
ListBase(Of T)::.CopyTo Method (Int32, array<T>[](), Int32, Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies a range of elements from the list to array, starting at arrayIndex.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public virtual void CopyTo(
    int index,
    T[] array,
    int arrayIndex,
    int count
)
```

Visual Basic (Declaration)

```vbnet
Public Overridable Sub CopyTo ( _
    index As Integer, _
    array As T(), _
    arrayIndex As Integer, _
    count As Integer _
)
```

Visual C++

```cpp
public:
    virtual void CopyTo ( 
        int index, 
        array<T>[] array, 
        int arrayIndex, 
        int count
    )
```

Parameters

index

Int32
The starting index in the source list of the range to copy.

array

array<T>[]()
The array to copy to. This array must have a size that is greater than or equal to Count + arrayIndex.
arrayIndex
   Int32
   The starting index in array to copy to.

count
   Int32
   The number of items to copy.
See Also

ListBase<Of<T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>:::Find Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual T Find(
    Predicate<T> predicate
)

Visual Basic (Declaration)

Public Overridable Function Find (_
    predicate As Predicate(Of T) _
) As T

Visual C++

public:
virtual T Find ( 
    Predicate<T>^ predicate 
)

Parameters

predicate
    Predicate<Of <T>>
    A delegate that defined the condition to check for.

Return Value

The first item that satisfies the condition predicate. If no item satisfies that condition, the default value for T is returned.
Remarks

If the default value for T (null or all-zero) matches the condition defined by predicate, and the list might contain the default value, then it is impossible to distinguish the different between finding the default value and not finding any item. To distinguish these cases, use `TryFind(Predicate<Of<T>>, T)`.
See Also

[ListBase<Of <T>>] Class
[Wintellect.PowerCollections] Namespace
[ListBase<Of <T>>].::TryFind(Predicate<Of <T>>, T%)

Send [comments] about this topic to Microsoft.
ListBase<(Of <T>)>...::FindIndex Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FindIndex(Predicate&lt;Of&lt;T&gt;&gt;)</code></td>
<td>Finds the index of the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.</td>
</tr>
<tr>
<td><code>FindIndex(Int32, Predicate&lt;Of&lt;T&gt;&gt;)</code></td>
<td>Finds the index of the first item, in the range of items extending from index to the end, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.</td>
</tr>
<tr>
<td><code>FindIndex(Int32, Int32, Predicate&lt;Of&lt;T&gt;&gt;)</code></td>
<td>Finds the index of the first item, in the range of count items starting from index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.</td>
</tr>
</tbody>
</table>
See Also

ListBase<(Of <T>)> Class
ListBase<(Of <T>)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::FindIndex Method (Predicate<Of <T>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public virtual int FindIndex(
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function FindIndex ( _
    predicate As Predicate(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:
virtual int FindIndex ( 
    Predicate<T>^ predicate
)
```

### Parameters

**predicate**

`Predicate(Of T)`

A delegate that defined the condition to check for.

### Return Value

The index of the first item that satisfies the condition `predicate`. If no item satisfies that condition, -1 is returned.
See Also

[ListBase(Of T)](Of T) Class
[Wintellect.PowerCollections](Namespace)

Send comments about this topic to Microsoft.
Finds the index of the first item, in the range of items extending from index to the end, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

**C#**

```csharp
public virtual int FindIndex(
    int index,
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function FindIndex ( _
    index As Integer, _
    predicate As Predicate(Of T) _
) As Integer
```

**Visual C++**

```csharp
public:
virtual int FindIndex ( 
    int index,
    Predicate<T>^ predicate
)
```

## Parameters

**index**

`Int32`

The starting index of the range to check.

**predicate**

`Predicate<(Of <T>)>`

A delegate that defined the condition to check for.

## Return Value

The index of the first item in the given range that satisfies the condition predicate. If no item satisfies that condition, -1 is returned.
See Also

[ListBase(Of T)](Of T) Class
[Wintellect.PowerCollections](PowerCollections) Namespace

Send [comments](comments) about this topic to Microsoft.
ListBase(Of T)::.FindIndex Method (Int32, Int32, Predicate(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the first item, in the range of count items starting from index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual int FindIndex(
        int index,
        int count,
        Predicate&lt;T&gt; predicate
    )

Visual Basic (Declaration)

Public Overridable Function FindIndex ( _
        index As Integer, _
        count As Integer, _
        predicate As Predicate(Of T) _
    ) As Integer

Visual C++

public:
virtual int FindIndex ( 
        int index,
        int count,
        Predicate&lt;T&gt;^ predicate
    )

Parameters

index
        Int32
        The starting index of the range to check.

count
        Int32
        The number of items in range to check.

predicate
        Predicate(Of &lt;T&gt;)

A delegate that defined the condition to check for.

**Return Value**

The index of the first item in the given range that satisfies the condition predicate. If no item satisfies that condition, -1 is returned.
See Also

ListBase<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase(Of T)::.FindLast Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public virtual T FindLast(
    Predicate<T> predicate
)
```

Visual Basic (Declaration)

```vbnet
Public Overridable Function FindLast (_
    predicate As Predicate(Of T) _
) As T
```

Visual C++

```cpp
public:
virtual T FindLast ( _
    Predicate<T>& predicate
)
```

Parameters

predicate

`Predicate(Of <T>)`

A delegate that defined the condition to check for.

Return Value

The last item that satisfies the condition predicate. If no item satisfies that condition, the default value for T is returned.
Remarks

If the default value for T (null or all-zero) matches the condition defined by predicate, and the list might contain the default value, then it is impossible to distinguish the different between finding the default value and not finding any item. To distinguish these cases, use `TryFindLast<Predicate<Of<T>>, T>`. 


See Also

ListBase<Of <T>> Class
Wintellect.PowerCollections Namespace
ListBase<Of <T>>...::TryFindLast(Predicate<Of <T>>, T%)

Send comments about this topic to Microsoft.
ListBase<(Of <T>)>:::FindLastIndex Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FindLastIndex(Predicate&lt;Of &lt;T&gt;&gt;)</code></td>
<td>Finds the index of the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.</td>
</tr>
<tr>
<td><code>FindLastIndex(Int32, Predicate&lt;Of &lt;T&gt;&gt;)</code></td>
<td>Finds the index of the last item, in the range of items extending from the beginning of the list to index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.</td>
</tr>
<tr>
<td><code>FindLastIndex(Int32, Int32, Predicate&lt;Of &lt;T&gt;&gt;)</code></td>
<td>Finds the index of the last item, in the range of count items ending at index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.</td>
</tr>
</tbody>
</table>
See Also

ListBase(Of T) Class
ListBase(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::FindLastIndex Method (Predicate<Of <T>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public virtual int FindLastIndex(
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function FindLastIndex ( _
    predicate As Predicate(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:
virtual int FindLastIndex ( _
    Predicate<T>^ predicate
)
```

### Parameters

predicate

`Predicate<T>`

A delegate that defined the condition to check for.

### Return Value

The index of the last item that satisfies the condition `predicate`. If no item satisfies that condition, -1 is returned.
See Also

ListBase<Of T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<(Of <T>)>::FindLastIndex Method (Int32, Predicate<(Of <T>)>)

See Also

Visual Basic (Declaration) ➤ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the last item, in the range of items extending from the beginning of the list to index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public virtual int FindLastIndex(
    int index,
    Predicate<T> predicate
)
```

### Visual Basic (Declaration)

```vbnet
Public Overridable Function FindLastIndex ( _
    index As Integer, _
    predicate As Predicate(Of T) _
) As Integer
```

### Visual C++

```csharp
public:
virtual int FindLastIndex (
    int index, 
    Predicate<T>^ predicate
)
```

## Parameters

**index**

Int32  
The ending index of the range to check.

**predicate**

Predicate(Of <T>)  
A delegate that defined the condition to check for.

## Return Value

The index of the last item in the given range that satisfies the condition predicate. If no item satisfies that condition, -1 is returned.
See Also

ListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::FindLastIndex Method (Int32, Int32, Predicate<Of <T>>)

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the last item, in the range of count items ending at index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual int FindLastIndex(
    int index,
    int count,
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function FindLastIndex ( _
    index As Integer, _
    count As Integer, _
    predicate As Predicate(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:
    virtual int FindLastIndex ( 
    int index, 
    int count, 
    Predicate<T>^ predicate
)
```

**Parameters**

**index**

Int32

The ending index of the range to check.

**count**

Int32

The number of items in range to check.

**predicate**

Predicate<(Of <T>)>
A delegate that defined the condition to check for.

**Return Value**

The index of the last item in the given range that satisfies the condition predicate. If no item satisfies that condition, -1 is returned.
See Also

ListBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Enumerates all of the items in the list, in order. The item at index 0 is enumerated first, then the item at index 1, and so on.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public override IEnumerator<T> GetEnumerator()
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides Function GetEnumerator As IEnumerator(Of T)
```

**Visual C++**

```cpp
public:
virtual IEnumerator<T>^ GetEnumerator () override
```

**Return Value**

An IEnumerate<T> that enumerates all the items in the list.
Remarks

The enumerator does not check for changes made to the structure of the list. Thus, changes to the list during enumeration may cause incorrect enumeration or out of range exceptions. Consider overriding this method and adding checks for structural changes.
See Also

ListBase<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::IndexOf Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IndexOf(T)</code></td>
<td>Finds the index of the first item in the list that is equal to item.</td>
</tr>
<tr>
<td><code>IndexOf(T, Int32)</code></td>
<td>Finds the index of the first item, in the range of items extending from index to the end, that is equal to item.</td>
</tr>
<tr>
<td><code>IndexOf(T, Int32, Int32)</code></td>
<td>Finds the index of the first item, in the range of count items starting from index, that is equal to item.</td>
</tr>
</tbody>
</table>
See Also

[ListBase(Of T)> Class
[ListBase(Of T)> Members
[Wintellect.PowerCollections] Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>::IndexOf Method (T)

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the first item in the list that is equal to item.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public virtual int IndexOf(T item)
```

Visual Basic (Declaration)

```vbnet
Public Overridable Function IndexOf(_
    item As T_
) As Integer
```

Visual C++

```cpp
public:
virtual int IndexOf(T item)
```

Parameters

`item`

* `T`  
  The item to search for.

Return Value

The index of the first item in the list that is equal to `item`. If no item is equal to `item`, -1 is returned.
Remarks

The default implementation of equality for type T is used in the search. This is the equality defined by IComparable<T> or object.Equals.
See Also

ListBase(Of <T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::IndexOf Method (T, Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the first item, in the range of items extending from index to the end, that is equal to item.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual int IndexOf(
    T item,
    int index
)

Visual Basic (Declaration)

Public Overridable Function IndexOf ( _
    item As T, _
    index As Integer _
) As Integer

Visual C++

public:
virtual int IndexOf ( _
    T item,
    int index
)

Parameters

item
    T
    The item to search for.

index
    Int32
    The starting index of the range to check.

Return Value

The index of the first item in the given range that that is equal to item. If no item is equal to item, -1 is returned.
Remarks

The default implementation of equality for type T is used in the search. This is the equality defined by IComparable<T> or object.Equals.
See Also

ListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<T>...::IndexOf Method (T, Int32, Int32)

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the first item, in the range of count items starting from index, that is equal to item.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public virtual int IndexOf(
    T item,
    int index,
    int count
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function IndexOf ( _
    item As T, _
    index As Integer, _
    count As Integer _
) As Integer
```

**Visual C++**

```csharp
public:
    virtual int IndexOf ( _
    T item, _
    int index, _
    int count
)
```

### Parameters

**item**

* T
  The item to search for.

**index**

* Int32
  The starting index of the range to check.

**count**

* Int32
The number of items in range to check.

**Return Value**

The index of the first item in the given range that is equal to item. If no item is equal to item, -1 is returned.
Remarks

The default implementation of equality for type T is used in the search. This is the equality defined by IComparable<T> or object.Equals.
See Also

ListBase<T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::Insert Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

This method must be overridden by the derived class to insert a new item at the given index.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public abstract void Insert(
    int index,
    T item
)

Visual Basic (Declaration)

Public MustOverride Sub Insert ( _
    index As Integer, _
    item As T _
)

Visual C++

public:
virtual void Insert ( 
    int index,
    T item
) abstract

Parameters

index
    Int32
    The index in the list to insert the item at. After the insertion, the inserted item is located at this index. The first item in the list has index 0.

item
    T
    The item to insert at the given index.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>index is less than zero or greater than Count.</td>
</tr>
</tbody>
</table>
See Also

[ListBase(Of T)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::LastIndexOf Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LastIndexOf(T)</code></td>
<td>Finds the index of the last item in the list that is equal to item.</td>
</tr>
<tr>
<td><code>LastIndexOf(T, Int32)</code></td>
<td>Finds the index of the last item, in the range of items extending from the beginning of the list to index, that is equal to item.</td>
</tr>
<tr>
<td><code>LastIndexOf(T, Int32, Int32)</code></td>
<td>Finds the index of the last item, in the range of count items ending at index, that is equal to item.</td>
</tr>
</tbody>
</table>
See Also

ListBase(Of T) Class
ListBase(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::LastIndexOf Method (T)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the last item in the list that is equal to item.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual int LastIndexOf(T item)

Visual Basic (Declaration)

Public Overridable Function LastIndexOf (_
    item As T _
) As Integer

Visual C++

public:
    virtual int LastIndexOf ( T item

Parameters

item
    T
    The item to search for.

Return Value

The index of the last item in the list that that is equal to item. If no item is equal to item, -1 is returned.
The default implementation of equality for type T is used in the search. This is the equality defined by IComparable<T> or object.Equals.
See Also

[ListBase<Of <T>>](#) Class
[Wintellect.PowerCollections](#) Namespace

Send [comments](#) about this topic to Microsoft.
ListBase<(Of <T>)>...::LastIndexOf Method (T, Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the last item, in the range of items extending from the beginning of the list to index, that is equal to item.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual int LastIndexOf(
    T item,
    int index
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function LastIndexOf ( _
    item As T, _
    index As Integer _
) As Integer
```

**Visual C++**

```cpp
public:
virtual int LastIndexOf ( 
    T item, 
    int index 
)
```

**Parameters**

**item**

T

The item to search for.

**index**

Int32

The ending index of the range to check.

**Return Value**

The index of the last item in the given range that that is equal to item. If no item is equal to item, -1 is returned.
Remarks

The default implementation of equality for type T is used in the search. This is the equality defined by IComparable<T> or object.Equals.
See Also

ListBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<
( Of < T > ) > Ref LastIndexOf Method < T, Int32, Int32 >

See Also

Visual Basic (Declaration) ➤ Visual Basic (Usage)
C#
Visual C++
J# JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the last item, in the range of count items ending at index, that is equal to item.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual int LastIndexOf(
    T item,
    int index,
    int count
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function LastIndexOf ( _
    item As T, _
    index As Integer, _
    count As Integer _
) As Integer
```

**Visual C++**

```cpp
public:
    virtual int LastIndexOf ( 
    T item, 
    int index, 
    int count 
)
```

**Parameters**

**item**

T

The item to search for.

**index**

Int32

The ending index of the range to check.

**count**

Int32
The number of items in range to check.

**Return Value**

The index of the last item in the given range that that is equal to item. If no item is equal to item, -1 is returned.
Remarks

The default implementation of equality for type T is used in the search. This is the equality defined by IComparable<T> or object.Equals.
See Also

ListBase<Of T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<(Of <T>)>::Range Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a view onto a sub-range of this list. Items are not copied; the returned IList<T> is simply a different view onto the same underlying items. Changes to this list are reflected in the view, and vice versa. Insertions and deletions in the view change the size of the view, but insertions and deletions in the underlying list do not.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public virtual IList<T> Range(
    int start,
    int count
)
```

Visual Basic (Declaration)

```vbnet
Public Overridable Function Range ( _
    start As Integer, _
    count As Integer _
) As IList(Of T)
```

Visual C++

```cpp
public:
virtual IList<T>^ Range ( _
    int start,
    int count
)
```

Parameters

**start**

`Int32`  
The starting index of the view.

**count**

`Int32`  
The number of items in the view.

Return Value

A list that is a view onto the given sub-part of this list.
Remarks

This method can be used to apply an algorithm to a portion of a list. For example:

Copy Code

`Algorithms.ReverseInPlace(deque.Range(3, 6))`

will reverse the 6 items beginning at index 3.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentOutOfRangeException</code></td>
<td>start or count is negative.</td>
</tr>
<tr>
<td><code>System..::ArgumentOutOfRangeException</code></td>
<td>start + count is greater than the size of the list.</td>
</tr>
</tbody>
</table>
See Also

[ListBase<Of T>] Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Searches the list for the first item that compares equal to item. If one is found, it is removed. Otherwise, the list is unchanged.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public override bool Remove(T item)
```

### Visual Basic (Declaration)

```vbnet
Public Overrides Function Remove(_
    item As T _) As Boolean
```

### Visual C++

```cpp
public:
    virtual bool Remove ( T item ) override
```

## Parameters

- **item**
  
  T
  
  The item to remove from the list.

## Return Value

True if an item was found and removed that compared equal to item. False if no such item was in the list.
Equality in the list is determined by the default sense of equality for T. If T implements IComparable<T>, the Equals method of that interface is used to determine equality. Otherwise, Object.Equals is used to determine equality.
See Also

ListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<(Of <T>)> RemoveAt Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

This method must be overridden by the derived class to remove the item at the given index.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
C#  

```csharp
public abstract void RemoveAt(int index)
```

Visual Basic (Declaration)  

```vbnet
Public MustOverride Sub RemoveAt(_
    index As Integer _
)
```

Visual C++  

```cpp
public:
virtual void RemoveAt(_
    int index
) abstract
```

Parameters  

index  

Int32  

The index in the list to remove the item at. The first item in the list has index 0.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>index is less than zero or greater than or equal to Count.</td>
</tr>
</tbody>
</table>
See Also

ListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<(Of <T>)>::System.Collections_IList.Add Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds an item to the end of the list. This method is equivalent to calling:

Copy Code

Insert(Count, item)

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private int IList.Add(
    Object value
)

Visual Basic (Declaration)

Private Function System.Collections.IList.Add ( _
    value As Object _
) As Integer Implements IList.Add

Visual C++

private:
    virtual int System.Collections.IList.Add ( 
        Object^ value
    ) sealed = IList::Add

Parameters

value
    Object
    The item to add to the list.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::ArgumentException</code></td>
<td>value cannot be converted to T.</td>
</tr>
</tbody>
</table>
See Also

ListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase(Of T)\ldots::System.Collections.IList.Clear Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all the items from the list, resulting in an empty list.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private void IList.Clear()

Visual Basic (Declaration)

Private Sub System.Collections.IList.Clear Implements IList.Clear

Visual C++

private:
virtual void System.Collections.IList.Clear () sealed = IList::Clear
See Also

ListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::System.Collections.IList.Contains Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if the list contains any item that compares equal to value.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private bool IList.Contains(
    Object value
)

Visual Basic (Declaration)

Private Function System.Collections.IList.Contains ( _
    value As Object _
) As Boolean Implements IList.Contains

Visual C++

private:
virtual bool System.Collections.IList.Contains ( 
    Object^ value
) sealed = IList::Contains

Parameters

value
    Object
    The item to search for.
Remarks

Equality in the list is determined by the default sense of equality for T. If T implements IComparable<T>, the Equals method of that interface is used to determine equality. Otherwise, Object.Equals is used to determine equality.
See Also

ListBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<(Of <T>)>...::System.Collections.IList.IndexOf Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Find the first occurrence of an item equal to value in the list, and returns the index of that item.

**Namespace**: Wintellect.PowerCollections

**Assembly**: PowerCollections (in PowerCollections.dll)
Syntax

C#

private int IList.IndexOf(
    Object value
)

Visual Basic (Declaration)

Private Function System.Collections.IList.IndexOf ( _
    value As Object _
) As Integer Implements IList.IndexOf

Visual C++

private:
virtual int System.Collections.IList.IndexOf ( _
    Object^ value
) sealed = IList::IndexOf

Parameters

value

Object
The item to search for.

Return Value

The index of value, or -1 if no item in the list compares equal to value.
Remarks

Equality in the list is determined by the default sense of equality for T. If T implements IComparable<T>, the Equals method of that interface is used to determine equality. Otherwise, Object.Equals is used to determine equality.
See Also

[ListBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::System.Collections.IList.Insert Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Insert a new item at the given index.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private void IList.Insert(
    int index,
    Object value
)

Visual Basic (Declaration)

Private Sub System.Collections.IList.Insert ( _
    index As Integer, _
    value As Object _
) Implements IList.Insert

Visual C++

private:
virtual void System.Collections.IList.Insert ( _
    int index,
    Object^ value
) sealed = IList::Insert

Parameters

index

Int32
The index in the list to insert the item at. After the insertion, the inserted item is located at this index. The first item in the list has index 0.

value

Object
The item to insert at the given index.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentOutOfRangeException</code></td>
<td>index is less than zero or greater than Count.</td>
</tr>
<tr>
<td><code>System..::ArgumentOutOfRangeException</code></td>
<td>value cannot be converted to T.</td>
</tr>
</tbody>
</table>
See Also

ListBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<(Of <T>)>...::System.Collections.IList.Remove Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Searches the list for the first item that compares equal to value. If one is found, it is removed. Otherwise, the list is unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private void IList.Remove(
    Object value
)

Visual Basic (Declaration)

Private Sub System.Collections.IList.Remove ( _
    value As Object _
) Implements IList.Remove

Visual C++

private:
    virtual void System.Collections.IList.Remove ( _
        Object^ value
    ) sealed = IList::Remove

Parameters

value
    Object
        The item to remove from the list.
Remarks

Equality in the list is determined by the default sense of equality for T. If T implements IComparable<T>, the Equals method of that interface is used to determine equality. Otherwise, Object.Equals is used to determine equality.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...::ArgumentException</td>
<td>value cannot be converted to T.</td>
</tr>
</tbody>
</table>
See Also

`ListBase<Of <T>>` Class
`Wintellect.PowerCollections` Namespace

Send [comments](mailto:) about this topic to Microsoft.
ListBase<
(Of <T>)>...::System.Collections.IList.RemoveAt Method

See Also

Visual Basic (Declaration) ▸ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes the item at the given index.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
private void IList.RemoveAt(
    int index
)
```

**Visual Basic (Declaration)**

```vbnet
Private Sub System.Collections.IList.RemoveAt (_
    index As Integer _
) Implements IList.RemoveAt
```

**Visual C++**

```cpp
private:
virtual void System.Collections.IList.RemoveAt (_
    int index
) sealed = IList::RemoveAt
```

### Parameters

**index**

- **Int32**
  - The index in the list to remove the item at. The first item in the list has index 0.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>index is less than zero or greater than or equal to Count.</td>
</tr>
</tbody>
</table>
See Also

**ListBase<Of<T>>** Class  
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
ListBase<Of <T>>...::TryFind Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the first item in the list that satisfies the condition defined by predicate.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual bool TryFind(
    Predicate<T> predicate,
    out T foundItem
)

Visual Basic (Declaration)

Public Overridable Function TryFind ( _
    predicate As Predicate(Of T), _
    <OutAttribute> ByRef foundItem As T _
) As Boolean

Visual C++

public:
    virtual bool TryFind ( 
        Predicate<T>^ predicate,
        [OutAttribute] T% foundItem
    )

Parameters

predicate
    Predicate<Of <T>>
    A delegate that defines the condition to check for.

foundItem
    T%
    If true is returned, this parameter receives the first item in the list that satisfies the condition defined by predicate.

Return Value

True if an item that satisfies the condition predicate was found. False if no item
in the list satisfies that condition.
See Also

[ListBase(Of T)](Of_T) Class
[Wintellect.PowerCollections](PowerCollections) Namespace

Send [comments](comments) about this topic to Microsoft.
ListBase<Of <T>>...:::TryFindLast Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the last item in the list that satisfies the condition defined by predicate.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public virtual bool TryFindLast(
    Predicate<T> predicate,
    out T foundItem
)
```

Visual Basic (Declaration)

```vbnet
Public Overridable Function TryFindLast ( _
    predicate As Predicate(Of T), _
    <OutAttribute> ByRef foundItem As T _
) As Boolean
```

Visual C++

```cpp
public:
virtual bool TryFindLast (  
    Predicate<T>^ predicate,
    [OutAttribute] T% foundItem
)
```

Parameters

predicate

`Predicate<(Of <T>)>`

A delegate that defines the condition to check for.

foundItem

`T%`

If true is returned, this parameter receives the last item in the list that satisfies the condition defined by predicate.

Return Value

True if an item that satisfies the condition predicate was found. False if no item
in the list satisfies that condition.
See Also

ListBase<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<(Of <T>)> Properties

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class ListBase<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public MustInherit Class ListBase(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class ListBase abstract
Type Parameters

T

The type exposes the following properties.
# Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Count</code></td>
<td>The property must be overridden by the derived class to return the number of items in the list. (Overrides <code>CollectionBase&lt;Of &lt;T&gt;&gt;::Count</code>.)</td>
</tr>
<tr>
<td><code>Item</code></td>
<td>The indexer must be overridden by the derived class to get and set values of the list at a particular index.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection&lt;(Of <code>T</code>)&gt;::IsReadOnly</td>
<td>(Inherited from CollectionBase&lt;(Of <code>T</code>)&gt;). Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td>ICollection::IsSynchronized</td>
<td>(Inherited from CollectionBase&lt;(Of <code>T</code>)&gt;). Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td>ICollection::SyncRoot</td>
<td>(Inherited from CollectionBase&lt;(Of <code>T</code>)&gt;).</td>
</tr>
<tr>
<td>IList::IsFixedSize</td>
<td>Returns whether the list is a fixed size. This implementation always returns false.</td>
</tr>
<tr>
<td>IList::IsReadOnly</td>
<td>Returns whether the list is read only. This implementation returns the value from ICollection&lt;<code>T</code>&gt;.IsReadOnly, which is by default, false.</td>
</tr>
<tr>
<td>IList::Item</td>
<td>Gets or sets the value at a particular index in the list.</td>
</tr>
</tbody>
</table>
See Also

ListBase<Of <T>>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::Count Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

The property must be overridden by the derived class to return the number of items in the list.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public abstract int Count { get; }
```

**Visual Basic (Declaration)**

```vbnet
Public MustOverride ReadOnly Property Count As Integer
```

**Visual C++**

```c++
public:
    virtual property int Count {
        int get () abstract override;
    }
```

Field Value

The number of items in the list.
See Also

ListBase<T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

The indexer must be overridden by the derived class to get and set values of the list at a particular index.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public abstract T this[
    int index
]{ get; set;}

Visual Basic (Declaration)

Public MustOverride Default Property Item ( _
    index As Integer _
) As T

Visual C++

public:
virtual property T default[int index] { 
    T get (int index) abstract;
    void set (int index, T value) abstract;
}

Parameters

index

Int32
The index in the list to get or set an item at. The first item in the list has index 0, and the last has index Count-1.

Return Value

The item at the given index.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>index is less than zero or greater than or equal to Count.</td>
</tr>
</tbody>
</table>
See Also

ListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<Of <T>>...::System.Collections.IList.IsFixedSize Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns whether the list is a fixed size. This implementation always returns false.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
private bool IList.IsFixedSize{ get; }
```

**Visual Basic (Declaration)**

```vbnet
Private ReadOnly Property System.Collections.IList.IsFixedSize As Boolean
```

**Visual C++**

```cpp
private:
virtual property bool System.Collections.IList.IsFixedSize {
    bool get () sealed = IList::IsFixedSize::get;
}
```

### Field Value

Always false, indicating that the list is not fixed size.
See Also

[ListBase<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ListBase<(Of <T>)>...::System.Collections.IList.IsReadOnly Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns whether the list is read only. This implementation returns the value from ICollection<T>.IsReadOnly, which is by default, false.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private bool IList.IsReadOnly{ get; }

Visual Basic (Declaration)

Private ReadOnly Property System.Collections.IList.IsReadOnly As Bool

Visual C++

private:
virtual property bool System.Collections.IList.IsReadOnly { bool get () sealed = IList::IsReadOnly::get; }

Field Value

By default, false, indicating that the list is not read only.
See Also

[ListBase<Of <T>>](#) Class
[Wintellect.PowerCollections](#) Namespace

Send [comments](#) about this topic to Microsoft.
ListBase(Of T)>...:::System.Collections.IList.Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets or sets the value at a particular index in the list.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private Object IList.Item[
    int index
] { get; set; }

Visual Basic (Declaration)

Private Property System.Collections.IList.Item ( _
    index As Integer _
) As Object Implements IList.Item

Visual C++

private:
    virtual property Object^ System.Collections.IList.Item[int index] { 
        Object^ get (int index) sealed = IList::Item::get;
        void set (int index, Object^ value) sealed = IList::Item::set;
    }

Parameters

index

    Int32
    The index in the list to get or set an item at. The first item in the list has
    index 0, and the last has index Count-1.

Field Value

The item at the given index.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>index is less than zero or greater than or equal to Count.</td>
</tr>
<tr>
<td><code>System::ArgumentException</code></td>
<td>value cannot be converted to T.</td>
</tr>
</tbody>
</table>
See Also

[ListBase<Of <T>>](#) Class
[Wintellect.PowerCollections](#) Namespace

Send [comments](#) about this topic to Microsoft.
The MultiDictionary class that associates values with a key. Unlike an Dictionary, each key can have multiple values associated with it. When indexing an MultiDictionary, instead of a single value associated with a key, you retrieve an enumeration of values.

When constructed, you can chose to allow the same value to be associated with a key multiple times, or only one time.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

[SerializableAttribute]
public class MultiDictionary<TKey, TValue> : MultiDictionaryBase<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute>
Public Class MultiDictionary(Of TKey, TValue) _
    Inherits MultiDictionaryBase(Of TKey, TValue) _
    Implements ICloneable

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class MultiDictionary : public MultiDictionaryBase<TKey, TValue>, ICloneable
Type Parameters

TKey
   The type of the keys.

TValue
   The type of values associated with the keys.
Inheritance Hierarchy

```csharp
System::Object
Wintellect.PowerCollections::CollectionBase<(Of <KeyValuePair<(Of <TKey, ICollection<(Of <TValue)>)>)>)
    Wintellect.PowerCollections::MultiDictionaryBase<(Of <TKey, TValue>)>
    Wintellect.PowerCollections::MultiDictionary<(Of <TKey, TValue>)>
```
See Also

MultiDictionary(Of TKey, TValue) Members
Wintellect.PowerCollections Namespace
System.Collections.Generic:::Dictionary(Of TKey, TValue)
Wintellect.PowerCollections:::OrderedMultiDictionary(Of TKey, TValue)

Send comments about this topic to Microsoft.
MultiDictionary(Of TKey, TValue) Members

See Also  Methods  Properties  Constructors  Explicit Interface Implementations

☐ Include Inherited Members  ☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class MultiDictionary<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class MultiDictionary(Of TKey, TValue)

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class MultiDictionary
Type Parameters

TKey
TValue

The type exposes the following members.
## Public Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MultiDictionary&lt;Of TKey, TValue&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>MultiDictionary&lt;Of TKey, TValue&gt;</code> New</td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td></td>
<td>Adds new values to be associated with a key. If duplicate values are permitted, this method always adds new key-value pairs to the dictionary.</td>
</tr>
<tr>
<td><strong>AddMany</strong></td>
<td>If duplicate values are not permitted, and key already has a value equal to one of values associated with it, then that value is replaced, and the number of values associate with key is unchanged.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes.</td>
</tr>
<tr>
<td></td>
<td>Since the returned object is just a view, changes to the collection will be reflected in the view.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from CollectionBase&lt;Of&lt;T&gt; &gt;.)</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all keys and values from the dictionary.</td>
</tr>
<tr>
<td></td>
<td>(Overrides MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;::Clear().)</td>
</tr>
<tr>
<td><strong>Clone</strong></td>
<td>Makes a shallow clone of this dictionary; i.e., if keys or values of the dictionary are reference types, then they are not cloned. If TKey or TValue is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td></td>
<td>Makes a deep clone of this dictionary. A new dictionary is created with a clone of each entry of this dictionary, by calling ICloneable.Clone on each element. If TKey or TValue is a value type, then each element is copied as if by simple assignment.</td>
</tr>
</tbody>
</table>

| **CloneContents** | |
|                   | |


**Contains**
Overloaded.

**ContainsKey**
Checks to see if the key is present in the dictionary and has at least one value associated with it.

**ConvertAll(Of TOutput)**
Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.
(Inherited from `CollectionBase<Of T>`.)

**CopyTo**
GetEnumerator to get all the items and copy them to the provided array.
(Inherited from `CollectionBase<Of T>`.)

**CountWhere**
Counts the number of items in the collection that satisfy the condition defined by predicate.
(Inherited from `CollectionBase<Of T>`.)

**Equals**
Determines whether the specified `Object` is equal to the current `Object`.
(Inherited from `Object`.)

**Exists**
Determines if the collection contains any item that satisfies the condition defined by predicate.
(Inherited from `CollectionBase<Of T>`.)

**FindAll**
Enumerates the items in the collection that satisfy the condition defined by predicate.
(Inherited from `CollectionBase<Of T>`.)

**ForEach**
Performs the specified action on each item in this collection.
(Inherited from `CollectionBase<Of T>`.)

**GetEnumerator**
Enumerate all the keys in the dictionary, and for each key, the collection of values for that key.
(Inherited from `MultiDictionaryBase<Of TKey, TValue>`.)

**GetHashCode**
Serves as a hash function for a particular type.

**GetHashCode()** is suitable for use in hashing algorithms and data structures like a hash table.
(Inherited from `Object`.)

**GetType**
Gets the `Type` of the current instance.
(Inherited from `Object`.)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Inheritance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Remove</strong></td>
<td>Overloaded. Removes all the items in the collection that satisfy the condition defined by predicate.</td>
<td>(Inherited from <code>CollectionBase&lt;Of&lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>RemoveAll</strong></td>
<td>Overloaded. Removes all the items in the collection that satisfy the condition defined by predicate.</td>
<td>(Inherited from <code>CollectionBase&lt;Of&lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>RemoveMany</strong></td>
<td>Overloaded. Replaces all values associated with key with the single value.</td>
<td>(Inherited from <code>MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>Replace</strong></td>
<td>Replaces all values associated with key with a new collection of values. If the collection does not permit duplicate values, and values has duplicate items, then only the last of duplicates is added.</td>
<td>(Inherited from <code>MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>ReplaceMany</strong></td>
<td>Replaces all values associated with key with a new collection of values. If the collection does not permit duplicate values, and values has duplicate items, then only the last of duplicates is added.</td>
<td>(Inherited from <code>MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>ToArray</strong></td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.</td>
<td>(Inherited from <code>CollectionBase&lt;Of&lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.</td>
<td>(Inherited from <code>MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>TrueForAll</strong></td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate.</td>
<td>(Inherited from <code>CollectionBase&lt;Of&lt;T&gt;&gt;</code>.)</td>
</tr>
</tbody>
</table>
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CountAllValues</strong></td>
<td>Gets a total count of values in the collection. This default implementation is slow; it enumerates all of the keys in the dictionary and calls CountValues on each. A derived class may be able to supply a more efficient implementation. (Inherited from MultiDictionaryBase&lt; Of &lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>CountValues</strong></td>
<td>Gets the number of values associated with a given key.</td>
</tr>
<tr>
<td><strong>EnumerateKeys</strong></td>
<td>Enumerate all the keys in the dictionary.</td>
</tr>
<tr>
<td><strong>EqualValues</strong></td>
<td>Determine if two values are equal.</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td><strong>TryEnumerateValuesForKey</strong></td>
<td>Determines if this dictionary contains a key equal to key. If so, all the values associated with that key are returned through the values parameter.</td>
</tr>
</tbody>
</table>
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td>Gets the number of key-value pairs in the dictionary. Each value associated with a given key is counted. If duplicate values are permitted, each duplicate value is included in the count. (Overrides MultiDictionaryBase&lt;Of TKey, TValue&gt;...&lt;/Count.)</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Returns a collection of all of the values in the dictionary associated with key, or changes the set of values associated with key. If the key is not present in the dictionary, an ICollection enumerating no values is returned. The returned collection of values is read-write, and can be used to modify the collection of values associated with the key. (Inherited from MultiDictionaryBase&lt;Of TKey, TValue&gt;.)</td>
</tr>
<tr>
<td><strong>KeyComparer</strong></td>
<td>Returns the IEqualityComparer&lt;T&gt; used to compare keys in this dictionary.</td>
</tr>
<tr>
<td><strong>Keys</strong></td>
<td>Gets a read-only collection all the keys in this dictionary. (Inherited from MultiDictionaryBase&lt;Of TKey, TValue&gt;.)</td>
</tr>
<tr>
<td><strong>KeyValuePairs</strong></td>
<td>Gets a read-only collection of all key-value pairs in the dictionary. If a key has multiple values associated with it, then a key-value pair is present for each value associated with the key. (Inherited from MultiDictionaryBase&lt;Of TKey, TValue&gt;.)</td>
</tr>
<tr>
<td><strong>ValueComparer</strong></td>
<td>Returns the IEqualityComparer&lt;T&gt; used to compare values in this dictionary.</td>
</tr>
<tr>
<td><strong>Values</strong></td>
<td>Gets a read-only collection of all the values in the dictionary. (Inherited from MultiDictionaryBase&lt;Of TKey, TValue&gt;.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;(</code>Of <code>&lt;T&gt;``)</code>, <code>::IsReadOnly</code></td>
<td>(Inherited from <code>CollectionBase&lt;(</code>Of <code>&lt;T&gt;``)</code>).</td>
</tr>
<tr>
<td><code>IDictionary&lt;(</code>Of <code>&lt;TKey&gt;, </code>ICollection&lt;(<code>Of </code>&lt;TValue&gt;``)&gt;<code>)</code>, <code>::Add</code></td>
<td>(Inherited from <code>MultiDictionaryBase&lt;(</code>Of <code>&lt;TKey, TValue&gt;``)</code>).</td>
</tr>
<tr>
<td><code>IDictionary&lt;(</code>Of <code>&lt;TKey, </code>ICollection&lt;(<code>Of </code>&lt;TValue&gt;``)&gt;<code>)</code>, <code>::Item</code></td>
<td>(Inherited from <code>MultiDictionaryBase&lt;(</code>Of <code>&lt;TKey, TValue&gt;``)</code>).</td>
</tr>
<tr>
<td><code>IDictionary&lt;(</code>Of <code>&lt;TKey, </code>ICollection&lt;(<code>Of </code>&lt;TValue&gt;``)&gt;<code>)</code>, <code>::TryGetValue</code></td>
<td>(Inherited from <code>MultiDictionaryBase&lt;(</code>Of <code>&lt;TKey, TValue&gt;``)</code>).</td>
</tr>
<tr>
<td><code>IDictionary&lt;(</code>Of <code>&lt;TKey, </code>ICollection&lt;(<code>Of </code>&lt;TValue&gt;``)&gt;<code>)</code>, <code>::Values</code></td>
<td>(Inherited from <code>MultiDictionaryBase&lt;(</code>Of <code>&lt;TKey, TValue&gt;``)</code>).</td>
</tr>
<tr>
<td><code>ICollection&lt;</code>, <code>::CopyTo</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>CollectionBase&lt;(</code>Of <code>&lt;T&gt;``)</code>).</td>
</tr>
<tr>
<td><code>ICollection&lt;</code>, <code>::IsSynchronized</code></td>
<td>Indicates whether the collection is synchronized. (Inherited from <code>CollectionBase&lt;(</code>Of <code>&lt;T&gt;``)</code>).</td>
</tr>
<tr>
<td><code>ICollection&lt;</code>, <code>::SyncRoot</code></td>
<td>Indicates the synchronization object for this collection. (Inherited from <code>CollectionBase&lt;(</code>Of <code>&lt;T&gt;``)</code>).</td>
</tr>
<tr>
<td><code>ICollection&lt;</code>, <code>::IEnumerator</code></td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the</td>
</tr>
</tbody>
</table>
IEnumerable.

IEnumerator<T> that was overridden by the derived classes to enumerate the members of the collection. (Inherited from CollectionBase<Of <T>>.)

ICloneable.

Implements ICloneable.Clone. Makes a shallow clone of this dictionary; i.e., if keys or values are reference types, then they are not cloned.

IEnumerator<T>.

IEnumerator<T> that was overridden by the derived classes to enumerate the members of the collection. (Inherited from CollectionBase<Of <T>>.)

ICloneable.

ICloneable::Clone.
See Also

MultiDictionary(Of TKey, TValue)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<(Of <TKey, TValue>)> Constructor

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MultiDictionary</strong>&lt;<em>(Of &lt;TKey, TValue&gt;</em>&gt;)<em>&gt;MultiDictionary**&lt;</em>(Of &lt;TKey, TValue&gt;*&gt;)*New(Boolean)</td>
<td>Create a new MultiDictionary. The default ordering of keys and values are used. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key &quot;foo&quot; could have &quot;a&quot;, &quot;a&quot;, and &quot;b&quot; associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key &quot;foo&quot; could have &quot;a&quot; and &quot;b&quot; associated with it, which key &quot;bar&quot; has values &quot;b&quot; and &quot;c&quot; associated with it.</td>
</tr>
<tr>
<td><strong>MultiDictionary</strong>&lt;<em>(Of &lt;TKey, TValue&gt;</em>&gt;)<em>&gt;MultiDictionary**&lt;</em>(Of &lt;TKey, TValue&gt;<em>&gt;)<em>New(Boolean, IEqualityComparer&lt;</em>(Of &lt;TKey&gt;</em>&gt;&gt;)</td>
<td>Create a new MultiDictionary. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key &quot;foo&quot; could have &quot;a&quot;, &quot;a&quot;, and &quot;b&quot; associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key &quot;foo&quot; could have &quot;a&quot; and &quot;b&quot; associated with it, which key &quot;bar&quot; has values &quot;b&quot; and &quot;c&quot; associated with it.</td>
</tr>
<tr>
<td><strong>MultiDictionary</strong>&lt;<em>(Of &lt;TKey, TValue&gt;</em>&gt;)<em>&gt;MultiDictionary**&lt;</em>(Of &lt;TKey, TValue&gt;<em>&gt;)<em>New(Boolean, IEqualityComparer&lt;</em>(Of &lt;TKey&gt;</em>&gt;&gt;),</td>
<td>Create a new MultiDictionary. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key &quot;foo&quot; could have &quot;a&quot;, &quot;a&quot;, and &quot;b&quot; associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys</td>
</tr>
</tbody>
</table>
can have the same value. For example, the key "foo" could have "a" and "b" associated with it, which key "bar" has values "b" and "c" associated with it.
See Also

MultiDictionary(Of TKey, TValue)> Class
MultiDictionary(Of TKey, TValue)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<(Of <TKey, TValue> )> Constructor (Boolean)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Create a new MultiDictionary. The default ordering of keys and values are used. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key "foo" could have "a", "a", and "b" associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key "foo" could have "a" and "b" associated with it, which key "bar" has values "b" and "c" associated with it.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public MultiDictionary(
    bool allowDuplicateValues
)
```

Visual Basic (Declaration)

```vbnet
Public Sub New ( _
    allowDuplicateValues As Boolean _
)
```

Visual C++

```cpp
public:
MultiDictionary ( 
    bool allowDuplicateValues
)
```

Parameters

allowDuplicateValues

Boolean
Can the same value be associated with a key multiple times?
Remarks

The default ordering of keys and values will be used, as defined by TKey and TValue's implementation of IComparable<T> (or IComparable if IComparable<T> is not implemented). If a different ordering should be used, other constructors allow a customComparer or IComparer to be passed to changed the ordering.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>TKey or TValue does not implement either IComparable&lt;T&gt; or IComparable.</td>
</tr>
</tbody>
</table>
See Also

*MultiDictionary(Of TKey, TValue)> Class

Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<Of <TKey, TValue>> Constructor (Boolean, IEqualityComparer<Of <TKey>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Create a new MultiDictionary. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key "foo" could have "a", "a", and "b" associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key "foo" could have "a" and "b" associated with it, which key "bar" has values "b" and "c" associated with it.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public MultiDictionary(
    bool allowDuplicateValues,
    IEqualityComparer<TKey> keyEqualityComparer
)
```

Visual Basic (Declaration)

```vbnet
Public Sub New (_
    allowDuplicateValues As Boolean, _
    keyEqualityComparer As IEqualityComparer(Of TKey) _
)
```

Visual C++

```cpp
public:
MultiDictionary (;
    bool allowDuplicateValues,
    IEqualityComparer<TKey>^ keyEqualityComparer
)
```

Parameters

allowDuplicateValues

```csharp
Boolean
```
Can the same value be associated with a key multiple times?

keyEqualityComparer

```csharp
IEqualityComparer(Of TKey)
```
An IEqualityComparer<TKey> instance that will be used to compare keys.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>TValue does not implement either IComparable&lt;TValue&gt; or IComparable.</td>
</tr>
</tbody>
</table>
See Also

`MultiDictionary(Of <TKey, TValue>)` Class
`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
MultiDictionary(Of TKey, TValue)> Constructor (Boolean, IEqualityComparer(Of TKey>, IEqualityComparer(Of TValue>)>

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Create a new MultiDictionary. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key "foo" could have "a", "a", and "b" associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key "foo" could have "a" and "b" associated with it, which key "bar" has values "b" and "c" associated with it.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public MultiDictionary(
    bool allowDuplicateValues,
    IEqualityComparer<TKey> keyEqualityComparer,
    IEqualityComparer<TValue> valueEqualityComparer
)

Visual Basic (Declaration)

Public Sub New (_
    allowDuplicateValues As Boolean, _
    keyEqualityComparer As IEqualityComparer(Of TKey), _
    valueEqualityComparer As IEqualityComparer(Of TValue)
)

Visual C++

public:
MultiDictionary ( _
    bool allowDuplicateValues, _
    IEqualityComparer<TKey>^ keyEqualityComparer, _
    IEqualityComparer<TValue>^ valueEqualityComparer
)

Parameters

allowDuplicateValues  
Boolean  
Can the same value be associated with a key multiple times?

keyEqualityComparer  
IEqualityComparer<TKey>  
An IEqualityComparer<TKey> instance that will be used to compare keys.

valueEqualityComparer  
IEqualityComparer<TValue>  

An IEqualityComparer<TValue> instance that will be used to compare values.
See Also

MultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary(Of TKey, TValue)> Methods

See Also

☐ Include Inherited Members ☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class MultiDictionary<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class MultiDictionary(Of TKey, TValue)

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class MultiDictionary
Type Parameters

TKey
TValue

The type exposes the following methods.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td></td>
<td>Adds new values to be associated with a key. If duplicate values are permitted, this method always adds new key-value pairs to the dictionary.</td>
</tr>
<tr>
<td><strong>AddMany</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If duplicate values are not permitted, and key already has a value equal to one of values associated with it, then that value is replaced, and the number of values associated with key is unchanged.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from MultiDictionaryBase&lt;Of TKey, TValue&gt;).</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase&lt;Of T&gt;).</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all keys and values from the dictionary. (Overides MultiDictionaryBase&lt;Of TKey, TValue&gt;).:Clear()</td>
</tr>
<tr>
<td><strong>Clone</strong></td>
<td>Makes a shallow clone of this dictionary; i.e., if keys or values of the dictionary are reference types, then they are not cloned. If TKey or TValue is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td><strong>CloneContents</strong></td>
<td>Makes a deep clone of this dictionary. A new dictionary is created with a clone of each entry of this dictionary, by calling ICloneable.Clone on each element. If TKey or TValue is a value type, then each element is copied as if by simple assignment.</td>
</tr>
</tbody>
</table>
Contains

Check if the key is present in the dictionary and has at least one value associated with it.

ConvertAll(Of <TOutput>)}

Convert all items in the collection by applying a delegate to each item. The resulting enumeration contains the result of applying the converter to each item in this collection, in order.

CopyTo

Copy all the items in the collection into an array.

CountWhere

Counts the number of items in the collection that satisfy the condition defined by predicate.

Exists

Determines if the collection contains any item that satisfies the condition defined by predicate.

FindAll

Enumerates the items in the collection that satisfy the condition defined by predicate.

ForEach

Performs the specified action on each item in this collection.

GetEnumerator

Enumerates all the keys in the dictionary, and for each key, the collection of values for that key.

GetHashCode

Serves as a hash function for a particular type.

GetType

Gets the Type of the current instance.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Inheritance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Remove</strong></td>
<td>Overloaded. Removes all the items in the collection that satisfy the condition defined by predicate.</td>
<td>(Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>RemoveAll</strong></td>
<td>Overloaded. Removes all the items in the collection that satisfy the condition defined by predicate.</td>
<td>(Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>RemoveMany</strong></td>
<td>Overloaded. Replaces all values associated with key with the single value.</td>
<td></td>
</tr>
<tr>
<td><strong>Replace</strong></td>
<td>Replaces all values associated with key with a new collection of values. If the collection does not permit duplicate values, and values has duplicate items, then only the last of duplicates is added.</td>
<td>(Inherited from MultiDictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>ReplaceMany</strong></td>
<td>Replaces all values associated with key with a new collection of values. If the collection does not permit duplicate values, and values has duplicate items, then only the last of duplicates is added.</td>
<td>(Inherited from MultiDictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>ToArray</strong></td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.</td>
<td>(Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.</td>
<td></td>
</tr>
<tr>
<td><strong>TrueForAll</strong></td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate.</td>
<td>(Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
</tbody>
</table>
# Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountAllValues</td>
<td>Gets a total count of values in the collection. This default implementation is slow; it enumerates all of the keys in the dictionary and calls CountValues on each. A derived class may be able to supply a more efficient implementation. (Inherited from MultiDictionaryBase(Of TKey, TValue&gt;).)</td>
</tr>
<tr>
<td>CountValues</td>
<td>Gets the number of values associated with a given key.</td>
</tr>
<tr>
<td>EnumerateKeys</td>
<td>Enumerate all the keys in the dictionary. (Overrides MultiDictionaryBase(Of TKey, TValue&gt;)...:EnumerateKeys().)</td>
</tr>
<tr>
<td>EqualValues</td>
<td>Determine if two values are equal.</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>TryEnumerateValuesForKey</td>
<td>Determines if this dictionary contains a key equal to key. If so, all the values associated with that key are returned through the values parameter.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDictionary(Of TKey, ICollection(Of TValue))::.Add</td>
<td>(Inherited from MultiDictionaryBase(Of TKey, TValue).)</td>
</tr>
<tr>
<td>IDictionary(Of TKey, ICollection(Of TValue))::.TryGetValue</td>
<td>(Inherited from MultiDictionaryBase(Of TKey, TValue).)</td>
</tr>
<tr>
<td>ICollection::.CopyTo</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase(Of T).)</td>
</tr>
<tr>
<td>IEnumerable::.GetEnumerator</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from CollectionBase(Of T).)</td>
</tr>
<tr>
<td>ICloneable::.Clone</td>
<td>Implements ICloneable.Clone. Makes a shallow clone of this dictionary; i.e., if keys or values are reference types, then they are not cloned.</td>
</tr>
</tbody>
</table>
See Also

MultiDictionary(Of TKey, TValue)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<(Of <TKey, TValue>)>...::Add Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add(KeyValuePair&lt;TKey, ICollection&lt;TValue&gt;&gt;)</strong></td>
<td>Adds a key-value pair to the collection. The value part of the pair must be a collection of values to associate with the key. If values are already associated with the given key, the new values are added to the ones associated with that key. (Inherited from <code>MultiDictionaryBase(Of TKey, TValue&gt;</code>.)</td>
</tr>
<tr>
<td><strong>Add(TKey, TValue)</strong></td>
<td>Adds a new value to be associated with a key. If duplicate values are permitted, this method always adds a new key-value pair to the dictionary. If duplicate values are not permitted, and key already has a value equal to value associated with it, then that value is replaced with value, and the number of values associate with key is unchanged.</td>
</tr>
</tbody>
</table>
See Also

MultiDictionary(Of TKey, TValue)> Class
MultiDictionary(Of TKey, TValue)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<(Of <TKey, TValue>)>::Add Method (TKey, TValue)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a new value to be associated with a key. If duplicate values are permitted, this method always adds a new key-value pair to the dictionary.

If duplicate values are not permitted, and key already has a value equal to value associated with it, then that value is replaced with value, and the number of values associate with key is unchanged.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public void Add(
    TKey key,
    TValue value
)

Visual Basic (Declaration)

Public Sub Add (_
    key As TKey, _
    value As TValue _
)

Visual C++

public:
    virtual void Add ( 
        TKey key, 
        TValue value
    ) sealed

Parameters

key
    TKey
    The key to associate with.

value
    TValue
    The value to associated with key.
See Also

MultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary(Of TKey, TValue)>...::Clear Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all keys and values from the dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public override sealed void Clear()
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Sub Clear
```

**Visual C++**

```cpp
public:
virtual void Clear () override sealed
```
See Also

MultiDictionary(Of TKey, TValue)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<Of TKey, TValue>>...::Clone Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a shallow clone of this dictionary; i.e., if keys or values of the dictionary are reference types, then they are not cloned. If TKey or TValue is a value type, then each element is copied as if by simple assignment.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public MultiDictionary<TKey, TValue> Clone()

Visual Basic (Declaration)

Public Function Clone As MultiDictionary(Of TKey, TValue)

Visual C++

public:
MultiDictionary<TKey, TValue>* Clone ()

Return Value

The cloned dictionary.
Remarks

Cloning the dictionary takes time $O(N)$, where $N$ is the number of key-value pairs in the dictionary.
See Also

MultiDictionary<Of <TKey, TValue>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary(Of <TKey, TValue>)::CloneContents Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a deep clone of this dictionary. A new dictionary is created with a clone of each entry of this dictionary, by calling ICloneable.Clone on each element. If TKey or TValue is a value type, then each element is copied as if by simple assignment.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public MultiDictionary<TKey, TValue> CloneContents()

Visual Basic (Declaration)

Public Function CloneContents As MultiDictionary(Of TKey, TValue)

Visual C++

public:
MultiDictionary<TKey, TValue>^ CloneContents ()

Return Value

The cloned dictionary.
Remarks

If TKey or TValue is a reference type, it must implement ICloneable. Otherwise, an InvalidOperationException is thrown.

Cloning the dictionary takes time $O(N \log N)$, where $N$ is the number of key-value pairs in the dictionary.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>TKey or TValue is a reference type that does not implement ICloneable.</td>
</tr>
</tbody>
</table>
See Also

MultiDictionary(Of TKey, TValue)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<(Of <TKey, TValue>)>...::Contains Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains(KeyValuePair(Of TKey, ICollection(Of TValue)))</td>
<td>Determines if this dictionary contains the given key and all of the values associated with that key. (Inherited from MultiDictionaryBase(Of TKey, TValue).)</td>
</tr>
<tr>
<td>Contains(TKey, TValue)</td>
<td>Checks to see if value is associated with key in the dictionary.</td>
</tr>
</tbody>
</table>
See Also

MultiDictionary(Of TKey, TValue)> Class
MultiDictionary(Of TKey, TValue)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary(Of TKey, TValue).::Contains Method (TKey, TValue)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Checks to see if value is associated with key in the dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public bool Contains(
    TKey key,
    TValue value
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function Contains (
    key As TKey, 
    value As TValue 
) As Boolean
```

**Visual C++**

```csharp
public:
virtual bool Contains ( 
    TKey key, 
    TValue value 
) sealed
```

Parameters

**key**

TKey
The key to check.

**value**

TValue
The value to check.

Return Value

True if value is associated with key.
See Also

MultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary(Of <TKey, TValue>)...::ContainsKey Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Checks to see if the key is present in the dictionary and has at least one value associated with it.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public bool ContainsKey(
    TKey key
)
```

Visual Basic (Declaration)

```vbnet
Public Function ContainsKey ( _
    key As TKey _
) As Boolean
```

Visual C++

```cpp
public:
    virtual bool ContainsKey ( 
    TKey key
)
```

Parameters

key

    TKey
    The key to check.

Return Value

True if key is present and has at least one value associated with it. Returns false otherwise.
See Also

MultiDictionary(Of TKey, TValue)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<(Of <TKey, TValue>)>::CountValues Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets the number of values associated with a given key.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntex

C#

protected int CountValues(TKey key)

Visual Basic (Declaration)

Protected Function CountValues(key As TKey) As Integer

Visual C++

protected:
virtual int CountValues(TKey key) sealed

Parameters

key
    TKey
    The key to count values of.

Return Value

The number of values associated with key. If key is not present in the dictionary, zero is returned.
See Also

MultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary(Of TKey, TValue)>...::EnumerateKeys Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerate all the keys in the dictionary.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

protected override sealed IEnumerator<TKey> EnumerateKeys()

Visual Basic (Declaration)

Protected Overrides NotOverridable Function EnumerateKeys As IEnumerator

Visual C++

protected:
virtual IEnumerator<TKey>^ EnumerateKeys () override sealed

Return Value

An IEnumerator<TKey> that enumerates all of the keys in the dictionary that have at least one value associated with them.
See Also

MultiDictionary(Of <TKey, TValue>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary(Of TKey, TValue)>...::EqualValues Method

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[Determine if two values are equal.]

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

protected bool EqualValues(
    TValue value1,
    TValue value2
)

Visual Basic (Declaration)

Protected Function EqualValues ( _
    value1 As TValue, _
    value2 As TValue _
) As Boolean

Visual C++

protected:
virtual bool EqualValues ( bool1 EqualValues ( bool1 EqualValues ( _
    value1 As TValue, _
    value2 As TValue _
) sealed

Parameters

value1
    TValue
    First value to compare.

value2
    TValue
    Second value to compare.

Return Value

True if the values are equal.
See Also

MultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<(Of <TKey, TValue>)>...::Remove Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove(TKey)</td>
<td>Removes a key and all associated values from the dictionary. If the key is not present in the dictionary, it is unchanged and false is returned.</td>
</tr>
<tr>
<td>Remove(TKey, TValue)</td>
<td>Removes a given value from the values associated with a key. If the last value is removed from a key, the key is removed also.</td>
</tr>
</tbody>
</table>
See Also

MultiDictionary(Of TKey, TValue)> Class
MultiDictionary(Of TKey, TValue)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<(Of <TKey, TValue>)>::Remove Method (TKey)

See Also

Visual Basic (Declaration) ▶ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes a key and all associated values from the dictionary. If the key is not present in the dictionary, it is unchanged and false is returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#
```csharp
public bool Remove(TKey key)
```

### Visual Basic (Declaration)
```vbnet
Public Function Remove(_
    key As TKey _) As Boolean
```

### Visual C++
```cpp
public:
    virtual bool Remove(_
        TKey key)
    sealed
```

## Parameters

key
```csharp
TKey
The key to remove.
```

## Return Value

True if the key was present and was removed. Returns false if the key was not present.
See Also

MultiDictionary<(Of <TKey, TValue>)) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<(Of TKey, TValue)>::Remove Method (TKey, TValue)

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes a given value from the values associated with a key. If the last value is removed from a key, the key is removed also.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public bool Remove(
    TKey key,
    TValue value
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function Remove ( _
    key As TKey, _
    value As TValue _
) As Boolean
```

**Visual C++**

```cpp
public:
    virtual bool Remove ( 
        TKey key, 
        TValue value
    ) sealed
```

### Parameters

**key**

- **TKey**
  - A key to remove a value from.

**value**

- **TValue**
  - The value to remove.

### Return Value

True if value was associated with key (and was therefore removed). False if value was not associated with key.
See Also

`MultiDictionary(Of <TKey, TValue>)` Class
`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
MultiDictionary<(Of <TKey, TValue>)>...::RemoveMany Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RemoveMany(IEnumerable&lt;TKey&gt;)</code></td>
<td>Remove all of the keys (and any associated values) in a collection of keys. If a key is not present in the dictionary, nothing happens.</td>
</tr>
<tr>
<td>(Inherited from <code>MultiDictionaryBase&lt;TKey, TValue&gt;</code>)</td>
<td>(Inherited from <code>MultiDictionaryBase&lt;TKey, TValue&gt;</code>)</td>
</tr>
<tr>
<td><code>RemoveMany(TKey, IEnumerable&lt;TValue&gt;)</code></td>
<td>Removes a collection of values from the values associated with a key. If the last value is removed from a key, the key is removed also.</td>
</tr>
<tr>
<td>(Inherited from <code>MultiDictionaryBase&lt;TKey, TValue&gt;</code>)</td>
<td>(Inherited from <code>MultiDictionaryBase&lt;TKey, TValue&gt;</code>)</td>
</tr>
</tbody>
</table>
See Also

MultiDictionary(Of TKey, TValue)> Class
MultiDictionary(Of TKey, TValue)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Implements ICloneable.Clone. Makes a shallow clone of this dictionary; i.e., if keys or values are reference types, then they are not cloned.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private Object ICloneable.Clone()

Visual Basic (Declaration)

Private Function System.ICloneable.Clone As Object Implements IClone

Visual C++

private:
virtual Object^ System.ICloneable.Clone () sealed = ICloneable::Clone

Return Value

The cloned dictionary.
See Also

MultiDictionary<Of <TKey, TValue>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<Of <TKey, TValue>>::TryEnumerateValuesForKey Method

See Also

Visual Basic (Declaration) ➔ Visual Basic (Usage)

C#

Visual C++

J#

JScript

XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this dictionary contains a key equal to key. If so, all the values associated with that key are returned through the values parameter.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
protected bool TryEnumerateValuesForKey(
    TKey key,
    out IEnumerator<TValue> values
)
```

### Visual Basic (Declaration)

```
Protected Function TryEnumerateValuesForKey (_
    key As TKey, _
    <OutAttribute> ByRef values As IEnumerable(Of TValue) _
) As Boolean
```

### Visual C++

```c++
protected:
virtual bool TryEnumerateValuesForKey (
    TKey key,
    [OutAttribute] IEnumerable<TValue>^% values
)
```

## Parameters

### key

- **Type**: TKey
- **Description**: The key to search for.

### values

- **Type**: IEnumerable<TValue>^%
- **Description**: Returns all values associated with key, if true was returned.

## Return Value

- **True**: If the dictionary contains key. False if the dictionary does not contain key.
See Also

MultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<Of <TKey, TValue>> Properties

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class MultiDictionary<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class MultiDictionary(Of TKey, TValue)

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class MultiDictionary
Type Parameters

TKey
TValue

The type exposes the following properties.
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Count**     | Gets the number of key-value pairs in the dictionary. Each value associated with a given key is counted. If duplicate values are permitted, each duplicate value is included in the count. 
(Overrides `MultiDictionaryBase(Of<TKey, TValue>)::.Count`.) |
| **Item**      | Returns a collection of all of the values in the dictionary associated with key, or changes the set of values associated with key. If the key is not present in the dictionary, an ICollection enumerating no values is returned. The returned collection of values is read-write, and can be used to modify the collection of values associated with the key.
(Inherited from `MultiDictionaryBase(Of<TKey, TValue>).`) |
| **KeyComparer** | Returns the IEqualityComparer<T> used to compare keys in this dictionary. |
| **Keys**      | Gets a read-only collection all the keys in this dictionary. 
(Inherited from `MultiDictionaryBase(Of<TKey, TValue>).`) |
| **KeyValuePairs** | Gets a read-only collection of all key-value pairs in the dictionary. If a key has multiple values associated with it, then a key-value pair is present for each value associated with the key. 
(Inherited from `MultiDictionaryBase(Of<TKey, TValue>).`) |
| **ValueComparer** | Returns the IEqualityComparer<T> used to compare values in this dictionary. |
| **Values**    | Gets a read-only collection of all the values in the dictionary. 
(Inherited from `MultiDictionaryBase(Of<TKey, TValue>).`) |
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;(Of </code>&lt;T<code>)&gt;&gt;::IsReadOnly (Inherited from </code>CollectionBase&lt;(Of <code>&lt;T</code>)&gt;).)`</td>
<td>(Inherited from <code>CollectionBase&lt;(Of </code>&lt;T<code>)&gt;).</code></td>
</tr>
<tr>
<td><code>IDictionary&lt;(Of </code>&lt;TKey, TValue&gt;<code>)&gt;::Item (Inherited from </code>MultiDictionaryBase&lt;(Of <code>&lt;TKey, TValue&gt;</code>)&gt;.)`</td>
<td>(Inherited from <code>MultiDictionaryBase&lt;(Of </code>&lt;TKey, TValue&gt;`)&gt;.)</td>
</tr>
<tr>
<td><code>ICollection&lt;(Of </code>&lt;TKey, TValue&gt;<code>)&gt;::Values (Inherited from </code>MultiDictionaryBase&lt;(Of <code>&lt;TKey, TValue&gt;</code>)&gt;.)`</td>
<td>Indicates whether the collection is synchronized. (Inherited from <code>CollectionBase&lt;(Of </code>&lt;T<code>)&gt;).</code></td>
</tr>
<tr>
<td><code>ICollection::IsSynchronized</code></td>
<td>Indicates the synchronization object for this collection. (Inherited from <code>CollectionBase&lt;(Of </code>&lt;T<code>)&gt;).</code></td>
</tr>
<tr>
<td><code>ICollection::SyncRoot</code></td>
<td></td>
</tr>
</tbody>
</table>

---

[Reference](#)
See Also

MultiDictionary(Of TKey, TValue>)
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary<(Of <TKey, TValue>)>...::Count Property

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets the number of key-value pairs in the dictionary. Each value associated with a given key is counted. If duplicate values are permitted, each duplicate value is included in the count.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed int Count{ get; }

Visual Basic (Declaration)

Public Overrides NotOverridable ReadOnly Property Count As Integer

Visual C++

public:
virtual property int Count {
int get () override sealed;
}

Field Value

The number of key-value pairs in the dictionary.
See Also

`MultiDictionary<Of <TKey, TValue>>` Class
`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
MultiDictionary(Of TKey, TValue)>...::KeyComparer Property

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the IEqualityComparer<T> used to compare keys in this dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public IEqualityComparer<TKey> KeyComparer{ get; }
```

**Visual Basic (Declaration)**

```
Public ReadOnly Property KeyComparer As IEqualityComparer(Of TKey)
```

**Visual C++**

```c++
public:
property IEqualityComparer<TKey>^ KeyComparer { IEqualityComparer<TKey>^ get (); }
```  

**Field Value**

If the dictionary was created using a comparer, that comparer is returned. Otherwise the default comparer for TKey (EqualityComparer<TKey>.Default) is returned.
See Also

MultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionary(Of TKey, TValue)>::ValueComparer Property

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the IEqualityComparer<T> used to compare values in this dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public IEqualityComparer<TValue> ValueComparer { get; }
```

**Visual Basic (Declaration)**

```vbnet
Public ReadOnly Property ValueComparer As IEqualityComparer(Of TValue)
```

**Visual C++**

```cpp
public:
property IEqualityComparer<TValue>^ ValueComparer { 
    IEqualityComparer<TValue>^ get ();
}
```

**Field Value**

If the dictionary was created using a comparer, that comparer is returned. Otherwise the default comparer for TValue (EqualityComparer<TValue>.Default) is returned.
See Also

MultiDictionary<Of <TKey, TValue>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<Of <TKey, TValue>> Class

See Also  Members

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

MultiDictionaryBase is a base class that can be used to more easily implement a class that associates multiple values to a single key. The class implements the generic IDictionary<TKey, ICollection<TValue>> interface.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

```csharp
[SerializableAttribute]
public abstract class MultiDictionaryBase<TKey, TValue> : CollectionBase<IKeyValuePair<TKey, ICollection<TValue>>>,
IEnumerable
```

Visual Basic (Declaration)

```vbnet
<SerializableAttribute> _
Public MustInherit Class MultiDictionaryBase(Of TKey, TValue) _
Inherits CollectionBase(Of KeyValuePair(Of TKey, ICollection<TValue>), _
ICollection(Of KeyValuePair(Of TKey, ICollection<TValue>), _
IEnumerable(Of KeyValuePair(Of TKey, ICollection<TValue>), _
IEnumerable
```

Visual C++

```cpp
[SerializableAttribute]
generic<typename TKey, typename TValue>
generic ref class MultiDictionaryBase abstract : public CollectionBas
IDictionary<TKey, ICollection<TValue>^>, ICollection<KeyValuePa
IEnumerable<KeyValuePair<TKey, ICollection<TValue>^>>, IEnumerable
```
Type Parameters

TKey
  The key type of the dictionary.
TValue
  The value type of the dictionary.
Remarks

To use MultiDictionaryBase as a base class, the derived class must override Count, Clear, Add, Remove(TKey), Remove(TKey,TValue), Contains(TKey,TValue), EnumerateKeys, and TryEnumerateValuesForKey.

It may wish consider overriding CountValues, CountAllValues, ContainsKey, and EqualValues, but these are not required.
Inheritance Hierarchy

System..:::Object
   Wintellect.PowerCollections..:::CollectionBase<Of <KeyValuePair<Of <TKey, ICollection<Of <TValue>>>>>>
   Wintellect.PowerCollections..:::MultiDictionaryBase<Of <TKey, TValue>>
      Wintellect.PowerCollections..:::OrderedMultiDictionary<Of <TKey, TValue>>
         Wintellect.PowerCollections..:::OrderedMultiDictionary<Of <TKey, TValue>>..:::View
            Wintellect.PowerCollections..:::MultiDictionary<Of <TKey, TValue>>
See Also

MultiDictionaryBase(Of TKey, TValue)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<(Of <TKey, TValue>)> Members

See Also Methods Properties Constructors Explicit Interface Implementations

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
- Syntax

**C#**

```csharp
[SerializableAttribute]
public abstract class MultiDictionaryBase<TKey, TValue>
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute> _
Public MustInherit Class MultiDictionaryBase(Of TKey, TValue)
```

**Visual C++**

```cpp
[SerializableAttribute]
genric<typename TKey, typename TValue> 
public ref class MultiDictionaryBase abstract
```
Type Parameters

TKey
TValue

The type exposes the following members.
## Protected Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultiDictionaryBase&lt;Of TKey, TValue&gt;&gt;</td>
<td></td>
</tr>
<tr>
<td>MultiDictionaryBase&lt;Of TKey, TValue&gt;&gt;</td>
<td></td>
</tr>
<tr>
<td>New</td>
<td></td>
</tr>
</tbody>
</table>
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Overloaded. Adds new values to be associated with a key. If duplicate values are permitted, this method always adds new key-value pairs to the dictionary.</td>
</tr>
<tr>
<td><strong>AddMany</strong></td>
<td>If duplicate values are not permitted, and key already has a value equal to one of values associated with it, then that value is replaced, and the number of values associate with key is unchanged.</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase(Of&lt;T&gt;).)</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Clears the dictionary. This method must be overridden in the derived class. (Overrides CollectionBase(Of(KeyValuePair(Of&lt;TKey, ICollection&lt;TValue&gt;&gt;)&gt;&gt;)::.Clear().)</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>ContainsKey</strong></td>
<td>Determines whether a given key is found in the dictionary.</td>
</tr>
<tr>
<td><strong>ConvertAll(Of&lt;TOutput&gt;)</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from CollectionBase(Of&lt;T&gt;).)</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the</td>
</tr>
</tbody>
</table>
provided array.
(Inherited from `CollectionBase<Of <T>>`.)

- **CountWhere**
  Counts the number of items in the collection that satisfy the condition defined by predicate.
  (Inherited from `CollectionBase<Of <T>>`.)

- **Count**
  Determines whether the specified `Object` is equal to the current `Object`.
  (Inherited from `Object`.)

- **Equals**
  Determines if the collection contains any item that satisfies the condition defined by predicate.
  (Inherited from `CollectionBase<Of <T>>`.)

- **FindAll**
  Enumerates the items in the collection that satisfy the condition defined by predicate.
  (Inherited from `CollectionBase<Of <T>>`.)

- **ForEach**
  Performs the specified action on each item in this collection.
  (Inherited from `CollectionBase<Of <T>>`.)

- **GetEnumerator**
  Enumerate all the keys in the dictionary, and for each key, the collection of values for that key.
  (Overrides `CollectionBase<Of <KeyValuePair<Of <TValue>>>>::GetEnumerator()`)

- **GetEnumerator()**
  Serves as a hash function for a particular type.

- **GetHashCode**
  `GetHashCode()` is suitable for use in hashing algorithms and data structures like a hash table.
  (Inherited from `Object`.)

- **GetType**
  Gets the `Type` of the current instance.
  (Inherited from `Object`.)

- **Remove**
  Overloaded.

- **RemoveAll**
  Removes all the items in the collection that satisfy the condition defined by predicate.
  (Inherited from `CollectionBase<Of <T>>`.)

- **RemoveMany**
  Overloaded.

- **Replace**
  Replaces all values associated with key with the single value value.

- **ReplaceMany**
  Replaces all values associated with key with a new collection of values. If the collection does not permit duplicate values, and values has duplicate items, then only
the last of duplicates is added.

Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from `CollectionBase<Of <T>>`.)

`ToArray` Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.

`ToString` (Overides `CollectionBase<Of <KeyValuePair<Of <TKey, ICollection<Of <TValue>>>>>>::ToString()`)

Determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from `CollectionBase<Of <T>>`.)
# Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CountAllValues</strong></td>
<td>Gets a total count of values in the collection. This default implementation is slow; it enumerates all of the keys in the dictionary and calls CountValues on each. A derived class may be able to supply a more efficient implementation.</td>
</tr>
<tr>
<td><strong>CountValues</strong></td>
<td>Gets a count of the number of values associated with a key. The default implementation is slow; it enumerators all of the values (using TryEnumerateValuesForKey) to count them. A derived class may be able to supply a more efficient implementation.</td>
</tr>
<tr>
<td><strong>EnumerateKeys</strong></td>
<td>Enumerate all the keys in the dictionary. This method must be overridden by a derived class. If the derived class does not use the default comparison for values, this methods should be overridden to compare two values for equality.</td>
</tr>
<tr>
<td><strong>EqualValues</strong></td>
<td>This is used for the correct implementation of ICollection.Contains on the Values and KeyValuePairs collections.</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>TryEnumerateValuesForKey</strong></td>
<td>Enumerate all of the values associated with a given key. This method must be overridden by the derived class. If the key exists and has values associated with it, an enumerator for those values is returned through values. If the key does not exist, false is returned.</td>
</tr>
</tbody>
</table>
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Gets the number of keys in the dictionary. This property must be overridden in the derived class.</td>
</tr>
<tr>
<td></td>
<td>(Overrides <code>CollectionBase&lt;TCollectionBase&lt;TKeyValuePair&lt;TKey, TValue&gt;&gt;&gt;::Count</code>)</td>
</tr>
<tr>
<td></td>
<td>Returns a collection of all of the values in the dictionary associated with key, or changes the set of values associated with key. If the key is not present in the dictionary, an ICollection enumerating no values is returned. The returned collection of values is read-write, and can be used to modify the collection of values associated with the key.</td>
</tr>
<tr>
<td>Item</td>
<td>Gets a read-only collection all the keys in this dictionary.</td>
</tr>
<tr>
<td>Keys</td>
<td>Gets a read-only collection of all key-value pairs in the dictionary. If a key has multiple values associated with it, then a key-value pair is present for each value associated with the key.</td>
</tr>
<tr>
<td>KeyValuePairs</td>
<td>Gets a read-only collection of all the values in the dictionary.</td>
</tr>
<tr>
<td>Values</td>
<td></td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;(Of &lt;T&gt;)&gt;</code>, <code>::IsReadOnly</code></td>
<td>(Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>IDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;</code>, <code>::Add</code></td>
<td></td>
</tr>
<tr>
<td><code>ICollection&lt;(Of &lt;TValue&gt;)&gt;</code>, <code>::Item</code></td>
<td></td>
</tr>
<tr>
<td><code>IDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;</code>, <code>::TryGetValue</code></td>
<td></td>
</tr>
<tr>
<td><code>ICollection&lt;(Of &lt;TValue&gt;)&gt;</code>, <code>::Values</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>ICollection,,::CopyTo</code></td>
<td>Indicates whether the collection is synchronized. (Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>ICollection,,::IsSynchronized</code></td>
<td>Indicates the synchronization object for this collection. (Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>ICollection,,::SyncRoot</code></td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the...</td>
</tr>
</tbody>
</table>
IEnumerable::GetEnumerator

IEnumerator<T> that was overridden by the derived classes to enumerate the members of the collection.
(Inherited from CollectionBase<Of <T>>.)
See Also

MultiDictionaryBase<(Of <TKey, TValue>)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<Of <TKey, TValue>> Constructor

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

protected MultiDictionaryBase()

Visual Basic (Declaration)

Protected Sub New

Visual C++

protected:
MultiDictionaryBase ()
See Also

`MultiDictionaryBase`< `(Of `<TKey, TValue>`)> Class
`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<Of TKey, TValue> Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#  

[SerializableAttribute]  
public abstract class MultiDictionaryBase<TKey, TValue>  

Visual Basic (Declaration)  

<SerializableAttribute> _  
Public MustInherit Class MultiDictionaryBase(Of TKey, TValue)  

Visual C++  

[SerializableAttribute]  
generic<typename TKey, typename TValue>  
public ref class MultiDictionaryBase abstract
Type Parameters

TKey
TValue

The type exposes the following methods.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Overloaded. Adds new values to be associated with a key. If duplicate values are permitted, this method always adds new key-value pairs to the dictionary.</td>
</tr>
<tr>
<td>AddMany</td>
<td>If duplicate values are not permitted, and key already has a value equal to one of values associated with it, then that value is replaced, and the number of values associate with key is unchanged. Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>AsReadOnly</td>
<td>Clear Clears the dictionary. This method must be overridden in the derived class. (Overides CollectionBase&lt;Of&lt;KeyValuePair&lt;Of&lt;TKey, ICollection&lt;Of&lt;TValue&gt;&gt;&gt;&gt;&gt;&gt;. )</td>
</tr>
<tr>
<td>Contains</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>ContainsKey</td>
<td>Determines whether a given key is found in the dictionary.</td>
</tr>
<tr>
<td>ConvertAll&lt;Of&lt;TOutput&gt;&gt;</td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the</td>
</tr>
</tbody>
</table>
provided array.
(Inherited from `CollectionBase<(Of 'T)>).)

**CountWhere**
Counts the number of items in the collection that satisfy the condition defined by predicate.
(Inherited from `CollectionBase<(Of 'T)>).)

**Equals**
Determines whether the specified `Object` is equal to the current `Object`.
(Inherited from `Object`.)

**Exists**
Determines if the collection contains any item that satisfies the condition defined by predicate.
(Inherited from `CollectionBase<(Of 'T)>).)

**FindAll**
Enumerates the items in the collection that satisfy the condition defined by predicate.
(Inherited from `CollectionBase<(Of 'T)>).)

**ForEach**
Performs the specified action on each item in this collection.
(Inherited from `CollectionBase<(Of 'T)>).)

**GetEnumerator**
Enumerates all the keys in the dictionary, and for each key, the collection of values for that key.
(Overrides `CollectionBase<(Of <KeyValuePair(Of 'TKey, ICollection<(Of 'TValue)>)>)>:GetEnumerator()`.)
Serves as a hash function for a particular type.

**GetHashCode**
`GetHashCode()` is suitable for use in hashing algorithms and data structures like a hash table.
(Inherited from `Object`.)

**GetType**
Gets the `Type` of the current instance.
(Inherited from `Object`.)

**Remove**
Overloaded.

**RemoveAll**
Removes all the items in the collection that satisfy the condition defined by predicate.
(Inherited from `CollectionBase<(Of 'T)>).)

**RemoveMany**
Overloaded.

**Replace**
Replaces all values associated with key with the single value.

**ReplaceMany**
Replaces all values associated with key with a new collection of values. If the collection does not permit duplicate values, and values has duplicate items, then only
the last of duplicates is added.
Creates an array of the correct size, and copies all the items
in the collection into the array, by calling CopyTo.
(Inherited from CollectionBase(Of T).)

- **ToArray**

Shows the string representation of the dictionary. The string
representation contains a list of the mappings in the
dictionary.
(Overrides CollectionBase(Of KeyValuePair(Of TKey, ICollection(Of TValue))).)

- **ToString**

Determines if all of the items in the collection satisfy the
condition defined by predicate.
(Inherited from CollectionBase(Of T).)
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountAllValues</td>
<td>Gets a total count of values in the collection. This default implementation is slow; it enumerates all of the keys in the dictionary and calls CountValues on each. A derived class may be able to supply a more efficient implementation.</td>
</tr>
<tr>
<td>CountValues</td>
<td>Gets a count of the number of values associated with a key. The default implementation is slow; it enumerators all of the values (using TryEnumerateValuesForKey) to count them. A derived class may be able to supply a more efficient implementation.</td>
</tr>
<tr>
<td>EnumerateKeys</td>
<td>Enumerate all the keys in the dictionary. This method must be overridden by a derived class. If the derived class does not use the default comparison for values, this methods should be overridden to compare two values for equality.</td>
</tr>
<tr>
<td>EqualValues</td>
<td>This is used for the correct implementation of ICollection.Contains on the Values and KeyValuePairs collections.</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>TryEnumerateValuesForKey</td>
<td>Enumerate all of the values associated with a given key. This method must be overridden by the derived class. If the key exists and has values associated with it, an enumerator for those values is returned through values. If the key does not exist, false is returned.</td>
</tr>
</tbody>
</table>
Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDictionary(Of TKey, ICollection(Of TValue)&gt;)..::Add IDictionary(Of TKey, ICollection(Of TValue)&gt;)..::TryGetValue</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase(Of T).) Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from CollectionBase(Of T).)</td>
</tr>
</tbody>
</table>
See Also

MultiDictionaryBase<(Of <TKey, TValue>)>  
Wintellect.PowerCollections Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
MultiDictionaryBase<
(Of <TKey, TValue>)>...::Add Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add(KeyValuePair(Of&lt;TKey, ICollection&lt;TValue&gt;&gt;)&gt;)</strong></td>
<td>Adds a key-value pair to the collection. The value part of the pair must be a collection of values to associate with the key. If values are already associated with the given key, the new values are added to the ones associated with that key. (Overrides <code>CollectionBase&lt;Of&lt;KeyValuePair&lt;Of&lt;TKey, ICollection&lt;TValue&gt;&gt;&gt;&gt;&gt;:Add(T).</code>)</td>
</tr>
<tr>
<td><strong>Add(TKey, TValue)</strong></td>
<td>Adds a new key-value pair to the dictionary. This method must be overridden in the derived class.</td>
</tr>
</tbody>
</table>
See Also

MultiDictionaryBase<Of <TKey, TValue>> Class
MultiDictionaryBase<Of <TKey, TValue>> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<Of <TKey, TValue>>::Add Method
(KeyValuePair<Of <TKey, ICollection<Of <TValue>>>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a key-value pair to the collection. The value part of the pair must be a collection of values to associate with the key. If values are already associated with the given key, the new values are added to the ones associated with that key.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
- **Syntax**

  **C#**

  ```csharp
  public override void Add( 
      KeyValuePair<TKey, ICollection<TValue>> item
  )
  ```

  **Visual Basic (Declaration)**

  ```vbnet
  Public Overrides Sub Add (_
      item As KeyValuePair(Of TKey, ICollection(Of TValue)) _
  )
  ```

  **Visual C++**

  ```cpp
  public:
  virtual void Add ( 
      KeyValuePair<TKey, ICollection<TValue>^> item
  ) override
  ```

**Parameters**

- **item**

  ```csharp
  KeyValuePair<(Of <TKey, ICollection>(Of <TValue>>))>
  ```

  A `KeyValuePair` contains the Key and Value collection to add.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase(Of TKey, TValue)>...::Add Method (TKey, TValue)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a new key-value pair to the dictionary. This method must be overridden in the derived class.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public abstract void Add(
    TKey key,
    TValue value
)
```

**Visual Basic (Declaration)**

```vbnet
Public MustOverride Sub Add ( _
    key As TKey, _
    value As TValue _
)
```

**Visual C++**

```cpp
public:
    virtual void Add ( 
        TKey key,
        TValue value
    ) abstract
```

**Parameters**

**key**

    TKey
    Key to add.

**value**

    TValue
    Value to associated with the key.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::ArgumentException</code></td>
<td>key is already present in the dictionary</td>
</tr>
</tbody>
</table>
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<Of <TKey, TValue>>::AddMany Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds new values to be associated with a key. If duplicate values are permitted, this method always adds new key-value pairs to the dictionary.

If duplicate values are not permitted, and key already has a value equal to one of values associated with it, then that value is replaced, and the number of values associate with key is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual void AddMany(
    TKey key,
    IEnumerable<TValue> values
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Sub AddMany (_
    key As TKey, _
    values As IEnumerable(Of TValue) _
)
```

**Visual C++**

```cpp
public:
    virtual void AddMany ( 
        TKey key,
        IEnumerable<TValue>^ values
    )
```

**Parameters**

**key**

- TKey
  - The key to associate with.

**values**

- IEnumerable(Of TValue>)
  - A collection of values to associate with key.
See Also

MultiDictionaryBase<(Of '<TKey, TValue>')> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<Of <TKey, TValue>>.::.Clear Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Clears the dictionary. This method must be overridden in the derived class.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public abstract void Clear()

Visual Basic (Declaration)

Public MustOverride Sub Clear

Visual C++

public:
virtual void Clear () abstract override
See Also

MultiDictionaryBase<(.Of <TKey, TValue>).> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<Of <TKey, TValue>>::Contains Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains(KeyValuePair&lt;TKey, ICollection&lt;TValue&gt;&gt;)</td>
<td>Determines if this dictionary contains the given key and all of the values associated with that key. (Overrides CollectionBase&lt;TKey, ICollection&lt;TValue&gt;&gt;::Contains(T)).</td>
</tr>
<tr>
<td>Contains(TKey, TValue)</td>
<td>Determines if this dictionary contains a key-value pair equal to key and value. The dictionary is not changed. This method must be overridden in the derived class.</td>
</tr>
</tbody>
</table>
See Also

MultiDictionaryBase<TKey, TValue> Class
MultiDictionaryBase<TKey, TValue> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<(Of <TKey, TValue>)>..:::Contains Method (KeyValuePair<(Of <TKey, ICollection<(Of <TValue>)>>)>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this dictionary contains the given key and all of the values associated with that key..

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override bool Contains(
    KeyValuePair<TKey, ICollection<TValue>> pair
)

Visual Basic (Declaration)

Public Overrides Function Contains ( _
    pair As KeyValuePair(Of TKey, ICollection(Of TValue)) _
) As Boolean

Visual C++

public:
    bool Contains ( 
        KeyValuePair<TKey, ICollection<TValue>^> pair
    ) override

Parameters

pair
    KeyValuePair<(Of <TKey, ICollection<(Of <TValue>)>>>)
    A key and collection of values to search for.

Return Value

True if the dictionary has associated all of the values in pair.Value with pair.Key.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<(Of <TKey, TValue>)>::Contains Method (TKey, TValue)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this dictionary contains a key-value pair equal to key and value. The dictionary is not changed. This method must be overridden in the derived class.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public abstract bool Contains(
    TKey key,
    TValue value
)
```

### Visual Basic (Declaration)

```vbnet
Public MustOverride Function Contains ( _
    key As TKey, _
    value As TValue _
) As Boolean
```

### Visual C++

```cpp
public:
    virtual bool Contains (
    TKey key,
    TValue value
) abstract
```

## Parameters

**key**
- TKey
  - The key to search for.

**value**
- TValue
  - The value to search for.

## Return Value

True if the dictionary has associated value with key.
See Also

**MultiDictionaryBase(Of TKey, TValue)>** Class  
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
MultiDictionaryBase<(Of <TKey, TValue>)>::ContainsKey Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines whether a given key is found in the dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual bool ContainsKey(TKey key)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function ContainsKey ( _
    key As TKey _
) As Boolean
```

**Visual C++**

```cpp
public:
    virtual bool ContainsKey ( 
    TKey key
    )
```

**Parameters**

`key`

`TKey`

Key to look for in the dictionary.

**Return Value**

True if the key is present in the dictionary.
Remarks

The default implementation simply calls TryEnumerateValuesForKey. It may be appropriate to override this method to provide a more efficient implementation.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase(Of TKey, TValue)>...::CountAllValues Method

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets a total count of values in the collection. This default implementation is slow; it enumerates all of the keys in the dictionary and calls CountValues on each. A derived class may be able to supply a more efficient implementation.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
protected virtual int CountAllValues()
```

**Visual Basic (Declaration)**

```vbnet
Protected Overridable Function CountAllValues As Integer
```

**Visual C++**

```cpp
protected:
virtual int CountAllValues ()
```

**Return Value**

The total number of values associated with all keys in the dictionary.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<(Of <TKey, TValue>)>::CountValues Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets a count of the number of values associated with a key. The default implementation is slow; it enumerators all of the values (using TryEnumerateValuesForKey) to count them. A derived class may be able to supply a more efficient implementation.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
protected virtual int CountValues(TKey key)
```

### Visual Basic (Declaration)

```vbnet
Protected Overridable Function CountValues (_
    key As TKey _
) As Integer
```

### Visual C++

```c++
protected:
virtual int CountValues ( TKey key
)
```

## Parameters

*key* `TKey`

The key to count values for.

## Return Value

The number of values associated with key.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase(Of TKey, TValue)>...:::EnumerateKeys Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerate all the keys in the dictionary. This method must be overridden by a derived class.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
protected abstract IEnumerable<TKey> EnumerateKeys()
```

**Visual Basic (Declaration)**

```vbnet
Protected MustOverride Function EnumerateKeys As IEnumerable(Of TKey)
```

**Visual C++**

```cpp
protected:
virtual IEnumerable<TKey>^ EnumerateKeys () abstract
```

**Return Value**

An IEnumerable<TKey> that enumerates all of the keys in the collection that have at least one value associated with them.
See Also

MultiDictionaryBase<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<(Of <TKey, TValue>)>::EqualValues Method

See Also

Visual Basic (Declaration) ▶ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

If the derived class does not use the default comparison for values, this methods should be overridden to compare two values for equality. This is used for the correct implementation of ICollection.Contains on the Values and KeyValuePairs collections.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

protected virtual bool EqualValues(
    TValue value1,
    TValue value2
)

Visual Basic (Declaration)

Protected Overridable Function EqualValues ( _
    value1 As TValue, _
    value2 As TValue _
) As Boolean

Visual C++

protected:
    virtual bool EqualValues ( 
    TValue value1,
    TValue value2
)

Parameters

value1
    TValue
    First value to compare.

value2
    TValue
    Second value to compare.

Return Value

True if the values are equal.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Enumerate all the keys in the dictionary, and for each key, the collection of values for that key.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public override IEnumerable<KeyValuePair<TKey, ICollection<TValue>>> GetEnumerator;

Visual Basic (Declaration)

Public Overrides Function GetEnumerator As IEnumerable(Of KeyValuePair(TKey, ICollection(TValue)))

Visual C++

public: virtual IEnumerable<KeyValuePair<TKey, ICollection<TValue>>> GetEnumerator;

Return Value

An enumerator to enumerate all the key, ICollection<value> pairs in the dictionary.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<Of <TKey, TValue>>...::Remove Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove(KeyValuePair&lt;Of &lt;TKey, ICollection&lt;Of &lt;TValue&gt;&gt;&gt;&gt;)&gt;</td>
<td>Removes a set of values from a given key. If all values associated with a key are removed, then the key is removed also. (Overrides CollectionBase&lt;Of &lt;KeyValuePair&lt;Of &lt;TKey, ICollection&lt;Of &lt;TValue&gt;&gt;&gt;&gt;&gt;&gt;&gt;::Remove(T).)</td>
</tr>
<tr>
<td>Remove(TKey)</td>
<td>Removes a key from the dictionary. This method must be overridden in the derived class.</td>
</tr>
<tr>
<td>Remove(TKey, TValue)</td>
<td>Removes a key-value pair from the dictionary. This method must be overridden in the derived class.</td>
</tr>
</tbody>
</table>
See Also

MultiDictionaryBase<(Of <TKey, TValue>)> Class
MultiDictionaryBase<(Of <TKey, TValue>)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase(Of TKey, TValue)::.Remove Method
(KeyValuePair(Of TKey, ICollection(Of TValue))>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes a set of values from a given key. If all values associated with a key are removed, then the key is removed also.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public override bool Remove(
    KeyValuePair<TKey, ICollection<TValue>> pair
)

Visual Basic (Declaration)

Public Overrides Function Remove ( _
   pair As KeyValuePair(Of TKey, ICollection(Of TValue)) _
) As Boolean

Visual C++

public:
    virtual bool Remove (    
        KeyValuePair<TKey, ICollection<TValue>^> pair
    ) override

Parameters

pair
    KeyValuePair<(Of <TKey, ICollection<(Of <TValue>)>>)>
    A KeyValuePair contains a key and a set of values to remove from that key.

Return Value

True if at least one values was found and removed.
See Also

MultiDictionaryBase<Of <TKey, TValue>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<Of <TKey, TValue>>..:::Remove Method (TKey)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes a key from the dictionary. This method must be overridden in the derived class.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public abstract bool Remove(TKey key)
```

#### Visual Basic (Declaration)

```vbnet
Public MustOverride Function Remove(_
    key As TKey _) As Boolean
```

#### Visual C++

```cpp
public:
    virtual bool Remove(TKey key) abstract
```

### Parameters

**key**

TKey

Key to remove from the dictionary.

### Return Value

True if the key was found, false otherwise.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<Of <TKey, TValue>>::Remove Method (TKey, TValue)

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes a key-value pair from the dictionary. This method must be overridden in the derived class.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public abstract bool Remove(
    TKey key,
    TValue value
)
```

### Visual Basic (Declaration)

```vbnet
Public MustOverride Function Remove ( _
    key As TKey, _
    value As TValue _
) As Boolean
```

### Visual C++

```cpp
public:
virtual bool Remove (  
    TKey key,  
    TValue value  
) abstract
```

## Parameters

**key**

- **TKey**
  - Key to remove from the dictionary.

**value**

- **TValue**
  - Associated value to remove from the dictionary.

## Return Value

True if the key-value pair was found, false otherwise.
See Also

**MultiDictionaryBase(Of TKey, TValue) Class**
**Wintellect.PowerCollections Namespace**

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
MultiDictionaryBase<
(Of <TKey, TValue>)>
:::RemoveMany Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveMany(IEnumerable&lt;TKey&gt;)</td>
<td>Remove all of the keys (and any associated values) in a collection of keys. If a key is not present in the dictionary, nothing happens.</td>
</tr>
<tr>
<td>RemoveMany(TKey, IEnumerable&lt;TValue&gt;)</td>
<td>Removes a collection of values from the values associated with a key. If the last value is removed from a key, the key is removed also.</td>
</tr>
</tbody>
</table>
See Also

MultiDictionaryBase(Of TKey, TValue) Class
MultiDictionaryBase(Of TKey, TValue) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Remove all of the keys (and any associated values) in a collection of keys. If a key is not present in the dictionary, nothing happens.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public int RemoveMany(
    IEnumerable<TKey> keyCollection
)
```

### Visual Basic (Declaration)

```vbnet
Public Function RemoveMany (_
    keyCollection As IEnumerable(Of TKey) _
) As Integer
```

### Visual C++

```cpp
public:
    int RemoveMany (  
        IEnumerable<TKey>^ keyCollection
    )
```

## Parameters

- **keyCollection**
  ```csharp
  IEnumerable<TKey>
  ```
  A collection of key values to remove.

## Return Value

The number of keys from the collection that were present and removed.
See Also

`MultiDictionaryBase<Of <TKey, TValue>>` Class
`Wintellect.PowerCollections` Namespace

Send [comments](#) about this topic to Microsoft.
MultiDictionaryBase<(Of <TKey, TValue>)>...:::RemoveMany Method (TKey, IEnumerable<(Of <TValue>)>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes a collection of values from the values associated with a key. If the last value is removed from a key, the key is removed also.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public virtual int RemoveMany(
    TKey key,
    IEnumerable<TValue> values
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function RemoveMany ( _
    key As TKey, _
    values As IEnumerable(Of TValue) _
) As Integer
```

**Visual C++**

```cpp
public:
    virtual int RemoveMany ( 
        TKey key,
        IEnumerable<TValue>^ values
    )
```

### Parameters

**key**

- **TKey**
- A key to remove values from.

**values**

- `IEnumerable<Of <TValue>>`
- A collection of values to remove.

### Return Value

The number of values that were present and removed.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<TKey, TValue>::Replace Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Replaces all values associated with key with the single value value.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual bool Replace(
    TKey key,
    TValue value
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function Replace ( _
    key As TKey, _
    value As TValue _
) As Boolean
```

**Visual C++**

```cpp
public:
    virtual bool Replace ( 
        TKey key, 
        TValue value 
    )
```

**Parameters**

**key**

- TKey
- The key to associate with.

**value**

- TValue
- The new values to be associated with key.

**Return Value**

Returns true if some values were removed. Returns false if key was not present in the dictionary before Replace was called.
Remarks

This implementation simply calls Remove, followed by Add.
See Also

MultiDictionaryBase<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase(Of TKey, TValue).ReplaceMany Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Replaces all values associated with key with a new collection of values. If the collection does not permit duplicate values, and values has duplicate items, then only the last of duplicates is added.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public bool ReplaceMany(
    TKey key,
    IEnumerable<TValue> values
)
```

### Visual Basic (Declaration)

```vbnet
Public Function ReplaceMany (_
    key As TKey, _
    values As IEnumerable(Of TValue) _
) As Boolean
```

### Visual C++

```cpp
public:
    bool ReplaceMany ( 
        TKey key,
        IEnumerable<TValue>* values
    )
```

## Parameters

**key**

- **TKey**
  - The key to associate with.

**values**

- `IEnumerable<Of TValue>`
  - The new values to be associated with key.

## Return Value

Returns true if some values were removed. Returns false if key was not present in the dictionary before Replace was called.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<(Of <TKey,
Method
See Also

[This topic is pre-release documentation and is subject to change in future
releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private void IDictionary<TKey, ICollection<TValue>>.Add(TKey key, ICollection<TValue> values)

Visual Basic (Declaration)

Implements System.Collections.Generic.IDictionary(Of TKey, System.Collections.Generic.ICollection(Of TValue)).Add

Visual C++

private:
virtual void System.Collections.Generic.IDictionary<TKey, System.Collections.Generic.ICollection<TValue>>::Add(TKey key, ICollection<TValue>^ values)

Parameters

key
  TKey

values
  ICollection<(Of <TValue>)>
See Also

*MultiDictionaryBase(Of TKey, TValue)* Class
*Wintellect.PowerCollections* Namespace

Send [comments](#) about this topic to Microsoft.
MultiDictionaryBase<
(Of <TKey,
Method
See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.] 

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
private bool IDictionary<TKey, ICollection<TValue>>.TryGetValue(TKey key, out ICollection<TValue> values)
```

**Visual Basic (Declaration)**

```vbnet
                <OutAttribute> ByRef values As ICollection(Of TValue) _
) As Boolean Implements IDictionary(Of TKey, ICollection(Of TValue)),
```

**Visual C++**

```cpp
private:
virtual bool System.Collections.Generic.IDictionary<TKey,System.Collections.Generic.ICollection<TValue>>::TryGetValue(TKey key, _
                [OutAttribute] ICollection<TValue>^ values
) sealed = IDictionary<TKey, ICollection<TValue>^>::TryGetValue
```

### Parameters

**key**

- TKey

**values**

- ICollection<(Of TValue)>%
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<(Of <TKey, TValue>)>::ToString Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public override string ToString()
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides Function ToString As String
```

**Visual C++**

```cpp
public:
virtual String^ ToString () override
```

**Return Value**

The string representation of the dictionary.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<
(Of <TKey, TValue>)>...::TryEnumerateValuesForKey
Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerate all of the values associated with a given key. This method must be overridden by the derived class. If the key exists and has values associated with it, an enumerator for those values is returned through values. If the key does not exist, false is returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
protected abstract bool TryEnumerateValuesForKey(TKey key,
    out IEnumerable<TValue> values)
```

**Visual Basic (Declaration)**

```vbnet
Protected MustOverride Function TryEnumerateValuesForKey(_
    _key As TKey,_
    _<OutAttribute> ByRef values As IEnumerable(Of TValue)_
) As Boolean
```

**Visual C++**

```cpp
protected:
    virtual bool TryEnumerateValuesForKey (TKey key,_
        [OutAttribute] IEnumerable<TValue>^% values
    ) abstract
```

**Parameters**

key

TKey

The key to get values for.

values

IEnumerable<(Of <TValue>)>%

If true is returned, this parameter receives an enumerators that enumerates the values associated with that key.

**Return Value**

True if the key exists and has values associated with it. False otherwise.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<(Of <TKey, TValue>)> Properties

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Syntax

### C#

```csharp
[SerializableAttribute]
public abstract class MultiDictionaryBase<TKey, TValue>
```

### Visual Basic (Declaration)

```vbnet
<SerializableAttribute> _
Public MustInherit Class MultiDictionaryBase(Of TKey, TValue)
```

### Visual C++

```cpp
[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class MultiDictionaryBase abstract
```
**Type Parameters**

**TKey**
**TValue**

The type exposes the following properties.
Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Gets the number of keys in the dictionary. This property must be overridden in the derived class. (Overrides <code>CollectionBase&lt;Of &lt;KeyValuePair&lt;Of &lt;TKey, ICollection&lt;Of &lt;TValue&gt;&gt;&gt;&gt;&gt;&gt;::Count.) Returns a collection of all the values in the dictionary associated with key, or changes the set of values associated with key. If the key is not present in the dictionary, an </code>IICollection` enumerating no values is returned. The returned collection of values is read-write, and can be used to modify the collection of values associated with the key.</td>
</tr>
<tr>
<td>Item</td>
<td>Gets a read-only collection all the keys in this dictionary.</td>
</tr>
<tr>
<td>Keys</td>
<td>Gets a read-only collection of all key-value pairs in the dictionary. If a key has multiple values associated with it, then a key-value pair is present for each value associated with the key.</td>
</tr>
<tr>
<td>KeyValuePairs</td>
<td>Gets a read-only collection of all the values in the dictionary.</td>
</tr>
<tr>
<td>Values</td>
<td></td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;Of &lt;T&gt;&gt;</code>::IsReadOnly</td>
<td>(Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IDictionary&lt;Of &lt;TKey, ICollection&lt;Of &lt;TValue&gt;&gt;&gt;</code>::Item</td>
<td>Indicates whether the collection is synchronized. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IDictionary&lt;Of &lt;TKey, ICollection&lt;Of &lt;TValue&gt;&gt;&gt;</code>::Values</td>
<td>Indicates the synchronization object for this collection. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection</code>::IsSynchronized</td>
<td></td>
</tr>
<tr>
<td><code>ICollection</code>::SyncRoot</td>
<td></td>
</tr>
</tbody>
</table>
See Also

MultiDictionaryBase(Of TKey, TValue)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase(Of TKey, TValue)::.Count Property

See Also

Visual Basic (Declaration) ▸ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets the number of keys in the dictionary. This property must be overridden in the derived class.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public abstract int Count { get; }

Visual Basic (Declaration)

Public MustOverride ReadOnly Property Count As Integer

Visual C++

public:
virtual property int Count {
    int get () abstract override;
}


See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<
(Of <TKey, TValue>)
...::Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a collection of all of the values in the dictionary associated with key, or changes the set of values associated with key. If the key is not present in the dictionary, an ICollection enumerating no values is returned. The returned collection of values is read-write, and can be used to modify the collection of values associated with the key.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual ICollection<TValue> this[TKey key] {
    get; set;
}
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Default Property Item ( _
    key As TKey _
) As ICollection(Of TValue)
```

**Visual C++**

```cpp
public: property ICollection<TValue>^ default[TKey key] { 
    ICollection<TValue>^ get (TKey key); 
    void set (TKey key, ICollection<TValue>^ value); 
}
```

**Parameters**

- **key**
  - TKey
  - The key to get the values associated with.

**Field Value**

An ICollection<TValue> with all the values associated with key.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<(Of <TKey, TValue>)>::Keys Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets a read-only collection all the keys in this dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual ICollection<TKey> Keys{
    get;
}

Visual Basic (Declaration)

Public Overridable ReadOnly Property Keys As ICollection(Of TKey)

Visual C++

public:
    virtual property ICollection<TKey>^ Keys {
        ICollection<TKey>^ get ();
    }

Field Value

An readonly ICollection<TKey> of all the keys in this dictionary.
See Also

**MultiDictionaryBase(Of TKey, TValue) Class**
**Wintellect.PowerCollections Namespace**

Send [comments](#) about this topic to Microsoft.
Gets a read-only collection of all key-value pairs in the dictionary. If a key has multiple values associated with it, then a key-value pair is present for each value associated with the key.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual ICollection<KeyValuePair<TKey, TValue>> KeyValuePairs;

Visual Basic (Declaration)

Public Overridable ReadOnly Property KeyValuePairs As ICollection(Of TValue)

Visual C++

public:
virtual property ICollection<KeyValuePair<TKey, TValue>>^ KeyValuePairs;
ICollection<KeyValuePair<TKey, TValue>>^ get ()
}
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<
(TKey,
Property

See Also

- Visual Basic (Declaration) - Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
private ICollection<TValue> IDictionary<TKey, ICollection<TValue>>.Item<TKey key
}{ get; set; }
```

#### Visual Basic (Declaration)

```vbnet
key As TKey _
) As ICollection(Of TValue) Implements IDictionary(Of TKey, ICollection)

private: property ICollection^ System.Collections.Generic.IDictionary<TKey, ICollection^ get (TKey key) sealed = IDictionary^ TKey void set (TKey key, ICollection^ value) sealed = IDictionary

Parameters

key
TKey
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private ICollection<ICollection<TValue>> IDictionary<TKey, ICollection<TValue>>

Private ReadOnly Property System.Collections.Generic.IDictionary<TKey, ICollection<TValue>^>

private: virtual property ICollection<ICollection<TValue>^>^ System.Collections.Generic.IDictionary<TKey, ICollection<TValue>^> get () sealed = IDictionary<TKey, ICollection<TValue>^>^
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase<(Of <TKey, TValue>)>...::Values Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets a read-only collection of all the values in the dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

**C#**

```csharp
public virtual ICollection<TValue> Values{ get; }
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable ReadOnly Property Values As ICollection(Of TValue)
```

**Visual C++**

```cpp
public:
virtual property ICollection<TValue>^ Values {
    ICollection<TValue>^ get();
}
```

**Return Value**

A read-only ICollection<TValue> of all the values in the dictionary.
See Also

MultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>> Class

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

OrderedBag<T> is a collection that contains items of type T. The item are maintained in a sorted order. Unlike a OrderedSet, duplicate items (items that compare equal to each other) are allows in an OrderedBag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
[SerializableAttribute]
public class OrderedBag<T> : CollectionBase<T>, ICloneable
```

Visual Basic (Declaration)

```vbnet
<SerializableAttribute>
Public Class OrderedBag(Of T) _
    Inherits CollectionBase(Of T) _
    Implements ICloneable
```

Visual C++

```cpp
[SerializableAttribute] generic<typename T>
public ref class OrderedBag : public CollectionBase<T>, ICloneable
```
Type Parameters

T
Remarks

The items are compared in one of three ways. If T implements IComparable<TKey> or IComparable, then the CompareTo method of that interface will be used to compare items. Alternatively, a comparison function can be passed in either as a delegate, or as an instance of IComparer<TKey>.

OrderedBag is implemented as a balanced binary tree. Inserting, deleting, and looking up an element all are done in log(N) + M time, where N is the number of keys in the tree, and M is the current number of copies of the element being handled.

Bag(Of<T>) is similar, but uses hashing instead of comparison, and does not maintain the keys in sorted order.
Inheritance Hierarchy

**System::Object**

**Wintellect.PowerCollections::CollectionBase<>**

**Wintellect.PowerCollections::OrderedBag<>**
See Also

OrderedBag(Of T) Members
Wintellect.PowerCollections Namespace
Wintellect.PowerCollections::Bag(Of T)

Send comments about this topic to Microsoft.
OrderedBag<Of <T>> Members

See Also  Methods  Properties  Constructors  Explicit Interface Implementations

☐  Include Inherited Members ☐  Include Protected Members
☐  .NET Compact Framework Members Only
☐  XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

```csharp
[SerializableAttribute]
public class OrderedBag<T>
```

Visual Basic (Declaration)

```vbnet
<SerializableAttribute> _
Public Class OrderedBag(Of T)
```

Visual C++

```cpp
[SerializableAttribute]
generic<typename T>
public ref class OrderedBag
```
Type Parameters

T

The type exposes the following members.
# Public Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OrderedBag&lt;Of &lt;T&gt;&gt;OrderedBag&lt;Of &lt;T&gt;&gt;New</code></td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Must be overridden to allow adding items to this collection.</td>
</tr>
<tr>
<td>AddMany</td>
<td>Adds all the items in collection to the bag. Get a read-only list view of the items in this ordered bag. The items in the list are in sorted order, with the smallest item at index 0. This view does not copy any data, and reflects any changes to the underlying OrderedBag.</td>
</tr>
<tr>
<td>AsList</td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>AsReadOnly</td>
<td>Removes all items from the bag. (Overides ICollection&lt;Of&lt;T&gt;&gt;...::Clear())</td>
</tr>
<tr>
<td>Clear</td>
<td>Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td>Clone</td>
<td>Makes a deep clone of this bag. A new bag is created with a clone of each element of this bag, by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td>CloneContents</td>
<td>Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using</td>
</tr>
</tbody>
</table>
**ConvertAll<Of TOutput>**
Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.
(Inherited from CollectionBase<Of T>.)

**CopyTo**
Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.
(Inherited from CollectionBase<Of T>.)

**CountWhere**
Counts the number of items in the collection that satisfy the condition defined by predicate.
(Inherited from CollectionBase<Of T>.)

**Difference**
Computes the difference of this bag with another bag. The difference of these two bags is all items that appear in this bag, but not in otherBag. If an item appears X times in this bag, and Y times in the other bag, the difference contains the item X - Y times (zero times if Y >= X). A new bag is created with the difference of the bags and is returned. This bag and the other bag are unchanged.

**DifferenceWith**
Computes the difference of this bag with another bag. The difference of these two bags is all items that appear in this bag, but not in otherBag. If an item appears X times in this bag, and Y times in the other bag, the difference contains the item X - Y times (zero times if Y >= X). This bag receives the difference of the two bags; the other bag is unchanged.

**DistinctItems**
Enumerates all the items in the bag, but enumerates equal items just once, even if they occur multiple times in the bag.

**Equals**
Determines whether the specified Object is equal to the current Object.
(Inherited from **Object**.)

- **Exists**
  Determines if the collection contains any item that satisfies the condition defined by predicate.
  (Inherited from **CollectionBase**<(**Of** **<T>**)>.)

- **FindAll**
  Enumerates the items in the collection that satisfy the condition defined by predicate.
  (Inherited from **CollectionBase**<(**Of** **<T>**)>.)

- **ForEach**
  Performs the specified action on each item in this collection.
  (Inherited from **CollectionBase**<(**Of** **<T>**)>.)

- **GetEnumerator**
  Returns an enumerator that enumerates all the items in the bag. The items are enumerated in sorted order.
  (Overides **CollectionBase**<(**Of** **<T>**)>...:**GetEnumerator**().)

Enumerates all of the items in this bag that are equal to item, according to the comparison mechanism that was used when the bag was created. The bag is not changed.

- **GetEqualItems**
  If the bag does contain an item equal to item, then the enumeration contains no items.

- **GetFirst**
  Returns the first item in the bag: the item that would appear first if the bag was enumerated. This is also the smallest item in the bag.

Serves as a hash function for a particular type. **GetHashCode**() is suitable for use in hashing algorithms and data structures like a hash table.

(Inherited from **Object**.)

- **GetHashCode**

- **GetLast**
  Returns the last item in the bag: the item that would appear last if the bag was enumerated. This is also the largest item in the bag.

- **GetType**
  Gets the **Type** of the current instance.
  (Inherited from **Object**.)

Get the index of the given item in the sorted order.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IndexOf</strong></td>
<td>The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1. If multiple equal items exist, the smallest index of the equal items is returned.</td>
</tr>
<tr>
<td><strong>Intersection</strong></td>
<td>Computes the intersection of this bag with another bag. The intersection of two bags is all items that appear in both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item Minimum(X,Y) times. A new bag is created with the intersection of the bags and is returned. This bag and the other bag are unchanged.</td>
</tr>
<tr>
<td><strong>IntersectionWith</strong></td>
<td>Computes the intersection of this bag with another bag. The intersection of two bags is all items that appear in both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item Minimum(X,Y) times. This bag receives the intersection of the two bags, the other bag is unchanged.</td>
</tr>
<tr>
<td><strong>IsDisjointFrom</strong></td>
<td>Determines if this bag is disjoint from another bag. Two bags are disjoint if no item from one set is equal to any item in the other bag.</td>
</tr>
<tr>
<td><strong>IsEqualTo</strong></td>
<td>Determines if this bag is equal to another bag. This bag is equal to otherBag if they contain the same items, each the same number of times.</td>
</tr>
<tr>
<td><strong>IsProperSubsetOf</strong></td>
<td>Determines if this bag is a proper subset of another bag. Neither bag is modified. This bag is a subset of otherBag if every element in this bag is also in otherBag, at least the same number of times. Additional, this bag must have strictly fewer items than otherBag.</td>
</tr>
<tr>
<td><strong>IsProperSupersetOf</strong></td>
<td>Determines if this bag is a proper superset of another bag. Neither bag is modified. This bag is a proper superset of otherBag if every element in otherBag is also in this bag, at least the same number of times. Additional, this bag must have strictly more items than otherBag.</td>
</tr>
</tbody>
</table>
**IsSubsetOf**

Determines if this bag is a subset of another bag. Neither bag is modified. This bag is a subset of otherBag if every element in this bag is also in otherBag, at least the same number of times.

**IsSupersetOf**

Determines if this bag is a superset of another bag. Neither bag is modified. This bag is a superset of otherBag if every element in otherBag is also in this bag, at least the same number of times.

**LastIndexOf**

Get the index of the given item in the sorted order. The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1. If multiple equal items exist, the largest index of the equal items is returned.

**NumberOfCopies**

Returns the number of copies of item in the bag. More precisely, returns the number of items in the bag that compare equal to item.

**Range**

Returns a View collection that can be used for enumerating a range of the items in the bag. Only items that are greater than from and less than to are included. The items are enumerated in sorted order. Items equal to the end points of the range can be included or excluded depending on the fromInclusive and toInclusive parameters.

**RangeFrom**

Returns a View collection that can be used for enumerating a range of the items in the bag. Only items that are greater than (and optionally, equal to) from are included. The items are enumerated in sorted order. Items equal to from can be included or excluded depending on the fromInclusive parameter.

**RangeTo**

Returns a View collection that can be used for enumerating a range of the items in the bag. Only items that are less than (and optionally, equal to) to are included. The items are enumerated in sorted order. Items equal to to can be included or excluded depending on the toInclusive parameter.

**Remove**

Must be overridden to allow removing items from
this collection.

- **RemoveAll**
  Removes all the items in the collection that satisfy the condition defined by predicate.
  (Inherited from `CollectionBase<Of <T>>`.)

- **RemoveAllCopies**
  Searches the bag for all items equal to item, and removes all of them from the bag. If not found, the bag is unchanged.

- **RemoveFirst**
  Removes the first item in the bag. This is also the smallest item in the bag.

- **RemoveLast**
  Removes the last item in the bag. This is also the largest item in the bag.

- **RemoveMany**
  Removes all the items in collection from the bag.
  Items not present in the bag are ignored.

- **Reversed**
  Returns a View collection that can be used for enumerating the items in the bag in reversed order.

- **Sum**
  Computes the sum of this bag with another bag. The sum of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item (X+Y) times. A new bag is created with the sum of the bags and is returned. This bag and the other bag are unchanged.

- **SumWith**
  Computes the sum of this bag with another bag. The sum of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item (X+Y) times. This bag receives the sum of the two bags, the other bag is unchanged.

- **SymmetricDifference**
  Computes the symmetric difference of this bag with another bag. The symmetric difference of two bags is all items that appear in either of the bags, but not both. If an item appears X times in one bag, and Y times in the other bag, the symmetric difference contains the item AbsoluteValue(X - Y times). A new bag is created with the symmetric difference of the bags and is returned. This bag and the other bag are
SymmetricDifferenceWith

Computes the symmetric difference of this bag with another bag. The symmetric difference of two bags is all items that appear in either of the bags, but not both. If an item appears X times in one bag, and Y times in the other bag, the symmetric difference contains the item AbsoluteValue(X - Y times). This bag receives the symmetric difference of the two bags; the other bag is unchanged.

ToArray

Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.
(Inherited from CollectionBase<Of<T>>.)

ToString

Shows the string representation of the collection. The string representation contains a list of the items in the collection. Contained collections (except string) are expanded recursively.
(Inherited from CollectionBase<Of<T>>.)

TrueForAll

Determines if all of the items in the collection satisfy the condition defined by predicate.
(Inherited from CollectionBase<Of<T>>.)

Union

Computes the union of this bag with another bag. The union of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the union contains the item Maximum(X,Y) times. A new bag is created with the union of the bags and is returned. This bag and the other bag are unchanged.

UnionWith

Computes the union of this bag with another bag. The union of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the union contains the item Maximum(X,Y) times. This bag receives the union of the two bags, the other bag is unchanged.
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
<td></td>
</tr>
</tbody>
</table>
### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comparer</strong></td>
<td>Returns the IComparer&lt;T&gt; used to compare items in this bag.</td>
</tr>
<tr>
<td><strong>Count</strong></td>
<td>Returns the number of items in the bag. (Overrides CollectionBase&lt;Of&lt;T&gt;&gt;::Count.)</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Get the item by its index in the sorted order. The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `ICollection<Of `T`>`.::IsReadOnly        | (Inherited from `CollectionBase<Of `T`>`.)
                                                                                                                                                  |
|                                          | Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from `CollectionBase<Of `T`>`.)
|                                          | Indicates whether the collection is synchronized. (Inherited from `CollectionBase<Of `T`>`.)
|                                          | Indicates the synchronization object for this collection. (Inherited from `CollectionBase<Of `T`>`.)
                                                                                                                                                  |
| `ICollection<Of `T`>`.::SyncRoot         | Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator<`T`> that was overridden by the derived classes to enumerate the members of the collection. (Inherited from `CollectionBase<Of `T`>`.)
|                                          | Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If `T` is a value type, then each element is copied as if by simple assignment. |
| `IEnumerable<Of `T`>`.::GetEnumerator |                                                                                                                                                                                                            |
|                                          |                                                                                                                                                                                                            |
| `ICloneable<Of `T`>`.::Clone             |                                                                                                                                                                                                            |
See Also

OrderedBag<Of <T>>

Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<(Of <T>)> Constructor

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderedBag(Of T)</td>
<td>Creates a new OrderedBag. The T must implement IComparable&lt;T&gt; or IComparable. The CompareTo method of this interface will be used to compare items in this bag.</td>
</tr>
<tr>
<td>OrderedBag(Of T) OrderedBag(Of T) New(IComparer(Of T))</td>
<td>Creates a new OrderedBag. The Compare method of the passed comparison object will be used to compare items in this bag.</td>
</tr>
<tr>
<td>OrderedBag(Of T) OrderedBag(Of T) New(IEnumerable(Of T))</td>
<td>Creates a new OrderedBag. The T must implement IComparable&lt;T&gt; or IComparable. The CompareTo method of this interface will be used to compare items in this bag. The bag is initialized with all the items in the given collection.</td>
</tr>
<tr>
<td>OrderedBag(Of T) OrderedBag(Of T) New(Comparison(Of T))</td>
<td>Creates a new OrderedBag. The passed delegate will be used to compare items in this bag.</td>
</tr>
<tr>
<td>OrderedBag(Of T) OrderedBag(Of T) New(IEnumerable(Of T), IComparer(Of T))</td>
<td>Creates a new OrderedBag. The Compare method of the passed comparison object will be used to compare items in this bag. The bag is initialized with all the items in the given collection.</td>
</tr>
<tr>
<td>OrderedBag(Of T) OrderedBag(Of T) New(IEnumerable(Of T), Comparison(Of T))</td>
<td>Creates a new OrderedBag. The passed delegate will be used to compare items in this bag. The bag is initialized with all the items in the given collection.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag<Of<T>> Class
OrderedBag<Of<T>> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>> Constructor

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedBag. The T must implement IComparable<T> or IComparable. The CompareTo method of this interface will be used to compare items in this bag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
C#
public OrderedBag()

Visual Basic (Declaration)
Public Sub New

Visual C++
public:
OrderedBag ()
Remarks

Items that are null are permitted, and will be sorted before all other items.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::InvalidOperationException</td>
<td>T does not implement IComparable&lt;TKey&gt;.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>> Constructor (IComparer<Of <T>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedBag. The Compare method of the passed comparison object will be used to compare items in this bag.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public OrderedBag(
    IComparer<T> comparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub New (_
    comparer As IComparer(Of T) _
)
```

**Visual C++**

```csharp
public:
    OrderedBag (  
        IComparer<T>^ comparer
    )
```

### Parameters

**comparer**

`IComparer<Of <T>>`

An instance of IComparer<T> that will be used to compare items.
Remarks

The GetHashCode and Equals methods of the provided IComparer<T> will never be called, and need not be implemented.
See Also

OrderedBag<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>> Constructor (IEnumerable<Of <T>>)  

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedBag. The T must implement IComparable<T> or IComparable. The CompareTo method of this interface will be used to compare items in this bag. The bag is initialized with all the items in the given collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedBag(
    IEnumerable<T> collection
)

Visual Basic (Declaration)

Public Sub New ( _
    collection As IEnumerable(Of T) _
)

Visual C++

public:
    OrderedBag ( _
        IEnumerable<T>^ collection
    )

Parameters

collection
    IEnumerable<(Of <T>)>
    A collection with items to be placed into the OrderedBag.
Remarks

Items that are null are permitted, and will be sorted before all other items.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>T does not implement IComparable&lt;TKey&gt;.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of <T> )> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<
(Of
<T>
)>
Constructor
(Comparison<
(Of
<T>
)>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedBag. The passed delegate will be used to compare items in this bag.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
C#

```csharp
public OrderedBag(
    Comparison<T> comparison
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub New (_
    comparison As Comparison(Of T) _
)
```

**Visual C++**

```cpp
public:
OrderedBag ( 
    Comparison<T>^ comparison
)
```

**Parameters**

comparison

```csharp
Comparison<Of <T>>
```

A delegate to a method that will be used to compare items.
See Also

OrderedBag(Of T)> Class
Winellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<(Of <T>)> Constructor (IEnumerable<(Of <T>)>, IComparer<(Of <T>)>)

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedBag. The Compare method of the passed comparison object will be used to compare items in this bag. The bag is initialized with all the items in the given collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedBag(
    IEnumerable<T> collection,
    IComparer<T> comparer
)

Visual Basic (Declaration)

Public Sub New ( _
    collection As IEnumerable(Of T), _
    comparer As IComparer(Of T) _
)

Visual C++

public:
    OrderedBag (    
    IEnumerable<T>^ collection,
    IComparer<T>^ comparer

Parameters

collection
    IEnumerable<Of <T>>
    A collection with items to be placed into the OrderedBag.

comparer
    IComparer<Of <T>>
    An instance of IComparer<T> that will be used to compare items.
Remarks

The GetHashCode and Equals methods of the provided IComparer<T> will never be called, and need not be implemented.
See Also

OrderedBag(Of <T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<(Of <T>)> Constructor (IEnumerable<(Of <T>)>), Comparison<(Of <T>)>)

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedBag. The passed delegate will be used to compare items in this bag. The bag is initialized with all the items in the given collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedBag(
    IEnumerable<T> collection,
    Comparison<T> comparison
)

Visual Basic (Declaration)

Public Sub New (_
    collection As IEnumerable(Of T), _
    comparison As Comparison(Of T) _
)

Visual C++

public:
OrderedBag (  
    IEnumerable<T>^ collection,
    Comparison<T>^ comparison
)

Parameters

collection
    IEnumerable<Of <T>>
    A collection with items to be placed into the OrderedBag.

comparison
    Comparison<Of <T>>
    A delegate to a method that will be used to compare items.
See Also

**OrderedBag(Of T)>** Class
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
OrderedBag<(Of <T>)> Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class OrderedBag<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class OrderedBag(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class OrderedBag
Type Parameters

The type exposes the following methods.
### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Must be overridden to allow adding items to this collection.</td>
</tr>
<tr>
<td><strong>AddMany</strong></td>
<td>Adds all the items in collection to the bag.</td>
</tr>
<tr>
<td><strong>AsList</strong></td>
<td>Get a read-only list view of the items in this ordered bag. The items in the list are in sorted order, with the smallest item at index 0. This view does not copy any data, and reflects any changes to the underlying OrderedBag. Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Removes all items from the bag. (Overrides CollectionBase&lt;Of &lt;T&gt;&gt;...::Clear()) Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment. Makes a deep clone of this bag. A new bag is created with a clone of each element of this bag, by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment. Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>(Overrides CollectionBase&lt;Of &lt;T&gt;&gt;...::Clear()) Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment. Makes a deep clone of this bag. A new bag is created with a clone of each element of this bag, by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment. Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using</td>
</tr>
<tr>
<td><strong>Clone</strong></td>
<td>Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment. Makes a deep clone of this bag. A new bag is created with a clone of each element of this bag, by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment. Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using</td>
</tr>
<tr>
<td><strong>CloneContents</strong></td>
<td>Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using</td>
</tr>
</tbody>
</table>
IComparable<T>.Equals or Object.Equals.

Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.

(Inherited from CollectionBase<Of <T> >.)

Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.

(Inherited from CollectionBase<Of <T> >.)

Counts the number of items in the collection that satisfy the condition defined by predicate.

(Inherited from CollectionBase<Of <T> >.)

Computes the difference of this bag with another bag. The difference of these two bags is all items that appear in this bag, but not in otherBag. If an item appears X times in this bag, and Y times in the other bag, the difference contains the item X - Y times (zero times if Y >= X). A new bag is created with the difference of the bags and is returned. This bag and the other bag are unchanged.

Computes the difference of this bag with another bag. The difference of these two bags is all items that appear in this bag, but not in otherBag. If an item appears X times in this bag, and Y times in the other bag, the difference contains the item X - Y times (zero times if Y >= X). This bag receives the difference of the two bags; the other bag is unchanged.

Enumerates all the items in the bag, but enumerates equal items just once, even if they occur multiple times in the bag.

Determines whether the specified Object is equal to the current Object.
Determines if the collection contains any item that satisfies the condition defined by predicate.
(Inherited from CollectionBase<Of <T>>.)

Enumerates the items in the collection that satisfy the condition defined by predicate.
(Inherited from CollectionBase<Of <T>>.)

Performs the specified action on each item in this collection.
(Inherited from CollectionBase<Of <T>>.)

Returns an enumerator that enumerates all the items in the bag. The items are enumerated in sorted order.
(Overrides CollectionBase<Of <T>>::GetEnumerator().)

Enumerates all of the items in this bag that are equal to item, according to the comparison mechanism that was used when the bag was created. The bag is not changed.

If the bag does contain an item equal to item, then the enumeration contains no items.

Returns the first item in the bag: the item that would appear first if the bag was enumerated. This is also the smallest item in the bag.

Serves as a hash function for a particular type.
GetHashCode() is suitable for use in hashing algorithms and data structures like a hash table.
(Inherited from Object.)

Returns the last item in the bag: the item that would appear last if the bag was enumerated. This is also the largest item in the bag.

Gets the Type of the current instance.
(Inherited from Object.)

Get the index of the given item in the sorted order.
IndexOf

The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1. If multiple equal items exist, the smallest index of the equal items is returned.

Intersection

Computes the intersection of this bag with another bag. The intersection of two bags is all items that appear in both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item Minimum(X,Y) times. A new bag is created with the intersection of the bags and is returned. This bag and the other bag are unchanged.

IntersectionWith

Computes the intersection of this bag with another bag. The intersection of two bags is all items that appear in both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item Minimum(X,Y) times. This bag receives the intersection of the two bags, the other bag is unchanged.

IsDisjointFrom

Determines if this bag is disjoint from another bag. Two bags are disjoint if no item from one set is equal to any item in the other bag.

IsEqualTo

Determines if this bag is equal to another bag. This bag is equal to otherBag if they contain the same items, each the same number of times.

IsProperSubsetOf

Determines if this bag is a proper subset of another bag. Neither bag is modified. This bag is a subset of otherBag if every element in this bag is also in otherBag, at least the same number of times. Additional, this bag must have strictly fewer items than otherBag.

IsProperSupersetOf

Determines if this bag is a proper superset of another bag. Neither bag is modified. This bag is a proper superset of otherBag if every element in otherBag is also in this bag, at least the same number of times. Additional, this bag must have strictly more items than otherBag.
**IsSubsetOf**
Determines if this bag is a subset of another bag. Neither bag is modified. This bag is a subset of otherBag if every element in this bag is also in otherBag, at least the same number of times.

**IsSupersetOf**
Determines if this bag is a superset of another bag. Neither bag is modified. This bag is a superset of otherBag if every element in otherBag is also in this bag, at least the same number of times.

**_LastIndexOf**
Get the index of the given item in the sorted order. The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1. If multiple equal items exist, the largest index of the equal items is returned.

**NumberOfCopies**
Returns the number of copies of item in the bag. More precisely, returns the number of items in the bag that compare equal to item.

**Range**
Returns a View collection that can be used for enumerating a range of the items in the bag. Only items that are greater than from and less than to are included. The items are enumerated in sorted order. Items equal to the end points of the range can be included or excluded depending on the fromInclusive and toInclusive parameters.

**RangeFrom**
Returns a View collection that can be used for enumerating a range of the items in the bag. Only items that are greater than (and optionally, equal to) from are included. The items are enumerated in sorted order. Items equal to from can be included or excluded depending on the fromInclusive parameter.

**RangeTo**
Returns a View collection that can be used for enumerating a range of the items in the bag. Only items that are less than (and optionally, equal to) to are included. The items are enumerated in sorted order. Items equal to to can be included or excluded depending on the toInclusive parameter.

**Remove**
Must be overridden to allow removing items from
this collection.

- **RemoveAll**
  Removes all the items in the collection that satisfy the condition defined by predicate. (Inherited from `CollectionBase<Of <T>>`.)

- **RemoveAllCopies**
  Searches the bag for all items equal to item, and removes all of them from the bag. If not found, the bag is unchanged.

- **RemoveFirst**
  Removes the first item in the bag. This is also the smallest item in the bag.

- **RemoveLast**
  Removes the last item in the bag. This is also the largest item in the bag.

- **RemoveMany**
  Removes all the items in collection from the bag. Items not present in the bag are ignored.

- **Reversed**
  Returns a View collection that can be used for enumerating the items in the bag in reversed order.

- **Sum**
  Computes the sum of this bag with another bag. The sum of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item (X+Y) times. A new bag is created with the sum of the bags and is returned. This bag and the other bag are unchanged.

- **SumWith**
  Computes the sum of this bag with another bag. The sum of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item (X+Y) times. This bag receives the sum of the two bags, the other bag is unchanged.

- **SymmetricDifference**
  Computes the symmetric difference of this bag with another bag. The symmetric difference of two bags is all items that appear in either of the bags, but not both. If an item appears X times in one bag, and Y times in the other bag, the symmetric difference contains the item `AbsoluteValue(X - Y)` times. A new bag is created with the symmetric difference of the bags and is returned. This bag and the other bag are
SymmetricDifferenceWith

Computes the symmetric difference of this bag with another bag. The symmetric difference of two bags is all items that appear in either of the bags, but not both. If an item appears X times in one bag, and Y times in the other bag, the symmetric difference contains the item AbsoluteValue(X - Y times). This bag receives the symmetric difference of the two bags; the other bag is unchanged.

ToArray

Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.
(Inherited from CollectionBase(Of <T>).)

ToString

Shows the string representation of the collection. The string representation contains a list of the items in the collection. Contained collections (except string) are expanded recursively.
(Inherited from CollectionBase(Of <T>).)

TrueForAll

Determines if all of the items in the collection satisfy the condition defined by predicate.
(Inherited from CollectionBase(Of <T>).)

Union

Computes the union of this bag with another bag. The union of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the union contains the item Maximum(X,Y) times. A new bag is created with the union of the bags and is returned. This bag and the other bag are unchanged.

UnionWith

Computes the union of this bag with another bag. The union of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the union contains the item Maximum(X,Y) times. This bag receives the union of the two bags, the other bag is unchanged.
# Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection::CopyTo</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>IEnumerable::GetEnumerator</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>ICloneable::Clone</td>
<td>Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
</tbody>
</table>
See Also

**OrderedBag<Of <T>>**  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>...::AddMany Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds all the items in collection to the bag.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public void AddMany(
    IEnumerable<T> collection
)
```

### Visual Basic (Declaration)

```vbnet
Public Sub AddMany ( _
    collection As IEnumerable(Of T) _
)
```

### Visual C++

```cpp
public:
void AddMany ( 
    IEnumerable<T>^ collection
)
```

## Parameters

**collection**

*IEnumerable<(Of <T>)*>

A collection of items to add to the bag.
Adding the collection takes time $O(M \log N)$, where $N$ is the number of items in the bag, and $M$ is the number of items in collection.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentNullException</td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of<T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>...:AsList Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Get a read-only list view of the items in this ordered bag. The items in the list are in sorted order, with the smallest item at index 0. This view does not copy any data, and reflects any changes to the underlying OrderedBag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

public `IList<T>` AsList()

**Visual Basic (Declaration)**

Public Function AsList As `IList(Of T)`

**Visual C++**

public: `IList<T>^` AsList ()

**Return Value**

A read-only `IList<T>` view onto this `OrderedBag`. 
See Also

**OrderedBag(Of T)> Class**

**Wintellect.PowerCollections** Namespace

Send [comments](mailto:) about this topic to Microsoft.
OrderedBag<(Of <T> )...::Clear Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all items from the bag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

public override sealed void Clear()

**Visual Basic (Declaration)**

Public Overrides NotOverridable Sub Clear

**Visual C++**

public:
virtual void Clear () override sealed
Remarks

Clearing the bag takes a constant amount of time, regardless of the number of items in it.
See Also

**OrderedBag(Of T)>** Class

**Wintellect.PowerCollections** Namespace

Send [comments](mailto:send-comments@microsoft.com) about this topic to Microsoft.
OrderedBag<Of <T>>::Clone Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public  OrderedBag<T>  Clone()

Visual Basic (Declaration)

Public Function Clone As  OrderedBag(Of T)

Visual C++

public:  
OrderedBag<T>^  Clone()  

Return Value

The cloned bag.
Remarks

Cloning the bag takes time $O(N)$, where $N$ is the number of items in the bag.
See Also

OrdenedBag<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag(Of T) :: CloneContents Method

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a deep clone of this bag. A new bag is created with a clone of each element of this bag, by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
# Syntax

**C#**

```csharp
public OrderedBag<T> CloneContents()
```

**Visual Basic (Declaration)**

```vbnet
Public Function CloneContents As OrderedBag(Of T)
```

**Visual C++**

```cpp
public:
OrderedBag<T> ^ CloneContents ()
```

## Return Value

The cloned bag.
Remarks

If T is a reference type, it must implement ICloneable. Otherwise, an InvalidOperationException is thrown.

Cloning the bag takes time O(N log N), where N is the number of items in the bag.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>T is a reference type that does not implement ICloneable.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag<Of<T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>::.Difference Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the difference of this bag with another bag. The difference of these two bags is all items that appear in this bag, but not in otherBag. If an item appears X times in this bag, and Y times in the other bag, the difference contains the item X - Y times (zero times if Y >= X). A new bag is created with the difference of the bags and is returned. This bag and the other bag are unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedBag<T> Difference(
    OrderedBag<T> otherBag
)

Visual Basic (Declaration)

Public Function Difference ( _
    otherBag As OrderedBag(Of T) _
) As OrderedBag(Of T)

Visual C++

public:
    OrderedBag<T>^ Difference ( _
    OrderedBag<T>^ otherBag
)

Parameters

otherBag
    OrderedBag<(Of <T>)> Bag to difference with.

Return Value

The difference of the two bags.
Remarks

The difference of two bags is computed in time $O(M + N \log M)$, where $M$ is the size of the larger bag, and $N$ is the size of the smaller bag.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::InvalidOperationException</td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
<tr>
<td>System::::ArgumentNullException</td>
<td>otherBag is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T> cheres::DifferenceWith Method

See Also

Visual Basic (Declaration) ☑ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the difference of this bag with another bag. The difference of these two bags is all items that appear in this bag, but not in otherBag. If an item appears X times in this bag, and Y times in the other bag, the difference contains the item X - Y times (zero times if Y >= X). This bag receives the difference of the two bags; the other bag is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public void DifferenceWith(
    OrderedBag<T> otherBag
)
```

Visual Basic (Declaration)

```vbnet
Public Sub DifferenceWith ( _
    otherBag As OrderedBag(Of T) _
)
```

Visual C++

```cpp
public:
    void DifferenceWith ( 
        OrderedBag<T>^ otherBag
    )
```

Parameters

otherBag

`OrderedBag<(Of <T>)>`
Bag to difference with.
Remarks

The difference of two bags is computed in time $O(M + N \log M)$, where $M$ is the size of the larger bag, and $N$ is the size of the smaller bag.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::InvalidOperationException</td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
<tr>
<td>System::::ArgumentNullException</td>
<td>otherBag is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>..::DistinctItems Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerates all the items in the bag, but enumerates equal items just once, even if they occur multiple times in the bag.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public IEnumerable<T> DistinctItems()

Visual Basic (Declaration)

Public Function DistinctItems As IEnumerable(Of T)

Visual C++

public: IEnumerable<T>^ DistinctItems ()

Return Value

An IEnumerable<T> that enumerates the unique items.
Remarks

If the bag is changed while items are being enumerated, the enumeration will terminate with an InvalidOperationException.
See Also

OrderedBag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>><>::GetEnumerator Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns an enumerator that enumerates all the items in the bag. The items are enumerated in sorted order.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public override sealed IEnumerator<T> GetEnumerator()
```

Visual Basic (Declaration)

```vbnet
Public Overrides NotOverridable Function GetEnumerator As IEnumerator
```

Visual C++

```cpp
public:
virtual IEnumerator<T>^ GetEnumerator() override sealed
```

Return Value

An enumerator for enumerating all the items in the OrderedBag.
Remarks

Typically, this method is not called directly. Instead the "foreach" statement is used to enumerate the items, which uses this method implicitly.

If an item is added to or deleted from the bag while it is being enumerated, then the enumeration will end with an InvalidOperationException.

Enumeration all the items in the bag takes time O(N), where N is the number of items in the bag.
See Also

OrderedBag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Enumerates all of the items in this bag that are equal to item, according to the comparison mechanism that was used when the bag was created. The bag is not changed.

If the bag does contain an item equal to item, then the enumeration contains no items.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public IEnumerable<T> GetEqualItems(T item)
```

#### Visual Basic (Declaration)

```vbnet
Public Function GetEqualItems(_
    item As T _
) As IEnumerable(Of T)
```

#### Visual C++

```cpp
public:
    IEnumerable<T>^ GetEqualItems(
        T item
    )
```

### Parameters

- **item**
  - **Type:** T
  - **Description:** The item to search for.

### Return Value

An IEnumerable<T> that enumerates all the items in the bag equal to item.
Remarks

Enumeration the items in the bag equal to item takes time $O(\log N + M)$, where $N$ is the total number of items in the bag, and $M$ is the number of items equal to item.
See Also

OrderedBag<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>...::GetFirst Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the first item in the bag: the item that would appear first if the bag was enumerated. This is also the smallest item in the bag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public T GetFirst()
```

**Visual Basic (Declaration)**

```vbnet
Public Function GetFirst As T
```

**Visual C++**

```cpp
public:
T GetFirst()
```

**Return Value**

The first item in the bag. If more than one item is smallest, the first one added is returned.
Remarks

GetFirst() takes time $O(\log N)$, where $N$ is the number of items in the bag.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::InvalidOperationException</td>
<td>The bag is empty.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<(Of <T>)>::GetLast Method

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the last item in the bag: the item that would appear last if the bag was enumerated. This is also the largest item in the bag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public T GetLast()
```

**Visual Basic (Declaration)**

```vbnet
Public Function GetLast As T
```

**Visual C++**

```cpp
public:
  T GetLast();
```

**Return Value**

The last item in the bag. If more than one item is largest, the last one added is returned.
Remarks

GetLast() takes time $O(\log N)$, where $N$ is the number of items in the bag.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::InvalidOperationException</code></td>
<td>The bag is empty.</td>
</tr>
</tbody>
</table>
See Also

**OrderedBag(Of T) Class**

**Wintellect.PowerCollections Namespace**

Send [comments](#) about this topic to Microsoft.
OrderedBag<Of <T>>::IndexOf Method

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Get the index of the given item in the sorted order. The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1. If multiple equal items exist, the smallest index of the equal items is returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public int IndexOf(
    T item
)
```

Visual Basic (Declaration)

```vbnet
Public Function IndexOf ( _
    item As T _
) As Integer
```

Visual C++

```cpp
public:
    int IndexOf ( 
        T item
    )
```

Parameters

- **item**
  - **T**
  - The item to get the index of.

Return Value

The index of the first item in the sorted bag equal to item, or -1 if the item is not present in the set.
Remarks

Finding the index takes time $O(\log N)$, which $N$ is the number of items in the set.
See Also

**OrderedBag(Of T)** Class  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>><::Intersection Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the intersection of this bag with another bag. The intersection of two bags is all items that appear in both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item Minimum(X,Y) times. A new bag is created with the intersection of the bags and is returned. This bag and the other bag are unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public OrderedBag<T> Intersection(
    OrderedBag<T> otherBag
)
```

Visual Basic (Declaration)

```vbnet
Public Function Intersection (_
    otherBag As OrderedBag(Of T) _
) As OrderedBag(Of T)
```

Visual C++

```cpp
public:
OrderedBag<T>^ Intersection (
    OrderedBag<T>^ otherBag
)
```

Parameters

otherBag

`OrderedBag<(Of <T>))>
Bag to intersection with.

Return Value

The intersection of the two bags.
Remarks

When equal items appear in both bags, the intersection will include an arbitrary choice of one of the two equal items.

The intersection of two bags is computed in time $O(N \log M)$, where $M$ is the size of the larger bag, and $N$ is the size of the smaller bag.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::InvalidOperationException</td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
<tr>
<td>System::ArgumentNullException</td>
<td>otherBag is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>.::IntersectionWith Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the intersection of this bag with another bag. The intersection of two bags is all items that appear in both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item Minimum(X,Y) times. This bag receives the intersection of the two bags, the other bag is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public void IntersectionWith(
    OrderedBag<T> otherBag
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub IntersectionWith ( _
    otherBag As OrderedBag(Of T) _
)
```

**Visual C++**

```cpp
public:
void IntersectionWith ( 
    OrderedBag<T>^ otherBag
)
```

**Parameters**

otherBag

`OrderedBag<(Of <T>)>`

Bag to intersection with.
Remarks

When equal items appear in both bags, the intersection will include an arbitrary choice of one of the two equal items.

The intersection of two bags is computed in time $O(N \log M)$, where $M$ is the size of the larger bag, and $N$ is the size of the smaller bag.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>otherBag is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this bag is disjoint from another bag. Two bags are disjoint if no item from one set is equal to any item in the other bag.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public bool IsDisjointFrom(
    OrderedBag<T> otherBag
)
```

Visual Basic (Declaration)

```vbnet
Public Function IsDisjointFrom ( _
    otherBag As OrderedBag(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
    bool IsDisjointFrom ( 
        OrderedBag<T>^ otherBag
    )
```

Parameters

otherBag
    ```csharp
    OrderedBag<T>
    ```
    Bag to check disjointness with.

Return Value

True if the two bags are disjoint, false otherwise.
Remarks

The answer is computed in time $O(N)$, where $N$ is the size of the smaller set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::InvalidOperationException</td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of <T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>.::.IsEqualTo Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this bag is equal to another bag. This bag is equal to otherBag if they contain the same items, each the same number of times.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public bool IsEqualTo(
    OrderedBag<T> otherBag
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function IsEqualTo ( _
    otherBag As OrderedBag(Of T) _
) As Boolean
```

**Visual C++**

```c++
public:
    bool IsEqualTo ( 
    OrderedBag<T>^ otherBag
)
```

**Parameters**

otherBag
    OrderedBag(Of Of <T>)
    OrderedBag to compare to

**Return Value**

True if this bag is equal to otherBag, false otherwise.
**Remarks**

IsEqualTo is computed in time $O(N)$, where $N$ is the number of items in this bag.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of <T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<(Of <T>)>..::IsProperSubsetOf Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this bag is a proper subset of another bag. Neither bag is modified. This bag is a subset of otherBag if every element in this bag is also in otherBag, at least the same number of times. Additional, this bag must have strictly fewer items than otherBag.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public bool IsProperSubsetOf(
    OrderedBag<T> otherBag
)
```

Visual Basic (Declaration)

```vbnet
Public Function IsProperSubsetOf ( _
    otherBag As OrderedBag(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
    bool IsProperSubsetOf ( 
        OrderedBag<T>^ otherBag
    )
```

Parameters

otherBag

`OrderedBag<Of <T>>`

OrderedBag to compare to.

Return Value

True if this is a proper subset of otherBag.
Remarks

IsSubsetOf is computed in time $O(N \log M)$, where $M$ is the size of the, and $N$ is the size of the this bag.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::InvalidOperationException</td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
<tr>
<td>System::::ArgumentNullException</td>
<td>otherBag is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of <T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<(Of <T>)>...::IsProperSupersetOf Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this bag is a proper superset of another bag. Neither bag is modified. This bag is a proper superset of otherBag if every element in otherBag is also in this bag, at least the same number of times. Additional, this bag must have strictly more items than otherBag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public bool IsProperSupersetOf(
    OrderedBag<T> otherBag
)
```

Visual Basic (Declaration)

```vbnet
Public Function IsProperSupersetOf ( _
    otherBag As OrderedBag(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
bool IsProperSupersetOf ( 
    OrderedBag<T>^ otherBag
)
```

Parameters

otherBag

OrderedBag<Of <T>>

OrderedBag to compare to.

Return Value

True if this is a proper superset of otherBag.
Remarks

IsProperSupersetOf is computed in time $O(M \log N)$, where $M$ is the number of unique items in otherBag.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..:::InvalidOperationException</td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
<tr>
<td>System..:::ArgumentNullException</td>
<td>otherBag is null.</td>
</tr>
</tbody>
</table>
See Also

**OrderedBag(Of T) Class**

**Wintellect.PowerCollections Namespace**

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
OrderedBag<Of <T>>...:..IsSubsetOf Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this bag is a subset of another bag. Neither bag is modified. This bag is a subset of otherBag if every element in this bag is also in otherBag, at least the same number of times.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public bool IsSubsetOf(
    OrderedBag<T> otherBag
)
```

Visual Basic (Declaration)

```vbnet
Public Function IsSubsetOf ( _
    otherBag As OrderedBag(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
    bool IsSubsetOf ( _
    OrderedBag<T>^ otherBag
    )
```

Parameters

otherBag

`OrderedBag<Of <T>>`

OrderedBag to compare to.

Return Value

True if this is a subset of otherBag.
Remarks

IsSubsetOf is computed in time $O(N \log M)$, where $M$ is the size of the otherBag, and $N$ is the size of the this bag.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..:::InvalidOperationException</td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
<tr>
<td>System..:::ArgumentNullException</td>
<td>otherBag is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>...:::IsSupersetOf Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this bag is a superset of another bag. Neither bag is modified. This bag is a superset of otherBag if every element in otherBag is also in this bag, at least the same number of times.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

    public bool IsSupersetOf(
        OrderedBag<T> otherBag
    )

Visual Basic (Declaration)

Public Function IsSupersetOf (_
    otherBag As OrderedBag(Of T)_
) As Boolean

Visual C++

public:
    bool IsSupersetOf (  
        OrderedBag<T>^ otherBag
    )

Parameters

otherBag
    OrderedBag<(Of <T>)>
    OrderedBag to compare to.

Return Value

True if this is a superset of otherBag.
Remarks

IsSupersetOf is computed in time $O(M \log N)$, where $M$ is the size of the otherBag, and $N$ is the size of the this set.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::InvalidOperationException</td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
<tr>
<td>System..::ArgumentNullException</td>
<td>otherBag is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of 'T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag(Of $T$)\ldots::\texttt{LastIndexOf} Method

\textbf{See Also}

\begin{itemize}
\item Visual Basic (Declaration)
\item Visual Basic (Usage)
\item C#
\item Visual C++
\item J#
\item JScript
\item XAML
\end{itemize}

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Get the index of the given item in the sorted order. The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1. If multiple equal items exist, the largest index of the equal items is returned.

\textbf{Namespace:} Wintellect.PowerCollections
\textbf{Assembly:} PowerCollections (in PowerCollections.dll)
Syntax

C#

public int LastIndexOf(
    T item
)

Visual Basic (Declaration)

Public Function LastIndexOf ( _
    item As T _
) As Integer

Visual C++

public:
    int LastIndexOf ( 
        T item
    )

Parameters

item
    T
    The item to get the index of.

Return Value

The index of the last item in the sorted bag equal to item, or -1 if the item is not present in the set.
Remarks

Finding the index takes time $O(\log N)$, which $N$ is the number of items in the set.
See Also

OrderedBag<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>]._::NumberOfCopies Method

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the number of copies of item in the bag. More precisely, returns the number of items in the bag that compare equal to item.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public int NumberOfCopies(
    T item
)

Visual Basic (Declaration)

Public Function NumberOfCopies ( _
    item As T _
) As Integer

Visual C++

public:
    int NumberOfCopies ( 
    T item
    )

Parameters

item
    T
    The item to search for in the bag.

Return Value

The number of items in the bag that compare equal to item.
Remarks

NumberOfCopies() takes time $O(\log N + M)$, where $N$ is the total number of items in the bag, and $M$ is the number of copies of item in the bag.
See Also

OrderedBag(Of<T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<(Of <T>)}::.Range Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a View collection that can be used for enumerating a range of the items in the bag. Only items that are greater than from and less than to are included. The items are enumerated in sorted order. Items equal to the end points of the range can be included or excluded depending on the fromInclusive and toInclusive parameters.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public OrderedBag<T>.Range(T from,
    bool fromInclusive,
    T to,
    bool toInclusive
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function Range (_
    from As T, _
    fromInclusive As Boolean, _
    to As T, _
    toInclusive As Boolean _
) As OrderedBag(Of T)>..::View
```

**Visual C++**

```cpp
public:
OrderedBag<(Of <T>),..::View> Range (/
    T from,
    bool fromInclusive,
    T to,
    bool toInclusive
)
```

**Parameters**

**from**

- **T**
  - The lower bound of the range.

**fromInclusive**

- **Boolean**
  - If true, the lower bound is inclusive--items equal to the lower bound will be included in the range. If false, the lower bound is exclusive--items equal to
the lower bound will not be included in the range.

to

T
The upper bound of the range.

toInclusive

Boolean
If true, the upper bound is inclusive--items equal to the upper bound will be included in the range. If false, the upper bound is exclusive--items equal to the upper bound will not be included in the range.

Return Value

An OrderedBag.View of items in the given range.
Remarks

If from is greater than or equal to to, the returned collection is empty.

Typically, this method is used in conjunction with a foreach statement. For example:

```csharp
foreach(T item in bag.Range(from, true, to, false)) {
    // process item
}
```

If an item is added to or deleted from the bag while the View is being enumerated, then the enumeration will end with an InvalidOperationException.

Calling Range does not copy the data in the tree, and the operation takes constant time.
See Also

OrderedBag<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag(Of T)>..::RangeFrom Method

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a View collection that can be used for enumerating a range of the items in the bag. Only items that are greater than (and optionally, equal to) from are included. The items are enumerated in sorted order. Items equal to from can be included or excluded depending on the fromInclusive parameter.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedBag<T>.OrderedBag<Of <T>>..::View RangeFrom(
    T from,
    bool fromInclusive
)

Visual Basic (Declaration)

Public Function RangeFrom ( _
    from As T, _
    fromInclusive As Boolean _
) As OrderedBag<Of <T>>..::View

Visual C++

public:
    OrderedBag<Of <T>>..::View^ RangeFrom ( 
    T from,
    bool fromInclusive
)

Parameters

from
    T
    The lower bound of the range.

fromInclusive
    Boolean
    If true, the lower bound is inclusive--items equal to the lower bound will be included in the range. If false, the lower bound is exclusive--items equal to the lower bound will not be included in the range.

Return Value
An OrderedBag.View of items in the given range.
Remarks

Typically, this method is used in conjunction with a foreach statement. For example:

```csharp
foreach(T item in bag.RangeFrom(from, true)) {
    // process item
}
```

If an item is added to or deleted from the bag while the View is being enumerated, then the enumeration will end with an InvalidOperationException.

Calling RangeFrom does not copy the data in the tree, and the operation takes constant time.
See Also

OrderedBag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T> >...::RangeTo Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a View collection that can be used for enumerating a range of the items in the bag. Only items that are less than (and optionally, equal to) to are included. The items are enumerated in sorted order. Items equal to to can be included or excluded depending on the toInclusive parameter.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public OrderedBag<T>.OrderedBag<Of<T>>..:::View RangeTo(T to,
    bool toInclusive
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function RangeTo (_
    to As T, _
    toInclusive As Boolean _
) As OrderedBag<Of<T>>..:::View
```

**Visual C++**

```cpp
public:
OrderedBag<Of<T>>..:::View^ RangeTo (_
    T to,
    bool toInclusive
)
```

**Parameters**

- **to**
  - `T`
    - The upper bound of the range.

- **toInclusive**
  - `Boolean`
    - If true, the upper bound is inclusive--items equal to the upper bound will be included in the range. If false, the upper bound is exclusive--items equal to the upper bound will not be included in the range.

**Return Value**
An OrderedBag.View of items in the given range.
**Remarks**

Typically, this method is used in conjunction with a foreach statement. For example:

```csharp
foreach(T item in bag.RangeTo(to, false)) {
    // process item
}
```

If an item is added to or deleted from the bag while the View is being enumerated, then the enumeration will end with an InvalidOperationException.

Calling RangeTo does not copy the data in the tree, and the operation takes constant time.
See Also

OrderedBag<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<\(\text{T}\)>::RemoveAllCopies Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Searches the bag for all items equal to item, and removes all of them from the bag. If not found, the bag is unchanged.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public int RemoveAllCopies(
    T item
)

Visual Basic (Declaration)

Public Function RemoveAllCopies (_
    item As T _
) As Integer

Visual C++

public:
    int RemoveAllCopies ( _
        T item
    )

Parameters

item
    T
    The item to remove.

Return Value

The number of copies of item that were found and removed.
Remarks

Equality between items is determined by the comparison instance or delegate used to create the bag.

RemoveAllCopies() takes time \( O(M \log N) \), where \( N \) is the total number of items in the bag, and \( M \) is the number of items equal to item.
See Also

OrderedBag<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>...::RemoveFirst Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes the first item in the bag. This is also the smallest item in the bag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

public T RemoveFirst()

**Visual Basic (Declaration)**

Public Function RemoveFirst As T

**Visual C++**

public:
T RemoveFirst ()

**Return Value**

The item that was removed, which was the smallest item in the bag.
Remarks

RemoveFirst() takes time $O(\log N)$, where $N$ is the number of items in the bag.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>The bag is empty.</td>
</tr>
</tbody>
</table>
See Also

**OrderedBag**(Of `<T>`) Class
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
OrderedBag<(Of <T>))::RemoveLast Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes the last item in the bag. This is also the largest item in the bag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public T RemoveLast()

Visual Basic (Declaration)

Public Function RemoveLast As T

Visual C++

public:
T RemoveLast ()

Return Value

The item that was removed, which was the largest item in the bag.
Remarks

RemoveLast() takes time $O(\log N)$, where $N$ is the number of items in the bag.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>The bag is empty.</td>
</tr>
</tbody>
</table>

- **Exceptions**

Exception | Condition
--- | ---
`System::InvalidOperationException` | The bag is empty.
See Also

OrderedBag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<(Of <T>)>.::RemoveMany Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all the items in collection from the bag. Items not present in the bag are ignored.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public int RemoveMany(
    IEnumerable<T> collection
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function RemoveMany ( _
    collection As IEnumerable(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:
    int RemoveMany ( _
        IEnumerable<T>^ collection
    )
```

### Parameters

collection

```csharp
IEnumerable<T>
```

A collection of items to remove from the bag.

### Return Value

The number of items removed from the bag.
Remarks

Equality between items is determined by the comparison instance or delegate used to create the bag.

Removing the collection takes time $O(M \log N)$, where $N$ is the number of items in the bag, and $M$ is the number of items in collection.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::ArgumentNullException</code></td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<(Of <T>)>::Reversed Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a View collection that can be used for enumerating the items in the bag in reversed order.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedBag<T>.OrderedBag<Of <T>>.:::View Reversed()

Visual Basic (Declaration)

Public Function Reversed As OrderedBag(Of <T>).:::View

Visual C++

public: OrderedBag<Of <T>>.:::View^ Reversed ()

Return Value

An OrderedBag.View of items in reverse order.
Remarks

Typically, this method is used in conjunction with a foreach statement. For example:

```csharp
foreach(T item in bag.Reversed()) {
    // process item
}
```

If an item is added to or deleted from the bag while the View is being enumerated, then the enumeration will end with an InvalidOperationException.

Calling Reverse does not copy the data in the tree, and the operation takes constant time.
See Also

OrderedBag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>.::Sum Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the sum of this bag with another bag. The sum of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item (X+Y) times. A new bag is created with the sum of the bags and is returned. This bag and the other bag are unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedBag<T> Sum(
    OrderedBag<T> otherBag
)

Visual Basic (Declaration)

Public Function Sum ( _
    otherBag As OrderedBag(Of T) _
) As OrderedBag(Of T)

Visual C++

public:
    OrderedBag<T>^ Sum ( 
        OrderedBag<T>^ otherBag
    )

Parameters

otherBag
    OrderedBag<(Of <T>)>
    Bag to sum with.

Return Value

The sum of the two bags.
Remarks

The sum of two bags is computed in time $O(M + N \log M)$, where $M$ is the size of the larger bag, and $N$ is the size of the smaller bag.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>otherBag is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag<Of<T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>><::SumWith Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the sum of this bag with another bag. The sum of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the sum contains the item (X+Y) times. This bag receives the sum of the two bags, the other bag is unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public void SumWith(
    OrderedBag<T> otherBag
)

Visual Basic (Declaration)

Public Sub SumWith ( _
    otherBag As OrderedBag(Of T) _
)

Visual C++

public:
void SumWith ( 
    OrderedBag<T>^ otherBag
)

Parameters

otherBag
    OrderedBag<(Of <T>))>
    Bag to sum with.
Remarks

The sum of two bags is computed in time $O(M + N \log M)$, where $M$ is the size of the larger bag, and $N$ is the size of the smaller bag.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>otherBag is null.</td>
</tr>
</tbody>
</table>
See Also

**OrderedBag(Of T) Class**
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
OrderedBag<(Of <T>)>.:::SymmetricDifference Method

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the symmetric difference of this bag with another bag. The symmetric difference of two bags is all items that appear in either of the bags, but not both. If an item appears X times in one bag, and Y times in the other bag, the symmetric difference contains the item AbsoluteValue(X - Y times). A new bag is created with the symmetric difference of the bags and is returned. This bag and the other bag are unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public  OrderedBag<T> SymmetricDifference( 
OrderedBag<T> otherBag 
)

Visual Basic (Declaration)

Public Function SymmetricDifference ( _ 
otherBag As  OrderedBag(Of T) _ 
) As  OrderedBag(Of T)

Visual C++

public: 
OrderedBag<T>^ SymmetricDifference ( 
 OrderedBag<T>^ otherBag 
)

Parameters

otherBag

OrderedBag<(Of <T>)>

Bag to symmetric difference with.

Return Value

The symmetric difference of the two bags.
Remarks

The symmetric difference of two bags is computed in time $O(M + N)$, where $M$ is the size of the larger bag, and $N$ is the size of the smaller bag.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>This bag and <code>otherBag</code> don't use the same method for comparing items.</td>
</tr>
<tr>
<td><code>System::::ArgumentNullException</code></td>
<td><code>otherBag</code> is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>...::SymmetricDifferenceWith Method

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the symmetric difference of this bag with another bag. The symmetric difference of two bags is all items that appear in either of the bags, but not both. If an item appears X times in one bag, and Y times in the other bag, the symmetric difference contains the item AbsoluteValue(X - Y times). This bag receives the symmetric difference of the two bags; the other bag is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

public void SymmetricDifferenceWith(
    OrderedBag<T> otherBag
)

### Visual Basic (Declaration)

Public Sub SymmetricDifferenceWith ( _
    otherBag As OrderedBag(Of T) _
)

### Visual C++

public:
void SymmetricDifferenceWith ( 
    OrderedBag<T>^ otherBag
)

## Parameters

otherBag

OrderedBag<(Of <T>)>

Bag to symmetric difference with.
Remarks

The symmetric difference of two bags is computed in time $O(M + N)$, where $M$ is the size of the larger bag, and $N$ is the size of the smaller bag.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>otherBag is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>...::System.ICloneable.Clone Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a shallow clone of this bag; i.e., if items of the bag are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
private Object ICloneable.Clone()
```

**Visual Basic (Declaration)**

```vbnet
Private Function System.ICloneable.Clone As Object Implements IClone
```

**Visual C++**

```cpp
private:
virtual Object^ System.ICloneable.Clone() sealed = ICloneable::Clone
```

### Return Value

The cloned bag.
Remarks

Cloning the bag takes time $O(N)$, where $N$ is the number of items in the bag.
See Also

OrderedBag<Of<T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<>...::Union Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the union of this bag with another bag. The union of two bags is all items from both of the bags. If an item appears $X$ times in one bag, and $Y$ times in the other bag, the union contains the item $\text{Maximum}(X,Y)$ times. A new bag is created with the union of the bags and is returned. This bag and the other bag are unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public OrderedBag<T> Union(OrderedBag<T> otherBag)
```

Visual Basic (Declaration)

```vbnet
Public Function Union(otherBag As OrderedBag(Of T)) As OrderedBag(Of T)
```

Visual C++

```cpp
public: OrderedBag<T>^ Union ( OrderedBag<T>^ otherBag )
```

Parameters

otherBag

`OrderedBag<(Of <T>)->`

Bag to union with.

Return Value

The union of the two bags.
Remarks

The union of two bags is computed in time $O(M + N \log M)$, where $M$ is the size of the larger bag, and $N$ is the size of the smaller bag.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::InvalidOperationException</code></td>
<td>This bag and otherBag don't use the same method for comparing items.</td>
</tr>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>otherBag is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<(Of <T>)>::UnionWith Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the union of this bag with another bag. The union of two bags is all items from both of the bags. If an item appears X times in one bag, and Y times in the other bag, the union contains the item Maximum(X,Y) times. This bag receives the union of the two bags, the other bag is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public void UnionWith(
    OrderedBag<T> otherBag
)
```

#### Visual Basic (Declaration)

```vbnet
Public Sub UnionWith ( _
    otherBag As OrderedBag(Of T) _
)
```

#### Visual C++

```cpp
public:
void UnionWith ( 
    OrderedBag<T>^ otherBag
)
```

### Parameters

otherBag

```csharp
OrderedBag<T>>(Of <T>)>
```
Bag to union with.
Remarks

The union of two bags is computed in time $O(M + N \log M)$, where $M$ is the size of the larger bag, and $N$ is the size of the smaller bag.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>This bag and <code>otherBag</code> don't use the same method for comparing items.</td>
</tr>
<tr>
<td><code>System::::ArgumentOutOfRangeException</code></td>
<td><code>otherBag</code> is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag<(Of <T)> Class  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<(Of <T>)> Properties

See Also

☐ Include Inherited Members ☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

```
[SerializableAttribute]
public class OrderedBag<T>
```

Visual Basic (Declaration)

```
<SerializableAttribute>
Public Class OrderedBag(Of T)
```

Visual C++

```
[SerializableAttribute]
generic<typename T>
public ref class OrderedBag
```
Type Parameters

T

The type exposes the following properties.
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>📧 Comparator</td>
<td>Returns the IComparer&lt;T&gt; used to compare items in this bag.</td>
</tr>
<tr>
<td>📧 Count</td>
<td>Returns the number of items in the bag. (overrides <code>CollectionBase(Of T)</code>.</td>
</tr>
<tr>
<td>📧 Item</td>
<td>Get the item by its index in the sorted order. The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection&lt;T&gt;...::IsReadOnly</td>
<td>(Inherited from CollectionBase&lt;T&gt;.) Indicates whether the collection is synchronized. (Inherited from CollectionBase&lt;T&gt;.) Indicates the synchronization object for this collection. (Inherited from CollectionBase&lt;T&gt;.)</td>
</tr>
</tbody>
</table>
See Also

OrderedBag<Of <T>>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>::.Comparer Property

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the IComparer<> used to compare items in this bag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public IComparer<T> Comparer{ get; }
```

**Visual Basic (Declaration)**

```vbnet
Public ReadOnly Property Comparer As IComparer(Of T)
```

**Visual C++**

```cpp
public:
    property IComparer<T>^ Comparer { 
        IComparer<T>^ get ();
    }
```

**Field Value**

If the bag was created using a comparer, that comparer is returned. If the bag was created using a comparison delegate, then a comparer equivalent to that delegate is returned. Otherwise the default comparer for T (Comparer<T>.Default) is returned.
See Also

OrderedBag<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of T>.Count Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the number of items in the bag.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed int Count{ get;}

Visual Basic (Declaration)

Public Overrides NotOverridable ReadOnly Property Count As Integer

Visual C++

public:
virtual property int Count {
    int get () override sealed;
}

Field Value

The number of items in the bag.
Remarks

The size of the bag is returned in constant time.
See Also

**OrderedBag<Of <T>>** Class  
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
OrderedBag(Of T)>...::Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Get the item by its index in the sorted order. The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public T this[
    int index
] { get; }

Visual Basic (Declaration)

Public ReadOnly Default Property Item ( _
    index As Integer _
) As T

Visual C++

public:
property T default[int index] {
    T get (int index);
}

Parameters

index

Int32
The index to get the item by.

Return Value

The item at the given index.
Remarks

The indexer takes time $O(\log N)$, which $N$ is the number of items in the set.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>index is less than zero or greater than or equal to Count.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of Of T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
The OrderedBag<T>.View class is used to look at a subset of the items inside an ordered bag. It is returned from the Range, RangeTo, RangeFrom, and Reversed methods.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

[DllImport("system.runtime.serialization")]
public class View : CollectionBase<T>

Visual Basic (Declaration)

<SerializableAttribute>
Public Class View 
    Inherits CollectionBase(Of T)

Visual C++

[DllImport("system.runtime.serialization")]
public ref class View : public CollectionBase<T>
Remarks

Views are dynamic. If the underlying bag changes, the view changes in sync. If a change is made to the view, the underlying bag changes accordingly.

Typically, this class is used in conjunction with a foreach statement to enumerate the items in a subset of the OrderedBag. For example:

```csharp
foreach(T item in bag.Range(from, to)) {
    // process item
}
```
Inheritance Hierarchy

System..:::Object
Wintellect.PowerCollections..:::CollectionBase(Of <T>)
Wintellect.PowerCollections..:::OrderedBag(Of <T>)..:::View
See Also

OrderedBag(Of <T>):::View Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>...::View Members

See Also  Methods  Properties  Explicit Interface Implementations

Include Inherited Members  Include Protected Members

.NET Compact Framework Members Only

XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
**Syntax**

**C#**

```csharp
[SerializableAttribute]
public class View
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute> _
Public Class View
```

**Visual C++**

```c++
[SerializableAttribute]
public ref class View
```

OrderedBag<T> is a collection that contains items of type T. The item are maintained in a sorted order. Unlike a OrderedSet, duplicate items (items that compare equal to each other) are allows in an OrderedBag.

The `OrderedBag<Of <T>>` generic type exposes the following members.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Must be overridden to allow adding items to this collection. Get a read-only list view of the items in this view. The items in the list are in sorted order, with the smallest item at index 0. This view does not copy any data, and reflects any changes to the underlying OrderedSet.</td>
</tr>
<tr>
<td><strong>AsList</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection{T} provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase(Of &lt;T&gt;).)</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Removes all the items within this view from the underlying bag. (Overates CollectionBase(Of &lt;T&gt;).::Clear())</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable{T}.Equals or Object.Equals.</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from CollectionBase(Of &lt;T&gt;).)</td>
</tr>
<tr>
<td><strong>ConvertAll(Of &lt;TOutput&gt;)</strong></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase(Of &lt;T&gt;).)</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from CollectionBase(Of &lt;T&gt;).)</td>
</tr>
</tbody>
</table>
Determines whether the specified **Object** is equal to the current **Object**.

(Inherited from **Object**.)

**Exists**

Determines if the collection contains any item that satisfies the condition defined by predicate.

(Inherited from **CollectionBase<*>(Of<T>).**)

**FindAll**

Enumerates the items in the collection that satisfy the condition defined by predicate.

(Inherited from **CollectionBase<*>(Of<T>).**)

**ForEach**

Performs the specified action on each item in this collection.

(Inherited from **CollectionBase<*>(Of<T>).**)

Enumerate all the items in this view.

**GetEnumerator**

(Overrides **CollectionBase<*>(Of<T>).**::* **GetEnumerator**)

**GetFirst**

Returns the first item in this view: the item that would appear first if the view was enumerated.

Serves as a hash function for a particular type.

**GetHashCode**

**GetHashCode** is suitable for use in hashing algorithms and data structures like a hash table.

(Inherited from **Object**.)

**GetLast**

Returns the last item in the view: the item that would appear last if the view was enumerated.

**GetType**

Gets the **Type** of the current instance.

(Inherited from **Object**.)

**IndexOf**

Get the first index of the given item in the view. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count-1.

**LastIndexOf**

Get the last index of the given item in the view. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count-1.

**Remove**

Must be overridden to allow removing items from this collection.

**RemoveAll**

Removes all the items in the collection that satisfy the condition defined by predicate.

(Inherited from **CollectionBase<*>(Of<T>).**)

**Reversed**

Creates a new View that has the same items as this view, in
the reversed order.
Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.
(Inherited from `CollectionBase<Of <T>>`.)

- **ToArray**

Shows the string representation of the collection. The string representation contains a list of the items in the collection.
(Contained collections (except string) are expanded recursively.
(Inherited from `CollectionBase<Of <T>>`.)

- **ToString**

Determines if all of the items in the collection satisfy the condition defined by predicate.
(Inherited from `CollectionBase<Of <T>>`.)

- **TrueForAll**
# Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![image] Count</td>
<td>Number of items in this view. (Overrides <code>CollectionBase&lt;Of &lt;T&gt;&gt;::Count</code>.)</td>
</tr>
<tr>
<td>![image] Item</td>
<td>Get the item by its index in the sorted order. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count-1.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection(Of T)::.IsReadOnly</td>
<td>(Inherited from CollectionBase(Of T).) Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.</td>
</tr>
<tr>
<td>ICollection::.CopyTo</td>
<td>(Inherited from CollectionBase(Of T).) Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td>ICollection::.IsSynchronized</td>
<td>(Inherited from CollectionBase(Of T).) Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td>ICollection::.SyncRoot</td>
<td>(Inherited from CollectionBase(Of T).) Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection.</td>
</tr>
<tr>
<td>IEnumerable::.GetEnumerator</td>
<td>(Inherited from CollectionBase(Of T).)</td>
</tr>
</tbody>
</table>
See Also

OrderedBag<Of <T>>:::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag(Of T)<...::View Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class View

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class View

Visual C++

[SerializableAttribute]
public ref class View

The OrderedBag<Of <T>> generic type exposes the following methods.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>❗️ Add</td>
<td>Must be overridden to allow adding items to this collection.</td>
</tr>
<tr>
<td>❗️ AsList</td>
<td>Get a read-only list view of the items in this view. The items in the list are in sorted order, with the smallest item at index 0. This view does not copy any data, and reflects any changes to the underlying OrderedSet.</td>
</tr>
<tr>
<td>❗️ AsReadOnly</td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>❗️ Clear</td>
<td>Removes all the items within this view from the underlying bag.</td>
</tr>
<tr>
<td>❗️ Contains</td>
<td>Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable&lt;T&gt;.Equals or Object.Equals.</td>
</tr>
<tr>
<td>❗️ ConvertAll&lt;Of &lt;TOutput&gt;&gt;</td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>❗️ CopyTo</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>❗️ CountWhere</td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a> is equal to the current <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>. (Inherited from <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.)</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate. (Inherited from [CollectionBase]&lt;(Of &lt;T&gt;)).</td>
</tr>
<tr>
<td><strong>FindAll</strong></td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from [CollectionBase]&lt;(Of &lt;T&gt;)).</td>
</tr>
<tr>
<td><strong>ForEach</strong></td>
<td>Performs the specified action on each item in this collection. (Inherited from [CollectionBase]&lt;(Of &lt;T&gt;)). Enumerate all the items in this view.</td>
</tr>
<tr>
<td><strong>GetEnumerator</strong></td>
<td>Enumerate all the items in this view. (Overrides [CollectionBase]&lt;(Of &lt;T&gt;):-::{GetEnumerator}().)</td>
</tr>
<tr>
<td><strong>GetFirst</strong></td>
<td>Returns the first item in this view: the item that would appear first if the view was enumerated.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. [GetHashCode()] is suitable for use in hashing algorithms and data structures like a hash table. (Inherited from <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.)</td>
</tr>
<tr>
<td><strong>GetLast</strong></td>
<td>Returns the last item in the view: the item that would appear last if the view was enumerated.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <a href="https://docs.microsoft.com/en-us/dotnet/api/system.type">Type</a> of the current instance. (Inherited from <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.)</td>
</tr>
<tr>
<td><strong>IndexOf</strong></td>
<td>Get the first index of the given item in the view. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count-1.</td>
</tr>
<tr>
<td><strong>LastIndexOf</strong></td>
<td>Get the last index of the given item in the view. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count-1.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Must be overridden to allow removing items from this collection.</td>
</tr>
<tr>
<td><strong>RemoveAll</strong></td>
<td>Removes all the items in the collection that satisfy the condition defined by predicate. (Inherited from [CollectionBase]&lt;(Of &lt;T&gt;)).</td>
</tr>
<tr>
<td><strong>Reversed</strong></td>
<td>Creates a new View that has the same items as this view, in</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>ToArray</strong></td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from <code>CollectionBase(Of &lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Shows the string representation of the collection. The string representation contains a list of the items in the collection. Contained collections (except string) are expanded recursively. (Inherited from <code>CollectionBase(Of &lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><strong>TrueForAll</strong></td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from <code>CollectionBase(Of &lt;T&gt;)</code>.)</td>
</tr>
</tbody>
</table>
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
### Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ICollection</strong>::&lt;code&gt;::CopyTo&lt;/code&gt;</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>IEnumerable</strong>::&lt;code&gt;::GetEnumerator&lt;/code&gt;</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;<code>T</code>&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of T)>::<View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Get a read-only list view of the items in this view. The items in the list are in sorted order, with the smallest item at index 0. This view does not copy any data, and reflects any changes to the underlying OrderedSet.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public IList<T> AsList()
```

**Visual Basic (Declaration)**

```vbnet
Public Function AsList As IList(Of T)
```

**Visual C++**

```c++
public:
IList<T>^ AsList ()
```

**Return Value**

A read-only IList<T> view onto this view.
See Also

OrderedBag<(Of <T>)>..::.View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>...::View...::Clear Method
See Also  Example

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all the items within this view from the underlying bag.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed void Clear()

Visual Basic (Declaration)

Public Overrides NotOverridable Sub Clear

Visual C++

public:
virtual void Clear () override sealed
Examples

The following removes all the items that start with "A" from an OrderedBag.

Copy Code

    bag.Range("A", "B").Clear();
See Also

**OrderedBag<(Of <T>)>::View** Class
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
OrderedBag<(Of <T>)>::View<>::GetEnumerator Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerate all the items in this view.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed IEnumerator&lt;T&gt; GetEnumerator()

Visual Basic (Declaration)

Public Overrides NotOverridable Function GetEnumerator As IEnumerator

Visual C++

public:
virtual IEnumerator&lt;T&gt;; GetEnumerator () override sealed

Return Value

An IEnumerator&lt;T&gt; with the items in this view.
See Also

OrderedBag<(Of <T>),...::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Returns the first item in this view: the item that would appear first if the view was enumerated.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public T GetFirst()

Visual Basic (Declaration)

Public Function GetFirst As T

Visual C++

public:
T GetFirst ()

Return Value

The first item in the view.
Remarks

GetFirst() takes time $O(\log N)$, where $N$ is the number of items in the bag.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>The view has no items in it.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of T)<> View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the last item in the view: the item that would appear last if the view was enumerated.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public T GetLast()

**Visual Basic (Declaration)**

Public Function GetLast As T

**Visual C++**

public:
T GetLast ()

**Return Value**

The last item in the view.
Remarks

GetLast() takes time $O(\log N)$, where $N$ is the number of items in the bag.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::InvalidOperationException</code></td>
<td>The view has no items in it.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag<(Of <T>)>::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Get the first index of the given item in the view. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count-1.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public int IndexOf(
    T item
)

Visual Basic (Declaration)

Public Function IndexOf ( _
    item As T _
) As Integer

Visual C++

public:
int IndexOf ( 
    T item
)

Parameters

item
    T
    The item to get the index of.

Return Value

The index of the first item in the view equal to item, or -1 if the item is not present in the view.
Remarks

Finding the index takes time $O(\log N)$, which $N$ is the number of items in the set.
See Also

OrderedBag<Of <T>>:::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>...::View...::LastIndexOf Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Get the last index of the given item in the view. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count-1.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public int LastIndexOf(
    T item
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function LastIndexOf ( _
    item As T _
) As Integer
```

**Visual C++**

```cpp
public:
int LastIndexOf ( 
    T item
)
```

**Parameters**

- **item**
  
  The item to get the index of.

**Return Value**

The index of the last item in the view equal to item, or -1 if the item is not present in the view.
Remarks

Finding the index takes time $O(\log N)$, which $N$ is the number of items in the set.
See Also

See Also

OrderedBag(Of T)> View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of `<T>`>...::View...::Reversed Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new View that has the same items as this view, in the reversed order.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public OrderedBag<T>.OrderedBag<(Of <T>)>::::View Reversed()
```

**Visual Basic (Declaration)**

```vbnet
Public Function Reversed As OrderedBag<(Of <T>)>::::View
```

**Visual C++**

```cpp
public: 
OrderedBag<(Of <T>)>::::View^ Reversed ()
```

### Return Value

A new View that has the reversed order of this view, with the same upper and lower bounds.
See Also

OrderedBag(Of T)>;, View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>::View Properties

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
- Syntax

**C#**

```csharp
[SerializableAttribute]
public class View
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute> _
Public Class View
```

**Visual C++**

```cpp
[SerializableAttribute]
public ref class View
```

The `OrderedBag<Of <T>>` generic type exposes the following properties.
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Count" /> <strong>Count</strong></td>
<td>Number of items in this view. (Overrides <a href="CollectionBase">CollectionBase</a>&lt;(Of &lt;T&gt;)&gt;::Count.)</td>
</tr>
<tr>
<td><img src="image" alt="Item" /> <strong>Item</strong></td>
<td>Get the item by its index in the sorted order. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count-1.</td>
</tr>
</tbody>
</table>
Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;T&gt;</code>::<code>IsReadOnly</code></td>
<td>(Inherited from <code>CollectionBase&lt;T&gt;</code>.) Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td><code>ICollection</code>::<code>IsSynchronized</code></td>
<td>(Inherited from <code>CollectionBase&lt;T&gt;</code>.) Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td><code>ICollection</code>::<code>SyncRoot</code></td>
<td>(Inherited from <code>CollectionBase&lt;T&gt;</code>.)</td>
</tr>
</tbody>
</table>
See Also

OrderedBag<Of T> View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<Of <T>>...::View...::Count Property

See Also

☐ Visual Basic (Declaration) ☐ Visual Basic (Usage)
☐ C#
☐ Visual C++
☐ J#
☐ JScript
☐ XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Number of items in this view.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed int Count{ get;}

Visual Basic (Declaration)

Public Overrides NotOverridable ReadOnly Property Count As Integer

Visual C++

public:
virtual property int Count {
    int get () override sealed;
}

Field Value

Number of items that lie within the bounds the view.
See Also

OrderedBag<\(\text{Of } \langle T \rangle\)> View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedBag<(Of <T>)>.::View::.::Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Get the item by its index in the sorted order. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count-1.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
- **Syntax**

  **C#**

  ```csharp
  public T this[
      int index
  ]{ get; }
  ```

  **Visual Basic (Declaration)**

  ```vbnet
  Public ReadOnly Default Property Item ( _
      index As Integer _
  ) As T
  ```

  **Visual C++**

  ```cpp
  public:
  property T default[int index] { 
      T get (int index);
  }
  ```

- **Parameters**

  index

  `Int32`

  The index to get the item by.

- **Return Value**

  The item at the given index.
Remarks

The indexer takes time $O(\log N)$, which $N$ is the number of items in the set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentOutOfRangeException</code></td>
<td>index is less than zero or greater than or equal to Count.</td>
</tr>
</tbody>
</table>
See Also

OrderedBag(Of T) View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary(Of<TKey, TValue>) Class  

See Also  Members  

Visual Basic (Declaration)  Visual Basic (Usage)  C#  Visual C++  J#  JScript  XAML  

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

OrderedDictionary<TKey, TValue> is a collection that maps keys of type TKey to values of type TValue. The keys are maintained in a sorted order, and at most one value is permitted for each key.

Namespace: Wintellect.PowerCollections  
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

[SerializableAttribute]
public class OrderedDictionary<TKey, TValue> : DictionaryBase<TKey, TValue>, ICloneable

Visual Basic (Declaration)

<SerializableAttribute>
Public Class OrderedDictionary(Of TKey, TValue) _
   Inherits DictionaryBase(Of TKey, TValue) _
   Implements ICloneable

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue> public ref class OrderedDictionary : public DictionaryBase<TKey, TValue>, ICloneable
Type Parameters

TKey
TValue
Remarks

The keys are compared in one of three ways. If TKey implements IComparable<TKey> or IComparable, then the CompareTo method of that interface will be used to compare elements. Alternatively, a comparison function can be passed in either as a delegate, or as an instance of IComparer<TKey>.

OrderedDictionary is implemented as a balanced binary tree. Inserting, deleting, and looking up an element all are done in log(N) type, where N is the number of keys in the tree.

Dictionary(Of TKey, TValue) is similar, but uses hashing instead of comparison, and does not maintain the keys in sorted order.
Inheritance Hierarchy

System::Object
Wintellect.PowerCollections::CollectionBase(Of KeyValuePair(Of TKey, TValue))>
Wintellect.PowerCollections::DictionaryBase(Of TKey, TValue>)
Wintellect.PowerCollections::OrderedDictionary(Of TKey, TValue)>
See Also

OrderedDictionary(Of TKey, TValue)> Members
Wintellect.PowerCollections Namespace
System.Collections.Generic:::Dictionary(Of TKey, TValue)>-

Send comments about this topic to Microsoft.
OrderedDictionary(Of TKey, TValue)> Members

See Also  Methods  Properties  Constructors  Explicit Interface Implementations

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class OrderedDictionary<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute>
Public Class OrderedDictionary(Of TKey, TValue)

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class OrderedDictionary
Type Parameters

TKey
TValue

The type exposes the following members.
## Public Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OrderedDictionary&lt;(&lt;TKey, TValue&gt;)&gt;</code></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><code>OrderedDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;New</code></td>
<td></td>
</tr>
</tbody>
</table>
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✶ Add</td>
<td>Overloaded. Adds multiple key-value pairs to a dictionary. If a key exists in both the current instance and dictionaryToAdd, then the value is updated with the value from keysAndValues&gt; (no exception is thrown). Since IDictionary&lt;TKey,TValue&gt; inherits from IEnumerable&lt;KeyValuePair&lt;TKey,TValue&gt;&gt;, this method can be used to merge one dictionary into another.</td>
</tr>
<tr>
<td>✶ AddMany</td>
<td>Provides a read-only view of this dictionary. The returned IDictionary&lt;TKey,TValue&gt; provides a view of the dictionary that prevents modifications to the dictionary.</td>
</tr>
<tr>
<td>✶ AsReadOnly</td>
<td>Use the method to provide access to the dictionary without allowing changes. Since the returned object is just a view, changes to the dictionary will be reflected in the view. (Inherited from DictionaryBase(Of TKey, TValue)).</td>
</tr>
<tr>
<td>✶ Clear</td>
<td>Removes all keys and values from the dictionary. (Overrides DictionaryBase(Of TKey, TValue) Clear().)</td>
</tr>
<tr>
<td>✶ Clone</td>
<td>Makes a shallow clone of this dictionary; i.e., if keys or values of the dictionary are reference types, then they are not cloned. If TKey or TValue is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td>✶ CloneContents</td>
<td>Makes a deep clone of this dictionary. A new dictionary is created with a clone of each entry of this dictionary, by calling ICloneable.Clone on each element. If TKey or TValue is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td>✶ Contains</td>
<td>Determines if a dictionary contains a given KeyValuePair. This implementation checks to see if the dictionary contains the given key, and if the value associated with the key is equal to (via object.Equals) the value. (Inherited from DictionaryBase(Of TKey, TValue)).</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>ContainsKey</strong></td>
<td>Determines if this dictionary contains a key equal to key. The dictionary is not changed. (Overrides <code>DictionaryBase&lt;Of TKey, TValue&gt;..::ContainsKey(TKey).</code>)</td>
</tr>
<tr>
<td><strong>ConvertAll&lt;Of TOutput&gt;</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;.</code>)</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;.</code>)</td>
</tr>
<tr>
<td><strong>CountWhere</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;.</code>)</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;.</code>)</td>
</tr>
<tr>
<td><strong>FindAll</strong></td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;.</code>)</td>
</tr>
<tr>
<td><strong>ForEach</strong></td>
<td>Performs the specified action on each item in this collection. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;.</code>)</td>
</tr>
<tr>
<td><strong>GetEnumerator</strong></td>
<td>Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a <code>KeyValuePair&lt;TKey,TValue&gt;</code>. The entries are enumerated in the sorted order of the keys. (Overrides <code>CollectionBase&lt;Of &lt;KeyValuePair&lt;Of TKey, TValue&gt;&gt;..::GetEnumerator()()</code>)</td>
</tr>
</tbody>
</table>

Serves as a hash function for a particular type. `GetHashCode()` is suitable for use in hashing algorithms.
- GetHashCode
  and data structures like a hash table.
  (Inherited from Object.)

- GetType
  Gets the Type of the current instance.
  (Inherited from Object.)
  Finds a key in the dictionary. If the dictionary already
  contains a key equal to the passed key, then the existing
  value is returned via value. If the dictionary doesn't contain
  that key, then value is associated with that key.

- GetValueElseAdd
  Returns a collection that can be used for enumerating some
  of the keys and values in the collection. Only keys that are
  greater than from and less than to are included. The keys
  are enumerated in sorted order. Keys equal to the end
  points of the range can be included or excluded depending
  on the fromInclusive and toInclusive parameters.

- Range
  Returns a collection that can be used for enumerating some
  of the keys and values in the collection. Only keys that are
  greater than (and optionally, equal to) from are included.

- RangeFrom
  The keys are enumerated in sorted order. Keys equal to from
  can be included or excluded depending on the fromInclusive parameter.

- RangeTo
  Returns a collection that can be used for enumerating some
  of the keys and values in the collection. Only items that are
  less than (and optionally, equal to) to are included. The
  items are enumerated in sorted order. Items equal to to can
  be included or excluded depending on the toInclusive parameter.

- Remove
  Removes a key from the dictionary. This method must be
  overridden in the derived class.

- RemoveAll
  Removes all the items in the collection that satisfy the
  condition defined by predicate.
  (Inherited from CollectionBase<Of <T>>.)

- RemoveMany
  Removes all the keys found in another collection (such as
  an array or List<TKey>). Each key in
  keyCollectionToRemove is removed from the dictionary.
  Keys that are not present are ignored.

Changes the value associated with a given key. If the
Replace
dictionary does not contain a key equal to the passed key, then an ArgumentException is thrown.

Reversed
Returns a View collection that can be used for enumerating the keys and values in the collection in reversed order.

ToArray
Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from CollectionBase<>().)

ToString
Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary. (Inherited from DictionaryBase<>().)

TrueForAll
Determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from CollectionBase<>().)

TryGetValue
Determines if this dictionary contains a key equal to key. If so, the value associated with that key is returned through the value parameter. (Overrides DictionaryBase<>()....:TryGetValue(TKey, TValue%).)
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Finalize" /></td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><img src="image" alt="MemberwiseClone" /></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comparer</strong></td>
<td>Returns the IComparer&lt;T&gt; used to compare keys in this dictionary. Returns the number of keys in the dictionary. (Overrides <code>CollectionBase&lt;Of &lt;KeyValuePair&lt;TKey,TValue&gt;&gt;&gt;.Count</code>.)</td>
</tr>
<tr>
<td><strong>Count</strong></td>
<td>Gets or sets the value associated with a given key. When getting a value, if this key is not found in the collection, then an ArgumentException is thrown. When setting a value, the value replaces any existing value in the dictionary. (Overrides <code>DictionaryBase&lt;Of &lt;TKey,TValue&gt;&gt;.Item[[TKey]].</code>.)</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Returns a collection of the keys in this dictionary. (Inherited from <code>DictionaryBase&lt;Of &lt;TKey,TValue&gt;&gt;</code>.) Returns a collection of the values in this dictionary. The ordering of values in this collection is the same as that in the Keys collection. (Inherited from <code>DictionaryBase&lt;Of &lt;TKey,TValue&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>Keys</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Values</strong></td>
<td></td>
</tr>
</tbody>
</table>
# Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection&lt;(Of &lt;T&gt;&gt;::&lt;IsReadOnly</td>
<td>(Inherited from CollectionBase&lt;(Of &lt;T&gt;&gt;).) Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;(Of &lt;T&gt;&gt;).)</td>
</tr>
<tr>
<td>ICollection::&lt;CopyTo</td>
<td>Indicates whether the collection is synchronized. (Inherited from CollectionBase&lt;(Of &lt;T&gt;&gt;).)</td>
</tr>
<tr>
<td>ICollection::&lt;IsSynchronized</td>
<td>Indicates the synchronization object for this collection. (Inherited from CollectionBase&lt;(Of &lt;T&gt;&gt;).)</td>
</tr>
<tr>
<td>ICollection::&lt;SyncRoot</td>
<td>Adds a key-value pair to the collection. If key or value are not of the expected types, an ArgumentException is thrown. If both key and value are of the expected types, the (overridden) Add method is called with the key and value to add. (Inherited from DictionaryBase&lt;(Of &lt;TKey, TValue)&gt;&gt;).)</td>
</tr>
<tr>
<td>IDictionary::&lt;Add</td>
<td>Clears this dictionary. Calls the (overridden) Clear method. (Inherited from DictionaryBase&lt;(Of &lt;TKey, TValue)&gt;&gt;).)</td>
</tr>
<tr>
<td>IDictionary::&lt;Clear</td>
<td>Determines if this dictionary contains a key equal to key. The dictionary is not changed. Calls the (overridden) ContainsKey method.</td>
</tr>
<tr>
<td>IDictionary::&lt;Contains</td>
<td>If key is not of the correct TKey for the</td>
</tr>
</tbody>
</table>
`GetEnumerator` - Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method. (Inherited from `DictionaryBase<Of <TKey, TValue>>`.)

`IsFixedSize` - Returns whether this dictionary is fixed size. This implemented always returns false. (Inherited from `DictionaryBase<Of <TKey, TValue>>`.)

`IsReadOnly` - Returns if this dictionary is read-only. This implementation always returns false. (Inherited from `DictionaryBase<Of <TKey, TValue>>`.)

`Item` - Gets or sets the value associated with a given key. When getting a value, if this key is not found in the collection, then null is returned. When setting a value, the value replaces any existing value in the dictionary. If either the key or value are not of the correct type for this dictionary, an ArgumentException is thrown. (Inherited from `DictionaryBase<Of <TKey, TValue>>`.)

`Keys` - Returns a collection of all the keys in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method. (Inherited from `DictionaryBase<Of <TKey, TValue>>`.)

Removes the key (and associated value) from the collection that is equal to the passed in key. If no key in the dictionary is equal to the passed key, the dictionary is
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDictionary::&lt;Remove&gt;</td>
<td>Calls the (overridden) Remove method. If key is not of the correct TKey for the dictionary, the dictionary is unchanged. (Inherited from DictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td>IDictionary::&lt;Values&gt;</td>
<td>Returns a collection of all the values in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method. (Inherited from DictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td>IEnumerable::&lt;GetEnumerator&gt;</td>
<td>Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method. (Inherited from DictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td>ICloneable::&lt;Clone&gt;</td>
<td>Implements ICloneable.Clone. Makes a shallow clone of this dictionary; i.e., if keys or values are reference types, then they are not cloned.</td>
</tr>
</tbody>
</table>
See Also

OrderedDictionary(Of TKey, TValue)
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<(Of <TKey, TValue>)> Constructor

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OrderedDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;OrderedDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;New()</code></td>
<td>Creates a new <code>OrderedDictionary</code>. The <code>TKey</code> must implemented <code>IComparable&lt;TKey&gt;</code> or <code>IComparable</code>. The <code>CompareTo</code> method of this interface will be used to compare keys in this dictionary.</td>
</tr>
<tr>
<td><code>OrderedDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;OrderedDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;New(IComparer&lt;(Of &lt;TKey, TValue&gt;)&gt;)</code></td>
<td>Creates a new <code>OrderedDictionary</code>. The Compare method of the passed comparison object will be used to compare keys in this dictionary.</td>
</tr>
<tr>
<td><code>OrderedDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;OrderedDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;New(IEnumerable&lt;(Of KeyValuePair&lt;(Of &lt;TKey, TValue&gt;)&gt;&gt;)&gt;)</code></td>
<td>Creates a new <code>OrderedDictionary</code>. The <code>Compare</code> method of the passed comparer object will be used to compare keys in this dictionary. A collection and keys and values (typically another <code>dictionary</code>) is used to initialized the contents of the dictionary.</td>
</tr>
<tr>
<td><code>OrderedDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;OrderedDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;New(Comparison&lt;(Of &lt;TKey&gt;)&gt;)</code></td>
<td>Creates a new <code>OrderedDictionary</code>. The passed delegate will be used to compare keys in this dictionary.</td>
</tr>
</tbody>
</table>
| `OrderedDictionary<(Of <TKey, TValue>)>OrderedDictionary<(Of <TKey, TValue>)>New()()` | Creates a new `OrderedDictionary`. The `CompareTo` method of this interface will be used to compare keys in this dictionary.
**OrderedDictionary**<sup>(Of &lt;TKey, TValue&gt;)</sup>&gt;New(Enumerable&lt;(Of &lt;KeyValuePair&lt;(Of &lt;TKey, TValue&gt;&gt;&gt;)&gt;), IComparer&lt;(Of &lt;TKey&gt;&gt;))

Creates a new OrderedDictionary. The passed delegate will be used to compare keys in this dictionary.

A collection and keys and values (typically another dictionary) is used to initialized the contents of the dictionary.

**OrderedDictionary**<sup>(Of &lt;TKey, TValue&gt;)</sup>&gt;OrderedDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;New(Enumerable&lt;(Of &lt;KeyValuePair&lt;(Of &lt;TKey, TValue&gt;&gt;&gt;)), Comparison&lt;(Of &lt;TKey&gt;&gt;))

compare keys in this dictionary.

A collection and keys and values (typically another dictionary) is used to initialized the contents of the dictionary.
See Also

OrderedDictionary(Of TKey, TValue)> Class
OrderedDictionary(Of TKey, TValue)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary(Of TKey, TValue)> Constructor

See Also

- Visual Basic (Declaration) - Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

 Creates a new OrderedDictionary. The TKey must implemented IComparable<TKey> or IComparable. The CompareTo method of this interface will be used to compare keys in this dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedDictionary()

Visual Basic (Declaration)

Public Sub New

Visual C++

public:
OrderedDictionary ()
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::InvalidOperationException</td>
<td>TKey does not implement IComparable&lt;TKey&gt;.</td>
</tr>
</tbody>
</table>
See Also

**OrderedDictionary(Of TKey, TValue)** Class
**Wintellect.PowerCollections** Namespace

Send comments about this topic to Microsoft.
OrderedDictionary(Of TKey, TValue) Constructor (IComparer(Of TKey))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedDictionary. The Compare method of the passed comparison object will be used to compare keys in this dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedDictionary(
    IComparer<TKey> comparer
)

Visual Basic (Declaration)

Public Sub New (_
    comparer As IComparer(Of TKey) _
)

Visual C++

public:
OrderedDictionary (  
    IComparer<TKey>^ comparer  
)

Parameters

comparer
    IComparer<Of <TKey>>
    An instance of IComparer<TKey> that will be used to compare keys.
Remarks

The GetHashCode and Equals methods of the provided IComparer<TKey> will never be called, and need not be implemented.
See Also

**OrderedDictionary(Of<TKey, TValue>)** Class
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
OrderedDictionary<(Of <TKey, TValue>))> Constructor (IEnumerable<(Of <KeyValuePair<(Of <TKey, TValue>))>)>

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedDictionary. The TKey must implemented IComparable<TKey> or IComparable. The CompareTo method of this interface will be used to compare keys in this dictionary.

A collection and keys and values (typically another dictionary) is used to initialized the contents of the dictionary.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedDictionary(
    IEnumerable<KeyValuePair<TKey, TValue>> keysAndValues
)

Visual Basic (Declaration)

Public Sub New ( _
    keysAndValues As IEnumerable(Of KeyValuePair(Of TKey, TValue))
)

Visual C++

public:
    OrderedDictionary (  
        IEnumerable<KeyValuePair<TKey, TValue>>^ keysAndValues
    )

Parameters

keysAndValues
    IEnumerable<(Of <KeyValuePair(Of TKey, TValue)>)>
    A collection of keys and values whose contents are used to initialized the dictionary.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>TKey does not implement <code>IComparable&lt;TKey&gt;</code></td>
</tr>
</tbody>
</table>
See Also

**OrderedDictionary(Of TKey, TValue) Class**

**Wintellect.PowerCollections Namespace**

Send [comments](#) about this topic to Microsoft.
OrderedDictionary<Of <TKey, TValue>> Constructor (Comparison<Of <TKey>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedDictionary. The passed delegate will be used to compare keys in this dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public OrderedDictionary(
    Comparison<TKey> comparison
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub New ( _
    comparison As Comparison(Of TKey) _
)
```

**Visual C++**

```cpp
public:
OrderedDictionary ( 
    Comparison<TKey>^ comparison
)
```

### Parameters

`comparison`  
`Comparison(Of TKey)>`  
A delegate to a method that will be used to compare keys.
**See Also**

`OrderedDictionary(Of TKey, TValue)>` Class  
`Wintellect.PowerCollections` Namespace

Send [comments](mailto:microsoft.com) about this topic to Microsoft.
OrderedDictionary<(Of <TKey, TValue>)> Constructor (IEnumerable<(Of <KeyValuePair<(Of <TKey, TValue>)>>>, IComparer<(Of <TKey>)>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedDictionary. The Compare method of the passed comparison object will be used to compare keys in this dictionary.

A collection and keys and values (typically another dictionary) is used to initialized the contents of the dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public OrderedDictionary(
    IEnumerable<KeyValuePair<TKey,TValue>> keysAndValues,
    IComparer<TKey> comparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub New (_
    keysAndValues As IEnumerable(Of KeyValuePair(Of TKey, TValue)),
    comparer As IComparer(Of TKey) _
)
```

**Visual C++**

```cpp
public: 
    OrderedDictionary ( 
        IEnumerable<KeyValuePair<TKey,TValue>>^ keysAndValues, 
        IComparer<TKey>^ comparer 
    )
```

### Parameters

**keysAndValues**

`IEnumerable<KeyValuePair<TKey,TValue>>` A collection of keys and values whose contents are used to initialized the dictionary.

**comparer**

`IComparer<TKey>` An instance of IComparer<TKey> that will be used to compare keys.
Remarks

The GetHashCode and Equals methods of the provided IComparer<TKey> will never be called, and need not be implemented.
See Also

OrderedDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary(Of<TKey, TValue>) Constructor (IEnumerable(Of KeyValuePair(Of<TKey, TValue>)), Comparison(Of<TKey>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedDictionary. The passed delegate will be used to compare keys in this dictionary.

A collection and keys and values (typically another dictionary) is used to initialized the contents of the dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
- Syntax

**C#**

```csharp
public OrderedDictionary(
    IEnumerable<KeyValuePair<TKey, TValue>> keysAndValues,
    Comparison<TKey> comparison
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub New (_
    keysAndValues As IEnumerable(Of KeyValuePair(Of TKey, TValue))
    comparison As Comparison(Of TKey) _
)
```

**Visual C++**

```cpp
public: 
    OrderedDictionary ( 
        IEnumerable<KeyValuePair<TKey, TValue>>^ keysAndValues, 
        Comparison<TKey>^ comparison 
    )
```

**Parameters**

- **keysAndValues**
  ```csharp
  IEnumerable<KeyValuePair<TKey, TValue>>
  ```

  A collection of keys and values whose contents are used to initialized the dictionary.

- **comparison**
  ```csharp
  Comparison<TKey>
  ```

  A delegate to a method that will be used to compare keys.
See Also

OrderedDictionary<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<Of <TKey, TValue>> Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
- **Syntax**

**C#**

```csharp
[SerializableAttribute]
public class OrderedDictionary<TKey, TValue>
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute>
Public Class OrderedDictionary(Of TKey, TValue)
```

**Visual C++**

```cpp
[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class OrderedDictionary
```
Type Parameters

TKey
TValue

The type exposes the following methods.
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Overloaded. Adds multiple key-value pairs to a dictionary. If a key exists in both the current instance and dictionaryToAdd, then the value is updated with the value from keysAndValues&gt; (no exception is thrown). Since IDictionary&lt;TKey,TValue&gt; inherits from IEnumerable&lt;KeyValuePair&lt;TKey,TValue&gt;&gt;, this method can be used to merge one dictionary into another.</td>
</tr>
<tr>
<td>AddMany</td>
<td>Provides a read-only view of this dictionary. The returned IDictionary&lt;TKey,TValue&gt; provides a view of the dictionary that prevents modifications to the dictionary. Use the method to provide access to the dictionary without allowing changes. Since the returned object is just a view, changes to the dictionary will be reflected in the view. (Inherited from DictionaryBase(Of TKey, TValue&gt;).)</td>
</tr>
<tr>
<td>AsReadOnly</td>
<td>Use the method to provide access to the dictionary without allowing changes. Since the returned object is just a view, changes to the dictionary will be reflected in the view. (Inherited from DictionaryBase(Of TKey, TValue&gt;).)</td>
</tr>
<tr>
<td>Clear</td>
<td>Removes all keys and values from the dictionary. (Overrides DictionaryBase(Of TKey, TValue&gt;)::.Clear().)</td>
</tr>
<tr>
<td>Clone</td>
<td>Makes a shallow clone of this dictionary; i.e., if keys or values of the dictionary are reference types, then they are not cloned. If TKey or TValue is a value type, then each element is copied as if by simple assignment. Makes a deep clone of this dictionary. A new dictionary is created with a clone of each entry of this dictionary, by calling ICloneable.Clone on each element. If TKey or TValue is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td>CloneContents</td>
<td>Determines if a dictionary contains a given KeyValuePair. This implementation checks to see if the dictionary contains the given key, and if the value associated with the key is equal to (via object.Equals) the value. (Inherited from DictionaryBase(Of TKey, TValue&gt;).)</td>
</tr>
</tbody>
</table>
**ContainsKey**

Determines if this dictionary contains a key equal to key. The dictionary is not changed.

(Overrides **DictionaryBase(Of TKey, TValue)>..::ContainsKey(TKey).**)

**ConvertAll(Of TOutput)>**

Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.

(Inherited from **CollectionBase(Of T)>**.)

**CopyTo**

Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.

(Inherited from **CollectionBase(Of T)>**.)

**CountWhere**

Counts the number of items in the collection that satisfy the condition defined by predicate.

(Inherited from **CollectionBase(Of T)>**.)

**Equals**

Determines whether the specified **Object** is equal to the current **Object**.

(Inherited from **Object**.)

**Exists**

Determines if the collection contains any item that satisfies the condition defined by predicate.

(Inherited from **CollectionBase(Of T)>**.)

**FindAll**

Enumerates the items in the collection that satisfy the condition defined by predicate.

(Inherited from **CollectionBase(Of T)>**.)

**ForEach**

Performs the specified action on each item in this collection.

(Inherited from **CollectionBase(Of T)>**.)

**GetEnumerator**

Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a KeyValueCollection<TKey, TValue>. The entries are enumerated in the sorted order of the keys.

(Overrides **CollectionBase(Of KeyValueCollection<TKey, TValue>>..::GetEnumerator()().**)

Serves as a hash function for a particular type. **GetHashCode()** is suitable for use in hashing algorithms
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetHashCode</td>
<td>and data structures like a hash table. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetValueElseAdd</td>
<td>Finds a key in the dictionary. If the dictionary already contains a key equal to the passed key, then the existing value is returned via value.</td>
</tr>
<tr>
<td></td>
<td>If the dictionary doesn't contain that key, then value is associated with that key.</td>
</tr>
<tr>
<td>Range</td>
<td>Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are greater than from and less than to are included. The keys are enumerated in sorted order. Keys equal to the end points of the range can be included or excluded depending on the fromInclusive and toInclusive parameters.</td>
</tr>
<tr>
<td>RangeFrom</td>
<td>Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are greater than (and optionally, equal to) from are included. The keys are enumerated in sorted order. Keys equal to from can be included or excluded depending on the fromInclusive parameter.</td>
</tr>
<tr>
<td>RangeTo</td>
<td>Returns a collection that can be used for enumerating some of the keys and values in the collection. Only items that are less than (and optionally, equal to) to are included. The items are enumerated in sorted order. Items equal to to can be included or excluded depending on the toInclusive parameter.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes a key from the dictionary. This method must be overridden in the derived class.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the items in the collection that satisfy the condition defined by predicate. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>RemoveMany</td>
<td>Removes all the keys found in another collection (such as an array or List&lt;TKey&gt;). Each key in keyCollectionToRemove is removed from the dictionary. Keys that are not present are ignored. Changes the value associated with a given key. If the</td>
</tr>
</tbody>
</table>
Replace
dictionary does not contain a key equal to the passed key, then an ArgumentException is thrown.

Reversed
Returns a View collection that can be used for enumerating the keys and values in the collection in reversed order.

ToArray
Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from `CollectionBase<Of <T>>`.)

ToString
Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary. (Inherited from `DictionaryBase<Of <TKey, TValue>>`.)

TrueForAll
Determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from `CollectionBase<Of <T>>`.)

TryGetValue
Determines if this dictionary contains a key equal to key. If so, the value associated with that key is returned through the value parameter. (Overrides `DictionaryBase<Of<TKey, TValue>>::.TryGetValue(TKey, TValue%)`.)
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
# Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection::CopyTo</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>IDictionary::Add</td>
<td>Adds a key-value pair to the collection. If key or value are not of the expected types, an ArgumentException is thrown. If both key and value are of the expected types, the (overridden) Add method is called with the key and value to add. (Inherited from DictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td>IDictionary::Clear</td>
<td>Clears this dictionary. Calls the (overridden) Clear method. (Inherited from DictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td>IDictionary::Contains</td>
<td>Determines if this dictionary contains a key equal to key. The dictionary is not changed. Calls the (overridden) ContainsKey method. If key is not of the correct TKey for the dictionary, false is returned. (Inherited from DictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td>IDictionary::GetEnumerator</td>
<td>Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method. (Inherited from DictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;.)</td>
</tr>
</tbody>
</table>
**IDictionary::Remove**

Removes the key (and associated value) from the collection that is equal to the passed in key. If no key in the dictionary is equal to the passed key, the dictionary is unchanged. Calls the (overridden) Remove method. If key is not of the correct TKey for the dictionary, the dictionary is unchanged.

(Inherited from DictionaryBase(Of TKey, TValue>).)

**IEnumerable::GetEnumerator**

Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method.

(Inherited from DictionaryBase(Of TKey, TValue>).)

**ICloneable::Clone**

Implements ICloneable.Clone. Makes a shallow clone of this dictionary; i.e., if keys or values are reference types, then they are not cloned.
See Also

**OrderedDictionary**(Of **TKey, TValue**)>
**Wintellect.PowerCollections** Namespace

Send [comments](https://example.com) about this topic to Microsoft.
OrderedDictionary<Of <TKey, TValue>>...::Add Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add(KeyValuePair(Of TKey, TValue)&gt;)</td>
<td>Adds a key-value pair to the collection. This implementation calls the Add method with the Key and Value from the item. (Inherited from DictionaryBase(Of TKey, TValue)).</td>
</tr>
<tr>
<td>Add(TKey, TValue)</td>
<td>Adds a new key and value to the dictionary. If the dictionary already contains a key equal to the passed key, then an ArgumentException is thrown (Overrides DictionaryBase(Of TKey, TValue)::.Add(TKey, TValue)).</td>
</tr>
</tbody>
</table>
See Also

OrderedDictionary(Of TKey, TValue) Class
OrderedDictionary(Of TKey, TValue) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<Of <TKey, TValue>>::Add Method (TKey, TValue)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a new key and value to the dictionary. If the dictionary already contains a key equal to the passed key, then an ArgumentException is thrown

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public override sealed void Add(
    TKey key,
    TValue value
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Sub Add ( _
    key As TKey, _
    value As TValue _
)
```

**Visual C++**

```cpp
public:
virtual void Add (  
    TKey key,  
    TValue value  
) override sealed
```

**Parameters**

**key**  
*TKey*  
The new key. "null" is a valid key value.

**value**  
*TValue*  
The new value to associated with that key.
Remarks

Equality between keys is determined by the comparison instance or delegate used to create the dictionary.

Adding an key and value takes time $O(\log N)$, where $N$ is the number of keys in the dictionary.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>key is already present in the dictionary</td>
</tr>
</tbody>
</table>
See Also

OrderedDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary(Of<TKey, TValue>).AddMany Method

See Also

Visual Basic (Declaration) → Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds multiple key-value pairs to a dictionary. If a key exists in both the current instance and dictionaryToAdd, then the value is updated with the value from keysAndValues> (no exception is thrown). Since IDictionary<TKey,TValue> inherits from IEnumerable< KeyValuePair<TKey, TValue>>, this method can be used to merge one dictionary into another.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public void AddMany(
    IEnumerable<KeyValuePair<TKey, TValue>> keysAndValues
)
```

#### Visual Basic (Declaration)

```vbnet
Public Sub AddMany (_
    keysAndValues As IEnumerable(Of KeyValuePair(Of TKey, TValue))
)
```

#### Visual C++

```csharp
public:
void AddMany (
    IEnumerable<KeyValuePair<TKey, TValue>>^ keysAndValues
)
```

### Parameters

**keysAndValues**

`IEnumerable<KeyValuePair<TKey, TValue>>`  
A collection of keys and values whose contents are added to the current dictionary.
Remarks

AddMany takes time $O(M \log (N+M))$, where $M$ is the size of keysAndValues>, and $N$ is the size of this dictionary.
See Also

OrderedDictionary(Of<TKey, TValue>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<(Of <TKey, TValue>)>::Clear Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all keys and values from the dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public override sealed void Clear()
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Sub Clear
```

**Visual C++**

```cpp
public:
    virtual void Clear () override sealed
```
Remarks

Clearing the dictionary takes a constant amount of time, regardless of the number of keys in it.
See Also

OrderedDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<(Of <TKey, TValue>)>::Clone Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a shallow clone of this dictionary; i.e., if keys or values of the dictionary are reference types, then they are not cloned. If TKey or TValue is a value type, then each element is copied as if by simple assignment.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedDictionary<TKey, TValue> Clone()

Visual Basic (Declaration)

Public Function Clone As OrderedDictionary(Of TKey, TValue)

Visual C++

public:
OrderedDictionary<TKey, TValue>^ Clone ()

Return Value

The cloned dictionary.
Remarks

Cloning the dictionary takes time $O(N)$, where $N$ is the number of keys in the dictionary.
See Also

OrderedDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<(Of <TKey, TValue>)>::CloneContents Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a deep clone of this dictionary. A new dictionary is created with a clone of each entry of this dictionary, by calling ICloneable.Clone on each element. If TKey or TValue is a value type, then each element is copied as if by simple assignment.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedDictionary<TKey, TValue> CloneContents()

Visual Basic (Declaration)

Public Function CloneContents As OrderedDictionary(Of TKey, TValue)

Visual C++

public:
OrderedDictionary&lt;TKey, TValue&gt;^ CloneContents ()

Return Value

The cloned dictionary.
Remarks

If TKey or TValue is a reference type, it must implement ICloneable. Otherwise, an InvalidOperationException is thrown.

Cloning the dictionary takes time $O(N \log N)$, where $N$ is the number of keys in the dictionary.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>TKey or TValue is a reference type that does not implement ICloneable.</td>
</tr>
</tbody>
</table>
See Also

OrderedDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this dictionary contains a key equal to key. The dictionary is not changed.

**Namespace:** Wintellect.PowerCollections  
**Assembly:**  PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed bool ContainsKey(TKey key)

Visual Basic (Declaration)

Public Overrides NotOverridable Function ContainsKey ( _
    key As TKey _
) As Boolean

Visual C++

public:
virtual bool ContainsKey ( TKey key
) override sealed

Parameters

key
   TKey
   The key to search for.

Return Value

True if the dictionary contains key. False if the dictionary does not contain key.
Remarks

Searching the dictionary for a key takes time $O(\log N)$, where $N$ is the number of keys in the dictionary.
See Also

**OrderedDictionary(Of TKey, TValue) Class**

**Wintellect.PowerCollections Namespace**

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a KeyValuePair<TKey,TValue>. The entries are enumerated in the sorted order of the keys.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed IEnumerator< KeyValuePair<TKey, TValue>> GetEnumerator

Visual Basic (Declaration)

Public Overrides NotOverridable Function GetEnumerator As IEnumerable

Visual C++

public:
virtual IEnumerable< KeyValuePair<TKey, TValue>>^ GetEnumerator ()

Return Value

An enumerator for enumerating all the elements in the OrderedDictionary.
Remarks

Typically, this method is not called directly. Instead the "foreach" statement is used to enumerate the elements of the dictionary, which uses this method implicitly.

If an element is added to or deleted from the dictionary while it is being enumerated, then the enumeration will end with an InvalidOperationException.

Enumeration all the entries in the dictionary takes time O(N log N), where N is the number of entries in the dictionary.
See Also

`OrderedDictionary(Of TKey, TValue)` Class
`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<Of <TKey, TValue>>::.GetValueElseAdd Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds a key in the dictionary. If the dictionary already contains a key equal to the passed key, then the existing value is returned via value. If the dictionary doesn't contain that key, then value is associated with that key.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public bool GetValueElseAdd(
    TKey key,
    ref TValue value
)
```

Visual Basic (Declaration)

```vbnet
Public Function GetValueElseAdd ( _
    key As TKey, _
    ByRef value As TValue _) As Boolean
```  

Visual C++

```cpp
public:
    bool GetValueElseAdd ( 
        TKey key, 
        TValue% value
    )
```

Parameters

key
   TKey
   The new key.

value
   TValue%
   The new value to associated with that key, if the key isn't present. If the key was present, returns the exist value associated with that key.

Return Value

True if key was already present, false if key wasn't present (and a new value was
added).
Remarks

between keys is determined by the comparison instance or delegate used to create the dictionary.

This method takes time $O(\log N)$, where $N$ is the number of keys in the dictionary. If a value is added, it is more efficient than calling TryGetValue followed by Add, because the dictionary is not searched twice.
See Also

OrderedDictionary(Of TKey, TValue) Class  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary(Of TKey, TValue)\:::Range Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are greater than from and less than to are included. The keys are enumerated in sorted order. Keys equal to the end points of the range can be included or excluded depending on the fromInclusive and toInclusive parameters.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public OrderedDictionary<TKey, TValue>.OrderedDictionary(Of TKey,
    TKey from,
    bool fromInclusive,
    TKey to,
    bool toInclusive
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function Range ( _
    from As TKey, _
    fromInclusive As Boolean, _
    to As TKey, _
    toInclusive As Boolean _
) As OrderedDictionary(Of TKey, TValue)>..::View
```

**Visual C++**

```cpp
public:
    OrderedDictionary<Of<TKey, TValue>>..::View^ Range (  
    TKey from,
    bool fromInclusive,
    TKey to,
    bool toInclusive
)
```

**Parameters**

`from`  
`TKey`  
The lower bound of the range.

`fromInclusive`  
`Boolean`  
If true, the lower bound is inclusive--keys equal to the lower bound will be included in the range. If false, the lower bound is exclusive--keys equal to
the lower bound will not be included in the range.

to

TKey
The upper bound of the range.

toInclusive

Boolean
If true, the upper bound is inclusive--keys equal to the upper bound will be included in the range. If false, the upper bound is exclusive--keys equal to the upper bound will not be included in the range.

**Return Value**

An OrderedDictionary.View of key-value pairs in the given range.
Remarks

If from is greater than or equal to to, the returned collection is empty.

The sorted order of the keys is determined by the comparison instance or delegate used to create the dictionary.

Typically, this property is used in conjunction with a foreach statement. For example:

```csharp
foreach(KeyValuePair<TKey, TValue> pair in dictionary.
    // process pair
}
```

Calling Range does not copy the data in the dictionary, and the operation takes constant time.
See Also

OrderedDictionary<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<\(\text{Of}\ <\text{TKey, TValue}>\)>\...::\RangeFrom\ Method


See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are greater than (and optionally, equal to) from are included. The keys are enumerated in sorted order. Keys equal to from can be included or excluded depending on the fromInclusive parameter.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public OrderedDictionary<TKey, TValue>.OrderedDictionary<(Of <TKey,
    TKey from,
    bool fromInclusive
)
```

Visual Basic (Declaration)

```vbnet
Public Function RangeFrom ( _
    from As TKey, _
    fromInclusive As Boolean _
) As OrderedDictionary<(Of <TKey, TValue>)>.::View
```

Visual C++

```cpp
public: 
OrderedDictionary<(Of <TKey, TValue>)>.::View^ RangeFrom ( 
    TKey from,
    bool fromInclusive
)
```

Parameters

from
    TKey
    The lower bound of the range.

fromInclusive
    Boolean
    If true, the lower bound is inclusive--keys equal to the lower bound will be included in the range. If false, the lower bound is exclusive--keys equal to the lower bound will not be included in the range.

Return Value
An OrderedDictionary.View of key-value pairs in the given range.
Remarks

The sorted order of the keys is determined by the comparison instance or delegate used to create the dictionary.

Typically, this property is used in conjunction with a foreach statement. For example:

```csharp
foreach(KeyValuePair<TKey, TValue> pair in dictionary)
{
    // process pair
}
```

Calling RangeFrom does not copy of the data in the dictionary, and the operation takes constant time.
See Also

OrderedDictionary<Of <TKey, TValue>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Returns a collection that can be used for enumerating some of the keys and values in the collection. Only items that are less than (and optionally, equal to) to are included. The items are enumerated in sorted order. Items equal to to can be included or excluded depending on the toInclusive parameter.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public OrderedDictionary<TKey, TValue>.OrderedDictionary(Of TKey, TValue>
    TKey to,
    bool toInclusive
)
```

Visual Basic (Declaration)

```vbnet
Public Function RangeTo ( _
    to As TKey, _
    toInclusive As Boolean _
) As OrderedDictionary(Of TKey, TValue>)..::View
```

Visual C++

```cpp
public:
OrderedDictionary(Of TKey, TValue>)..::View^ RangeTo ( 
    TKey to,
    bool toInclusive
)
```

Parameters

to

TKey
The upper bound of the range.

toInclusive

Boolean
If true, the upper bound is inclusive--keys equal to the upper bound will be included in the range. If false, the upper bound is exclusive--keys equal to the upper bound will not be included in the range.

Return Value
An OrderedDictionary.View of key-value pairs in the given range.
Remarks

The sorted order of the keys is determined by the comparison instance or delegate used to create the dictionary.

Typically, this property is used in conjunction with a foreach statement. For example:

```csharp
foreach(KeyValuePair<TKey, TValue> pair in dictionary.{
    // process pair
}
```

Calling RangeTo does not copy the data in the dictionary, and the operation takes constant time.
See Also

OrderedDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary< Of <TKey, TValue>>::RemoveMany Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all the keys found in another collection (such as an array or List<TKey>). Each key in keyCollectionToRemove is removed from the dictionary. Keys that are not present are ignored.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public int RemoveMany(
    IEnumerable<TKey> keyCollectionToRemove
)
```

### Visual Basic (Declaration)

```vbnet
Public Function RemoveMany ( _
    keyCollectionToRemove As IEnumerable(Of TKey) _
) As Integer
```

### Visual C++

```c++
public:
int RemoveMany ( 
    IEnumerable<TKey>^ keyCollectionToRemove
)
```

## Parameters

**keyCollectionToRemove**

**IEnumerator<TKey>**

A collection of keys to remove from the dictionary.

## Return Value

The number of keys removed from the dictionary.
Remarks

RemoveMany takes time $O(M \log N)$, where $M$ is the size of keyCollectionToRemove, and $N$ is this size of this collection.
See Also

**OrderedDictionary(Of TKey, TValue)>** Class
**Wintellect.PowerCollections** Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<Of <TKey, TValue>>...::Replace Method

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Changes the value associated with a given key. If the dictionary does not contain a key equal to the passed key, then an ArgumentException is thrown.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public void Replace(
    TKey key,
    TValue value
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub Replace (_
    key As TKey, _
    value As TValue _
)
```

**Visual C++**

```cpp
public:
void Replace ( _
    TKey key, _
    TValue value
)
```

### Parameters

**key**

- **Key** `TKey`
  - The new key.

**value**

- **Value** `TValue`
  - The new value to associate with that key.
Remarks

Unlike adding or removing an element, changing the value associated with a key can be performed while an enumeration (foreach) on the dictionary is in progress.

Equality between keys is determined by the comparison instance or delegate used to create the dictionary.

Replace takes time $O(\log N)$, where $N$ is the number of entries in the dictionary.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.Collections.Generic.KeyNotFoundException</code></td>
<td>key is not present in the dictionary</td>
</tr>
</tbody>
</table>
See Also

OrderedDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<Of<TKey, TValue>>::Reversed Method

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a View collection that can be used for enumerating the keys and values in the collection in reversed order.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedDictionary<TKey, TValue>.OrderedDictionary(Of<TKey, TValue>.

Visual Basic (Declaration)

Public Function Reversed As OrderedDictionary(Of TKey, TValue)....

Visual C++

public:
OrderedDictionary(Of TKey, TValue)::.::View^ Reversed ()

Return Value

An OrderedDictionary.View of key-value pairs in reverse order.
Remarks

Typically, this method is used in conjunction with a foreach statement. For example:

```csharp
foreach(KeyValuePair<TKey, TValue> pair in dictionary.Reversed())
{
    // process pair
}
```

If an entry is added to or deleted from the dictionary while the View is being enumerated, then the enumeration will end with an InvalidOperationException.

Calling Reverse does not copy the data in the dictionary, and the operation takes constant time.
See Also

**OrderedDictionary<(Of <TKey, TValue>)>** Class  
**Wintellect.PowersCollections** Namespace

Send [comments](#) about this topic to Microsoft.
OrderedDictionary(Of TKey, TValue)>...::System.ICloneable.Clone Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Implements ICloneable.Clone. Makes a shallow clone of this dictionary; i.e., if keys or values are reference types, then they are not cloned.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private Object ICloneable.Clone()

Visual Basic (Declaration)

Private Function System.ICloneable.Clone As Object Implements IClone

Visual C++

private:
virtual Object^ System.ICloneable.Clone () sealed = ICloneable::Clon

Return Value

The cloned dictionary.
See Also

OrderedDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary(Of TKey, TValue).TryGetValue

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this dictionary contains a key equal to key. If so, the value associated with that key is returned through the value parameter.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public override sealed bool TryGetValue(
    TKey key,
    out TValue value
)
```

Visual Basic (Declaration)

```vbnet
Public Overrides NotOverridable Function TryGetValue ( _
    key As TKey, _
    <OutAttribute> ByRef value As TValue _
) As Boolean
```

Visual C++

```cpp
public:
virtual bool TryGetValue ( 
    TKey key,
    [OutAttribute] TValue% value
) override sealed
```

Parameters

key

- **TKey**
  - The key to search for.

value

- **TValue%**
  - Returns the value associated with key, if true was returned.

Return Value

True if the dictionary contains key. False if the dictionary does not contain key.
Remarks

TryGetValue takes time $O(\log N)$, where $N$ is the number of entries in the dictionary.
See Also

OrderedDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary(Of TKey, TValue) Properties

See Also

☐ Include Inherited Members ☑ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class OrderedDictionary<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class OrderedDictionary(Of TKey, TValue)

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class OrderedDictionary
Type Parameters

TKey
TValue

The type exposes the following properties.
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comparer</strong></td>
<td>Returns the IComparer&lt;T&gt; used to compare keys in this dictionary.</td>
</tr>
<tr>
<td><strong>Count</strong></td>
<td>Returns the number of keys in the dictionary.</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Gets or sets the value associated with a given key. When getting a value, if this key is not found in the collection, then an ArgumentException is thrown. When setting a value, the value replaces any existing value in the dictionary.</td>
</tr>
<tr>
<td><strong>Keys</strong></td>
<td>Returns a collection of the keys in this dictionary.</td>
</tr>
<tr>
<td><strong>Values</strong></td>
<td>Returns a collection of the values in this dictionary. The ordering of values in this collection is the same as that in the Keys collection.</td>
</tr>
</tbody>
</table>

(Overrides `CollectionBase<Of <KeyValuePair<Of <TKey, TValue>>>>.`)

(Overrides `DictionaryBase<Of <TKey, TValue>>.`)
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;(Of &lt;T&gt;)&gt;::IsReadOnly</code></td>
<td>(Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>) Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td><code>ICollection&lt;&gt;::IsSynchronized</code></td>
<td>(Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>) Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td><code>ICollection&lt;&gt;::SyncRoot</code></td>
<td>(Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>) Returns whether this dictionary is fixed size. This implemented always returns false.</td>
</tr>
<tr>
<td><code>IDictionary&lt;&gt;::IsFixedSize</code></td>
<td>(Inherited from <code>DictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;</code>) Returns if this dictionary is read-only. This implementation always returns false.</td>
</tr>
<tr>
<td><code>IDictionary&lt;&gt;::IsReadOnly</code></td>
<td>(Inherited from <code>DictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;</code>) Gets or sets the value associated with a given key. When getting a value, if this key is not found in the collection, then null is returned. When setting a value, the value replaces any existing value in the dictionary. If either the key or value are not of the correct type for this dictionary, an ArgumentException is thrown.</td>
</tr>
<tr>
<td><code>IDictionary&lt;&gt;::Item</code></td>
<td>(Inherited from <code>DictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;</code>) Returns a collection of all the keys in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method.</td>
</tr>
<tr>
<td><code>IDictionary&lt;&gt;::Keys</code></td>
<td>(Inherited from <code>DictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;</code>)</td>
</tr>
</tbody>
</table>
Returns a collection of all the values in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method. (Inherited from `DictionaryBase<Of <TKey, TValue>>`.)
See Also

OrderedDictionary(Of TKey, TValue)>  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<(Of <TKey, TValue>)>.Comparer Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the IComparer<T> used to compare keys in this dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public IComparer<TKey> Comparer{ get; }

Visual Basic (Declaration)

Public ReadOnly Property Comparer As IComparer(Of TKey)

Visual C++

public:
    property IComparer<TKey>^ Comparer { 
        IComparer<TKey>^ get ();
    }

Field Value

If the dictionary was created using a comparer, that comparer is returned. If the dictionary was created using a comparison delegate, then a comparer equivalent to that delegate is returned. Otherwise the default comparer for TKey (Comparer<TKey>.Default) is returned.
See Also

**OrderedDictionary(Of TKey, TValue) Class**

**Wintellect.PowerCollections Namespace**

Send comments about this topic to Microsoft.
OrderedDictionary<(Of <TKey, TValue>)>.::Count Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the number of keys in the dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed int Count { get; }

Visual Basic (Declaration)

Public Overrides NotOverridable ReadOnly Property Count As Integer

Visual C++

public:
    virtual property int Count {
        int get () override sealed;
    }

Field Value

The number of keys in the dictionary.
Remarks

The size of the dictionary is returned in constant time..
See Also

OrderedDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<(Of <TKey, TValue>)>::Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets or sets the value associated with a given key. When getting a value, if this key is not found in the collection, then an ArgumentException is thrown. When setting a value, the value replaces any existing value in the dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public override sealed TValue this[
    TKey key
] { get; set; }
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Default Property Item ( _
    key As TKey _
) As TValue
```

**Visual C++**

```cpp
public:
virtual property TValue default[TKey key] { 
    TValue get (TKey key) override sealed;
    void set (TKey key, TValue value) override sealed;
}
```

**Parameters**

key
   TKey

**Field Value**

The value associated with the key
Remarks

The indexer takes time $O(\log N)$, where $N$ is the number of entries in the dictionary.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentException</code></td>
<td>A value is being retrieved, and the key is not present in the dictionary.</td>
</tr>
<tr>
<td><code>System..::ArgumentNullException</code></td>
<td>key is null.</td>
</tr>
</tbody>
</table>
See Also

OrderedDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<
(Or <TKey, TValue>)>...::View Class

See Also  Members

- Visual Basic (Declaration)  - Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

The OrderedDictionary<
(TKey,TValue).View class is used to look at a subset of the keys and values inside an ordered dictionary. It is returned from the Range, RangeTo, RangeFrom, and Reversed methods.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

[SerializableAttribute]
public class View : DictionaryBase<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class View _
    Inherits DictionaryBase(Of TKey, TValue)

Visual C++

[SerializableAttribute]
public ref class View : public DictionaryBase<TKey, TValue>
Remarks

Views are dynamic. If the underlying dictionary changes, the view changes in sync. If a change is made to the view, the underlying dictionary changes accordingly.

Typically, this class is used in conjunction with a foreach statement to enumerate the keys and values in a subset of the OrderedDictionary. For example:

```csharp
foreach(KeyValuePair<TKey, TValue> pair in dictionary) {
    // process pair
}
```
Inheritance Hierarchy

System::Object

Wintellect.PowerCollections::CollectionBase<<KeyValuePair<<KeyValuePair<TKey,TValue>>>>

Wintellect.PowerCollections::DictionaryBase<<KeyValuePair<TKey,TValue>>>

Wintellect.PowerCollections::OrderedDictionary<<KeyValuePair<TKey,TValue>>>...::View
See Also

OrderedDictionary<(Of <TKey, TValue>)>View Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<(Of <TKey, TValue>)>...::View Members
See Also  Methods  Properties  Explicit Interface Implementations

 ![ ]

 - Include Inherited Members  - Include Protected Members
 - .NET Compact Framework Members Only
 - XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class View

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class View

Visual C++

[SerializableAttribute]
public ref class View

OrderedDictionary<TKey, TValue> is a collection that maps keys of type TKey to values of type TValue. The keys are maintained in a sorted order, and at most one value is permitted for each key.

The OrderedDictionary(Of TKey, TValue>) generic type exposes the following members.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✷ Add</td>
<td>Overloaded.</td>
</tr>
<tr>
<td></td>
<td>Provides a read-only view of this dictionary. The returned IDictionary&lt;TKey,TValue&gt; provides a view of the dictionary that prevents modifications to the dictionary. Use the method to provide access to the dictionary without allowing changes. Since the returned object is just a view, changes to the dictionary will be reflected in the view. (Inherited from DictionaryBase&lt;TOf&lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td>✷ AsReadOnly</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Provides a read-only view of this dictionary. Use the method to provide access to the dictionary without allowing changes. Since the returned object is just a view, changes to the dictionary will be reflected in the view. (Inherited from DictionaryBase&lt;TOf&lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td>✷ Clear</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Removes all the keys and values within this view from the underlying OrderedDictionary. (Overrides DictionaryBase&lt;TOf&lt;TKey, TValue&gt;&gt;::Clear().)</td>
</tr>
<tr>
<td>✷ Contains</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Determines if a dictionary contains a given KeyValuePair. This implementation checks to see if the dictionary contains the given key, and if the value associated with the key is equal to (via object.Equals) the value. (Inherited from DictionaryBase&lt;TOf&lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td>✷ ContainsKey</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tests if the key is present in the part of the dictionary being viewed. (Overrides DictionaryBase&lt;TOf&lt;TKey, TValue&gt;&gt;::ContainsKey(TKey).)</td>
</tr>
<tr>
<td>✷ ConvertAll&lt;Of&lt;TOutput&gt;&gt;</td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from CollectionBase&lt;OF&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>✷ CopyTo</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;OF&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>✷ Counts</td>
<td></td>
</tr>
</tbody>
</table>
|               | Counts the number of items in the collection that satisfy the
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Inheritance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CountWhere</strong></td>
<td>Condition defined by predicate. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>).</td>
<td></td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
<td></td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>).</td>
<td></td>
</tr>
<tr>
<td><strong>FindAll</strong></td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>).</td>
<td></td>
</tr>
<tr>
<td><strong>ForEach</strong></td>
<td>Performs the specified action on each item in this collection. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>).</td>
<td></td>
</tr>
<tr>
<td><strong>GetEnumerator</strong></td>
<td>Enumerate all the keys and values in this view. (Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.).</td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>).</td>
<td><code>GetHashCode()</code> is suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>).</td>
<td></td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Removes a key from the dictionary. This method must be overridden in the derived class.</td>
<td>Removes all the items in the collection that satisfy the condition defined by predicate. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>).</td>
</tr>
<tr>
<td><strong>RemoveAll</strong></td>
<td>Removes all the items in the collection that satisfy the condition defined by predicate. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>).</td>
<td></td>
</tr>
<tr>
<td><strong>Reversed</strong></td>
<td>Creates a new View that has the same keys and values as this, in the reversed order.</td>
<td></td>
</tr>
<tr>
<td><strong>ToArray</strong></td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>).</td>
<td></td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary. (Inherited from <code>DictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;</code>).</td>
<td></td>
</tr>
</tbody>
</table>
**TrueForAll**
Determines if all of the items in the collection satisfy the condition defined by predicate.
(Inherited from `CollectionBase<Of <T>>`.)

**TryGetValue**
Determines if this view contains a key equal to key. If so, the value associated with that key is returned through the value parameter.
(Overrides `DictionaryBase<Of <TKey, TValue>>`: `TryGetValue(TKey, TValue%)`.)
### Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td>Number of keys in this view. (Overrides <strong>CollectionBase</strong>&lt;(Of <strong>KeyValuePair</strong>&lt;Of &lt;TKey, TValue&gt;&gt;)&gt;&gt;&gt;::<strong>Count</strong>.) Gets or sets the value associated with a given key. When getting a value, if this key is not found in the collection, then an ArgumentException is thrown. When setting a value, the value replaces any existing value in the dictionary. When setting a value, the key must be within the range of keys being viewed. (Overrides <strong>DictionaryBase</strong>&lt;(Of &lt;TKey, TValue&gt;&gt;)&gt;::<strong>Item</strong>[[[TKey]]].)</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Returns a collection of the keys in this dictionary. (Inherited from <strong>DictionaryBase</strong>&lt;(Of &lt;TKey, TValue&gt;)&gt;.) Returns a collection of the values in this dictionary. The ordering of values in this collection is the same as that in the Keys collection. (Inherited from <strong>DictionaryBase</strong>&lt;(Of &lt;TKey, TValue&gt;)&gt;.)</td>
</tr>
<tr>
<td><strong>Keys</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Values</strong></td>
<td></td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;(Of &lt;T&gt;)&gt;::IsReadOnly</code></td>
<td>(Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>) Copied all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.</td>
</tr>
<tr>
<td><code>ICollection::CopyTo</code></td>
<td>Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td><code>ICollection::IsSynchronized</code></td>
<td>Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td><code>ICollection::SyncRoot</code></td>
<td>Adds a key-value pair to the collection. If key or value are not of the expected types, an ArgumentException is thrown. If both key and value are of the expected types, the (overridden) Add method is called with the key and value to add.</td>
</tr>
<tr>
<td><code>IDictionary::Add</code></td>
<td>Clears this dictionary. Calls the (overridden) Clear method.</td>
</tr>
<tr>
<td><code>IDictionary::Clear</code></td>
<td>Determines if this dictionary contains a key equal to key. The dictionary is not changed. Calls the (overridden) ContainsKey method.</td>
</tr>
</tbody>
</table>
| `IDictionary::Contains` | If key is not of the correct TKey for the
dictionary,
false is returned.
(Inherited from DictionaryBase(Of TKey, TValue>.)

Returns an enumerator that enumerates all
the entries in the dictionary. Each entry is
returned as a DictionaryEntry. The entries
are enumerated in the same orders as the
(overridden) GetEnumerator method.
(Inherited from DictionaryBase(Of TKey, TValue>.)

Returns whether this dictionary is fixed
size. This implemented always returns false.
(Inherited from DictionaryBase(Of TKey, TValue>.)

Returns if this dictionary is read-only. This
implementation always returns false.
(Inherited from DictionaryBase(Of TKey, TValue>.)

Gets or sets the value associated with a
given key. When getting a value, if this key
is not found in the collection, then null is
returned. When setting a value, the value
replaces any existing value in the
dictionary. If either the key or value are not
of the correct type for this dictionary, an
ArgumentException is thrown.
(Inherited from DictionaryBase(Of TKey, TValue>.)

Returns a collection of all the keys in the
dictionary. The values in this collection will
be enumerated in the same order as the
(overridden) GetEnumerator method.
(Inherited from DictionaryBase(Of TKey, TValue>.)

Removes the key (and associated value)
from the collection that is equal to the
passed in key. If no key in the dictionary is
equal to the passed key, the dictionary is
**IDictionary::Remove**

Unchanged. Calls the (overridden) Remove method. If key is not of the correct TKey for the dictionary, the dictionary is unchanged.

(Inherited from `DictionaryBase<(of <TKey, TValue>)>`) 

**IDictionary::Values**

Returns a collection of all the values in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method.

(Inherited from `DictionaryBase<of <TKey, TValue)>`.)

**IEnumerable::GetEnumerator**

Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method.

(Inherited from `DictionaryBase<of <TKey, TValue>)`.)

---

```
IDictionary::Remove

IDictionary::Values

IEnumerable::GetEnumerator
```
See Also

OrderedDictionary(Of TKey, TValue) View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<(Of<TKey, TValue>)>...::View Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class View

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class View

Visual C++

[SerializableAttribute]
public ref class View

The `OrderedDictionary(Of TKey, TValue)` generic type exposes the following methods.
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Overloaded. Provides a read-only view of this dictionary. The returned IDictionary&lt;TKey,TValue&gt; provides a view of the dictionary that prevents modifications to the dictionary. Use the method to provide access to the dictionary without allowing changes. Since the returned object is just a view, changes to the dictionary will be reflected in the view. (Inherited from DictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this dictionary. The returned IDictionary&lt;TKey,TValue&gt; provides a view of the dictionary that prevents modifications to the dictionary. Use the method to provide access to the dictionary without allowing changes. Since the returned object is just a view, changes to the dictionary will be reflected in the view. (Inherited from DictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all the keys and values within this view from the underlying OrderedDictionary. (Overides DictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;::Clear().)</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Determines if a dictionary contains a given KeyValuePair. This implementation checks to see if the dictionary contains the given key, and if the value associated with the key is equal to (via object.Equals) the value. (Inherited from DictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>ContainsKey</strong></td>
<td>Tests if the key is present in the part of the dictionary being viewed. (Overides DictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;::ContainsKey(TKey).)</td>
</tr>
<tr>
<td><strong>ConvertAll&lt;Of&lt;TOutput&gt;&gt;</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
</tbody>
</table>
| **Count**    | Counts the number of items in the collection that satisfy the condition. (Inherited from ICollection<T>.)**
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Inheritance</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountWhere</td>
<td>condition defined by predicate.</td>
<td>(Inherited from CollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Exists</td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate.</td>
<td>(Inherited from CollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>FindAll</td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate.</td>
<td>(Inherited from CollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>ForEach</td>
<td>Performs the specified action on each item in this collection.</td>
<td>(Inherited from CollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>GetEnumerator</td>
<td>Enumerate all the keys and values in this view.</td>
<td>(Inherited from CollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>GetEnumerator</td>
<td>(Overrides CollectionBase(Of <code>&lt;KeyValuePair&lt;TKey, TValue&gt;</code>)).</td>
<td></td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type.</td>
<td></td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes a key from the dictionary. This method must be overridden in the derived class.</td>
<td></td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the items in the collection that satisfy the condition defined by predicate.</td>
<td>(Inherited from CollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>Reversed</td>
<td>Creates a new View that has the same keys and values as this, in the reversed order.</td>
<td></td>
</tr>
<tr>
<td>ToArray</td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.</td>
<td>(Inherited from CollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>ToString</td>
<td>Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.</td>
<td>(Inherited from DictionaryBase(Of <code>&lt;TKey, TValue&gt;</code>).)</td>
</tr>
</tbody>
</table>
- **TrueForAll**
  Determines if all of the items in the collection satisfy the condition defined by predicate.
  (Inherited from `CollectionBase<OF T>`.)

- **TryGetValue**
  Determines if this view contains a key equal to key. If so, the value associated with that key is returned through the value parameter.
  (Overrides `DictionaryBase<OF TKey, TValue>`.): TryGetValue(TKey, TValue%).)
# Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🎁 <code>Finalize</code></td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>🎁 <code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
### Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `ICollection::CopyTo` | Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from `CollectionBase<T>`.)
| `IDictionary::Add` | Adds a key-value pair to the collection. If key or value are not of the expected types, an ArgumentException is thrown. If both key and value are of the expected types, the (overridden) Add method is called with the key and value to add. (Inherited from `DictionaryBase<TKey, TValue>`.)
| `IDictionary::Clear` | Clears this dictionary. Calls the (overridden) Clear method. (Inherited from `DictionaryBase<TKey, TValue>`.)
| `IDictionary::Contains` | Determines if this dictionary contains a key equal to key. The dictionary is not changed. Calls the (overridden) ContainsKey method. If key is not of the correct TKey for the dictionary, false is returned. (Inherited from `DictionaryBase<TKey, TValue>`.)
| `IDictionary::GetEnumerator` | Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method. (Inherited from `DictionaryBase<TKey, TValue>`.): |

Cellar to top.
**IDictionary::Remove**

Removes the key (and associated value) from the collection that is equal to the passed in key. If no key in the dictionary is equal to the passed key, the dictionary is unchanged. Calls the (overridden) Remove method. If key is not of the correct TKey for the dictionary, the dictionary is unchanged.

(Inherited from `DictionaryBase<Of <TKey, TValue>>`.)

**IEnumerable::GetEnumerator**

Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method.

(Inherited from `DictionaryBase<Of <TKey, TValue>>`.)
See Also

OrderedDictionary(Of TKey, TValue)\View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
View::Add Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add(KeyValuePair(Of&lt;TKey, TValue&gt;))</td>
<td>Adds a key-value pair to the collection. This implementation calls the Add method with the Key and Value from the item. (Inherited from DictionaryBase(Of&lt;TKey, TValue&gt;).)</td>
</tr>
<tr>
<td>Add(TKey, TValue)</td>
<td>Adds a new key-value pair to the dictionary. (Inherited from DictionaryBase(Of&lt;TKey, TValue&gt;).)</td>
</tr>
</tbody>
</table>
See Also

**`OrderedDictionary(Of TKey, TValue)` View Class**

**`OrderedDictionary(Of TKey, TValue)` View Members**

**`Wintellect.PowerCollections` Namespace**

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
OrderedDictionary<Of <TKey, TValue>> View::Clear Method

See Also Example

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all the keys and values within this view from the underlying OrderedDictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public override sealed void Clear()
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Sub Clear
```

**Visual C++**

```c++
public:
virtual void Clear () override sealed
```
Examples

The following removes all the keys that start with "A" from an OrderedDictionary.

```
    dictionary.Range("A", "B").Clear();
```
See Also

**OrderedDictionary(Of TKey, TValue)>**, **::View** Class
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
OrderedDictionary(Of TKey, TValue)>...:ContainsKey Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Tests if the key is present in the part of the dictionary being viewed.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public override sealed bool ContainsKey(TKey key)
```

Visual Basic (Declaration)

```vbnet
Public Overrides NotOverridable Function ContainsKey ( _
    key As TKey _
) As Boolean
```

Visual C++

```cpp
public:
    virtual bool ContainsKey ( 
        TKey key
    ) override sealed
```

Parameters

d Key
    TKey
    Key to check for.

Return Value

True if the key is within this view.
See Also

OrderedDictionary(Of TKey, TValue)> View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Enumerate all the keys and values in this view.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public override sealed IEnumerator<KeyValuePair<TKey, TValue>> GetEnumerable()
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Function GetEnumerator As IEnumerable
```

**Visual C++**

```cpp
public:
virtual IEnumerable<KeyValuePair<TKey, TValue>>^ GetEnumerator() override
```

**Return Value**

An IEnumerator of KeyValuePairs with the keys and views in this view.
See Also

OrderedDictionary<(Of <TKey, TValue>>) View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<Of<TKey, TValue>>::Reversed Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new View that has the same keys and values as this, in the reversed order.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

public `OrderedDictionary<TKey, TValue> OrderedDictionary<Of <TKey, TValue>>`.

**Visual Basic (Declaration)**

Public Function Reversed As `OrderedDictionary(Of <TKey, TValue>>`. 

**Visual C++**

```cpp
public: 
`OrderedDictionary<Of <TKey, TValue>>>.:::View`^ Reversed ()
```

**ReturnValue**

A new View that has the reversed order of this view.
See Also

OrderedDictionary(Of TKey, TValue>), Dictionary, Dictionary, Dictionary::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;::View:::TryGetValue Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this view contains a key equal to key. If so, the value associated with that key is returned through the value parameter.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public override sealed bool TryGetValue(
    TKey key,
    out TValue value
)
```

### Visual Basic (Declaration)

```vbnet
Public Overrides NotOverridable Function TryGetValue ( _
    key As TKey, _
    <OutAttribute> ByRef value As TValue _
) As Boolean
```

### Visual C++

```cpp
public:
    virtual bool TryGetValue ( 
        TKey key,
        [OutAttribute] TValue% value
    ) override sealed
```

## Parameters

- **key**
  - TKey
  - The key to search for.

- **value**
  - TValue%
  - Returns the value associated with key, if true was returned.

## Return Value

True if the key is within this view.
See Also

OrderedDictionary(Of TKey, TValue) View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<
(Of TKey, TValue)>...::View Properties

See Also

☐ Include Inherited Members ☑ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class View

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class View

Visual C++

[SerializableAttribute]
public ref class View

The `OrderedDictionary<(Of <TKey, TValue>)>` generic type exposes the following properties.
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td>(Overrides <code>CollectionBase&lt;(Of </code>&lt;KeyValuePair&lt;(Of <code>&lt;TKey, TValue&gt;</code>)&gt;&gt;)&gt;::Count.)&lt;br&gt;Gets or sets the value associated with a given key. When getting a value, if this key is not found in the collection, then an ArgumentException is thrown. When setting a value, the value replaces any existing value in the dictionary. When setting a value, the key must be within the range of keys being viewed.&lt;br&gt;(Overrides <code>DictionaryBase&lt;(Of </code>&lt;TKey, TValue&gt;<code>)&gt;::.Item[[[TKey]]].)&lt;br&gt;**(Overrides </code>DictionaryBase&lt;(Of <code>&lt;TKey, TValue&gt;</code>)&gt;::.Item[[[TKey]]].)&lt;br&gt;**(Overrides <code>DictionaryBase&lt;(Of </code>&lt;TKey, TValue&gt;`)&gt;::.Item[[[TKey]]].)</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Returns a collection of the keys in this dictionary.&lt;br&gt;(Inherited from <code>DictionaryBase&lt;(Of </code>&lt;TKey, TValue&gt;<code>)&gt;.)&lt;br&gt;**Returns a collection of the values in this dictionary. The ordering of values in this collection is the same as that in the Keys collection.&lt;br&gt;(Inherited from </code>DictionaryBase&lt;(Of <code>&lt;TKey, TValue&gt;</code>)&gt;.)</td>
</tr>
</tbody>
</table>
# Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection</code>&lt;(<code>Of</code> <code>T</code>)&gt;::<code>IsReadOnly</code></td>
<td>Indicates whether the collection is synchronized. (Inherited from <code>CollectionBase&lt;(</code>Of<code> </code>T<code>)&gt;</code>. )</td>
</tr>
<tr>
<td><code>ICollection</code>::<code>IsSynchronized</code></td>
<td>Indicates the synchronization object for this collection. (Inherited from <code>CollectionBase&lt;(</code>Of<code> </code>T<code>)&gt;</code>. )</td>
</tr>
<tr>
<td><code>ICollection</code>::<code>SyncRoot</code></td>
<td>Returns whether this dictionary is fixed size. This implemented always returns false. (Inherited from <code>DictionaryBase&lt;(</code>Of<code> TKey, TValue&gt;)</code>. )</td>
</tr>
<tr>
<td><code>IDictionary</code>::<code>IsFixedSize</code></td>
<td>Returns if this dictionary is read-only. This implementation always returns false. (Inherited from <code>DictionaryBase&lt;(</code>Of<code> TKey, TValue&gt;)</code>. )</td>
</tr>
<tr>
<td><code>IDictionary</code>::<code>IsReadOnly</code></td>
<td>Gets or sets the value associated with a given key. When getting a value, if this key is not found in the collection, null is returned. When setting a value, the value replaces any existing value in the dictionary. If either the key or value are not of the correct type for this dictionary, an ArgumentException is thrown. (Inherited from <code>DictionaryBase&lt;(</code>Of<code> TKey, TValue&gt;)</code>. )</td>
</tr>
<tr>
<td><code>IDictionary</code>::<code>Item</code></td>
<td>Returns a collection of all the keys in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method. (Inherited from <code>DictionaryBase&lt;(</code>Of<code> TKey, TValue&gt;)</code>. )</td>
</tr>
<tr>
<td><code>IDictionary</code>::<code>Keys</code></td>
<td></td>
</tr>
</tbody>
</table>
Returns a collection of all the values in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method. (Inherited from DictionaryBase<Of <TKey, TValue>>.)
See Also

OrderedDictionary(Of TKey, TValue)>\n\n::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedDictionary<(Of <TKey, TValue>)>...::View...::Count Property

See Also

- Visual Basic (Declaration) - Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Number of keys in this view.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed int Count{ get; }

Visual Basic (Declaration)

Public Overrides NotOverridable ReadOnly Property Count As Integer

Visual C++

public:
virtual property int Count {
    int get () override sealed;
}

Field Value

Number of keys that lie within the bounds the view.
See Also

- `OrderedDictionary<Of <TKey, TValue>>`.::View Class
- `Wintellect.PowerCollections` Namespace

Send [comments](mailto:support@microsoft.com) about this topic to Microsoft.
OrderedDictionary(Of<TKey, TValue>)::View::Item Property

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets or sets the value associated with a given key. When getting a value, if this key is not found in the collection, then an ArgumentException is thrown. When setting a value, the value replaces any existing value in the dictionary. When setting a value, the key must be within the range of keys being viewed.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed TValue this[
    TKey key
]{ get; set;}

Visual Basic (Declaration)

Public Overrides NotOverridable Default Property Item ( _
    key As TKey _
) As TValue

Visual C++

public:
virtual property TValue default[TKey key] { 
    TValue get (TKey key) override sealed;
    void set (TKey key, TValue value) override sealed;
}

Parameters

key
    TKey

Field Value

The value associated with the key.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::ArgumentException</code></td>
<td>A value is being retrieved, and the key is not present in the dictionary, or a value is being set, and the key is outside the range of keys being viewed by this View.</td>
</tr>
</tbody>
</table>
See Also

OrderedDictionary(Of TKey, TValue)>..::.View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
The OrderedMultiDictionary class that associates values with a key. Unlike an OrderedDictionary, each key can have multiple values associated with it. When indexing an OrderedMultidictionary, instead of a single value associated with a key, you retrieve an enumeration of values.

All of the key are stored in sorted order. Also, the values associated with a given key are kept in sorted order as well.

When constructed, you can chose to allow the same value to be associated with a key multiple times, or only one time.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
-Syntax

C#

[SerializableAttribute]
public class OrderedMultiDictionary<TKey, TValue> : MultiDictionary<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute>
Public Class OrderedMultiDictionary(Of TKey, TValue) _
    Inherits MultiDictionaryBase(Of TKey, TValue) _
    Implements ICloneable

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class OrderedMultiDictionary : public MultiDictionaryBase<TKey, TValue>
    ICloneable
Type Parameters

TKey
   The type of the keys.
TValue
   The type of values associated with the keys.
Inheritance Hierarchy

System::Object
Wintellect.PowerCollections::CollectionBase(Of KeyValuePair(Of TKey, ICollection(Of TValue))>)
Wintellect.PowerCollections::MultiDictionaryBase(Of TKey, TValue)
Wintellect.PowerCollections::OrderedMultiDictionary(Of TKey, TValue)
See Also

OrderedMultiDictionary<(Of <TKey, TValue>)> Members
Wintellect.PowerCollections Namespace
Wintellect.PowerCollections::MultiDictionary<(Of <TKey, TValue>)>
Wintellect.PowerCollections::OrderedDictionary<(Of <TKey, TValue>)>

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of<TKey, TValue>)> Members

See Also  Methods  Properties  Constructors  Explicit Interface Implementations

☐ Include Inherited Members  ☑ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class OrderedMultiDictionary<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute>
Public Class OrderedMultiDictionary(Of TKey, TValue)

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class OrderedMultiDictionary
Type Parameters

TKey
TValue

The type exposes the following members.
## Public Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OrderedMultiDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;</code></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><code>OrderedMultiDictionary&lt;(Of &lt;TKey, TValue&gt;)&gt;</code></td>
<td>New</td>
</tr>
</tbody>
</table>
### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td></td>
<td>Adds new values to be associated with a key. If duplicate values are permitted, this method always adds new key-value pairs to the dictionary.</td>
</tr>
<tr>
<td><strong>AddMany</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If duplicate values are not permitted, and key already has a value equal to one of values associated with it, then that value is replaced, and the number of values associate with key is unchanged.</td>
</tr>
</tbody>
</table>

((Inherited from MultiDictionaryBase<Of TKey, TValue>.)

**AsReadOnly**

Provides a read-only view of this collection. The returned ICollection<T> provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes.

Since the returned object is just a view, changes to the collection will be reflected in the view.

(Inherited from CollectionBase<Of T>.)

**Clear**

Removes all keys and values from the dictionary.

(Overrides MultiDictionaryBase<Of TKey, TValue>);

**Clone**

Makes a shallow clone of this dictionary; i.e., if keys or values of the dictionary are reference types, then they are not cloned. If TKey or TValue is a value type, then each element is copied as if by simple assignment.

**CloneContents**

Makes a deep clone of this dictionary. A new dictionary is created with a clone of each entry of this dictionary, by calling ICloneable.Clone on each element. If TKey or TValue is a value type, then each element is copied as if by simple assignment.
**Contains**

Checks to see if the key is present in the dictionary and has at least one value associated with it.

(Overrides `MultiDictionaryBase(Of <TKey, TValue>)..::.ContainsKey(TKey).`)

**ContainsKey**

Converts to see if the key is present in the dictionary and has at least one value associated with it.

(Overrides `MultiDictionaryBase(Of <TKey, TValue>)..::.ContainsKey(TKey).`)

**ConvertAll(Of <TOutput>])**

Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.

(Inherited from `CollectionBase(Of <T>).`)

**CopyTo**

Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.

(Inherited from `CollectionBase(Of <T>).`)

**CountWhere**

Counts the number of items in the collection that satisfy the condition defined by predicate.

(Inherited from `CollectionBase(Of <T>).`)

**Equals**

Determines whether the specified `Object` is equal to the current `Object`.

(Inherited from `Object`.)

**Exists**

Determines if the collection contains any item that satisfies the condition defined by predicate.

(Inherited from `CollectionBase(Of <T>).`)

**FindAll**

Enumerates the items in the collection that satisfy the condition defined by predicate.

(Inherited from `CollectionBase(Of <T>).`)

**ForEach**

Performs the specified action on each item in this collection.

(Inherited from `CollectionBase(Of <T>).`)

**GetEnumerator**

Enumerates all the keys in the dictionary, and for each key, the collection of values for that key.

(Inherited from `MultiDictionaryBase(Of <TKey, TValue>).`)

**GetHashCode**

Serves as a hash function for a particular type.

`GetHashCode()` is suitable for use in hashing algorithms and data structures like a hash table.

(Inherited from `Object`.)
**GetType**
Gets the Type of the current instance.
(Inherited from Object.)

Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are greater than from and less than to are included. The keys are enumerated in sorted order. Keys equal to the end points of the range can be included or excluded depending on the fromInclusive and toInclusive parameters.

**Range**
Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are greater than (and optionally, equal to) from are included. The keys are enumerated in sorted order. Keys equal to from can be included or excluded depending on the fromInclusive parameter.

**RangeFrom**
Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are greater than (and optionally, equal to) from are included. The keys are enumerated in sorted order. Keys equal to from can be included or excluded depending on the fromInclusive parameter.

**RangeTo**
Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are less than (and optionally, equal to) to are included. The keys are enumerated in sorted order. Keys equal to to can be included or excluded depending on the toInclusive parameter.

**Remove**
Overloaded.

Removes all the items in the collection that satisfy the condition defined by predicate.
(Inherited from CollectionBase<Of <T> >.)

**RemoveAll**
Overloaded.

Removes all the items in the collection that satisfy the condition defined by predicate.
(Inherited from CollectionBase<Of <T> >.)

**RemoveMany**
Overloaded.

Replaces all values associated with key with the single value.
(Inherited from MultiDictionaryBase<Of <TKey, TValue> >.)

**Replace**
Replaces all values associated with key with a new collection of values. If the collection does not permit duplicate values, and values has duplicate items, then only the last of duplicates is added.
(Inherited from MultiDictionaryBase<Of <TKey, TValue> >.)

**ReplaceMany**
Returns a View collection that can be used for enumerating the keys and values in the collection in reversed order.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ToArray</strong></td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary. (Inherited from <code>MultiDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;</code>)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TrueForAll</strong></td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CountAllValues</strong></td>
<td>Gets a total count of values in the collection.</td>
</tr>
<tr>
<td></td>
<td>(Overrides MultiDictionaryBase&lt;Of TKey, TValue&gt;...............:CountAllValues().)</td>
</tr>
<tr>
<td><strong>CountValues</strong></td>
<td>Gets the number of values associated with a given key.</td>
</tr>
<tr>
<td></td>
<td>(Overrides MultiDictionaryBase&lt;Of TKey, TValue&gt;...............:CountValues(TKey).)</td>
</tr>
<tr>
<td><strong>EnumerateKeys</strong></td>
<td>Enumerate all of the keys in the dictionary.</td>
</tr>
<tr>
<td></td>
<td>(Overrides MultiDictionaryBase&lt;Of TKey, TValue&gt;...............:EnumerateKeys().)</td>
</tr>
<tr>
<td><strong>EqualValues</strong></td>
<td>Determine if two values are equal.</td>
</tr>
<tr>
<td></td>
<td>(Overrides MultiDictionaryBase&lt;Of TKey, TValue&gt;...............:EqualValues(TValue, TValue).)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an Object to attempt to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td><strong>TryEnumerateValuesForKey</strong></td>
<td>Determines if this dictionary contains a key equal to key. If so, all the</td>
</tr>
<tr>
<td></td>
<td>values associated with that key are returned through the values</td>
</tr>
<tr>
<td></td>
<td>(Overrides MultiDictionaryBase&lt;Of TKey, TValue&gt;...............:TryEnumerateValuesForKey(TKey, IEnumerator&lt;Of TValue&gt;&gt;&lt;%).)</td>
</tr>
</tbody>
</table>
# Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Gets the number of key-value pairs in the dictionary. Each value associated with a given key is counted. If duplicate values are permitted, each duplicate value is included in the count. (Overrides <code>MultiDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;.::Count.) Returns a collection of all of the values in the dictionary associated with key, or changes the set of values associated with key. If the key is not present in the dictionary, an ICollection enumerating no values is returned. The returned collection of values is read-write, and can be used to modify the collection of values associated with the key. (Inherited from </code>MultiDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;.)</td>
</tr>
<tr>
<td>Item</td>
<td>Returns the IComparer&lt;T&gt; used to compare keys in this dictionary.</td>
</tr>
<tr>
<td>KeyComparer</td>
<td>Gets a read-only collection all the keys in this dictionary. (Inherited from `MultiDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;.)</td>
</tr>
<tr>
<td>Keys</td>
<td>Gets a read-only collection of all key-value pairs in the dictionary. If a key has multiple values associated with it, then a key-value pair is present for each value associated with the key. (Overrides `MultiDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;...::KeyValuePairs.)</td>
</tr>
<tr>
<td>KeyValuePairs</td>
<td></td>
</tr>
<tr>
<td>ValueComparer</td>
<td>Returns the IComparer&lt;T&gt; used to compare values in this dictionary.</td>
</tr>
<tr>
<td>Values</td>
<td>Gets a read-only collection of all the values in the dictionary. (Inherited from `MultiDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection&lt;Of T&gt;,::IsReadOnly</td>
<td>(Inherited from CollectionBase&lt;Of T&gt;.)</td>
</tr>
<tr>
<td>IDictionary&lt;Of TKey, ICollection&lt;Of TValue&gt;&gt;,::Add</td>
<td>(Inherited from MultiDictionaryBase&lt;Of TKey, TValue&gt;.)</td>
</tr>
<tr>
<td>IDictionary&lt;Of TKey, ICollection&lt;Of TValue&gt;&gt;,::Item</td>
<td>(Inherited from MultiDictionaryBase&lt;Of TKey, TValue&gt;.)</td>
</tr>
<tr>
<td>IDictionary&lt;Of TKey, ICollection&lt;Of TValue&gt;&gt;,::TryGetValue</td>
<td>(Inherited from MultiDictionaryBase&lt;Of TKey, TValue&gt;.)</td>
</tr>
<tr>
<td>ICollection&lt;Of TKey, ICollection&lt;Of TValue&gt;&gt;,::Values</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;Of T&gt;.)</td>
</tr>
<tr>
<td>ICollection::CopyTo</td>
<td>Indicates whether the collection is synchronized. (Inherited from CollectionBase&lt;Of T&gt;.)</td>
</tr>
<tr>
<td>ICollection::IsSynchronized</td>
<td>Indicates the synchronization object for this collection. (Inherited from CollectionBase&lt;Of T&gt;.)</td>
</tr>
<tr>
<td>ICollection::SyncRoot</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the</td>
</tr>
</tbody>
</table>
**IEnumerator<T>** that was overridden by the derived classes to enumerate the members of the collection.
(Inherited from **CollectionBase(Of<T>).*

Implements ICloneable.Clone. Makes a shallow clone of this dictionary; i.e., if keys or values are reference types, then they are not cloned.
See Also

OrderedMultiDictionary(Of TKey, TValue>)
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>}> Constructor

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OrderedMultiDictionary&lt;Of &lt;TKey, TValue&gt;&gt;</code></td>
<td>Create a new <code>OrderedMultiDictionary</code>. The default ordering of keys and values are used. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key &quot;foo&quot; could have &quot;a&quot;, &quot;a&quot;, and &quot;b&quot; associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key &quot;foo&quot; could have &quot;a&quot; and &quot;b&quot; associated with it, which key &quot;bar&quot; has values &quot;b&quot; and &quot;c&quot; associated with it.</td>
</tr>
<tr>
<td><code>OrderedMultiDictionary&lt;Of &lt;TKey, TValue&gt;&gt;</code></td>
<td>Create a new <code>OrderedMultiDictionary</code>. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key &quot;foo&quot; could have &quot;a&quot;, &quot;a&quot;, and &quot;b&quot; associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key &quot;foo&quot; could have &quot;a&quot; and &quot;b&quot; associated with it, which key &quot;bar&quot; has values &quot;b&quot; and &quot;c&quot; associated with it.</td>
</tr>
<tr>
<td><code>New(Boolean)</code></td>
<td></td>
</tr>
<tr>
<td><code>New(Boolean, IComparer&lt;Of &lt;TKey&gt;&gt;)</code></td>
<td></td>
</tr>
</tbody>
</table>
Create a new OrderedMultiDictionary. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key "foo" could have "a", "a", and "b" associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key "foo" could have "a" and "b" associated with it, which key "bar" has values "b" and "c" associated with it.

OrderedMultiDictionary<
(OfTypeKey,
TValue>)
>OrderedMultiDictionary<
(OfTypeKey,
TValue>)
>New(Boolean,
Comparer<(OfTypeKey)>)

OrderedMultiDictionary<
(OfTypeKey,
TValue>)
>OrderedMultiDictionary<
(OfTypeKey,
TValue>)
>New(Boolean,
Comparer<(OfTypeKey)>)
,Comparer<(OfTypeValue)>)

OrderedMultiDictionary<
(OfTypeKey,
TValue>)
>OrderedMultiDictionary<
(OfTypeKey,
TValue>)
>New(Boolean,
Comparer<(OfTypeKey)>)
,Comparer<(OfTypeValue)>)

Create a new OrderedMultiDictionary. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key "foo" could have "a", "a", and "b" associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key "foo" could have "a" and "b" associated with it, which key "bar" has values "b" and "c" associated with it.

Create a new OrderedMultiDictionary. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key "foo" could have "a", "a", and
OrderedMultiDictionary<\(\text{Of}\ <\text{TKey},\ <\text{TValue}>\)>\linebreak OrderedMultiDictionary<\(\text{Of}\ <\text{TKey},\ <\text{TValue}>\)>\linebreak New(Boolean, \text{Comparison}<\(\text{Of}\ <\text{TKey}>\), \text{Comparison}<\(\text{Of}\ <\text{TValue}>\)>)

"b" associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key "foo" could have "a" and "b" associated with it, which key "bar" has values "b" and "c" associated with it.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
OrderedMultiDictionary(Of TKey, TValue) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Create a new OrderedMultiDictionary. The default ordering of keys and values are used. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key "foo" could have "a", "a", and "b" associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key "foo" could have "a" and "b" associated with it, which key "bar" has values "b" and "c" associated with it.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedMultiDictionary(
        bool allowDuplicateValues
    )

Visual Basic (Declaration)

Public Sub New (_
        allowDuplicateValues As Boolean _
    )

Visual C++

public:
    OrderedMultiDictionary (
        bool allowDuplicateValues
    )

Parameters

allowDuplicateValues
        Boolean
        Can the same value be associated with a key multiple times?
Remarks

The default ordering of keys and values will be used, as defined by TKey and TValue's implementation of IComparable<T> (or IComparable if IComparable<T> is not implemented). If a different ordering should be used, other constructors allow a customComparer or IComparer to be passed to changed the ordering.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>TKey or TValue does not implement either IComparable&lt;T&gt; or IComparable.</td>
</tr>
</tbody>
</table>
See Also

OrderedMultiDictionary<(Of <TKey, TValue>)> Class
Wintellect.Power Collections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<Of <TKey, TValue>> Constructor (Boolean, IComparer<Of <TKey>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Create a new OrderedMultiDictionary. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key "foo" could have "a", "a", and "b" associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key "foo" could have "a" and "b" associated with it, which key "bar" has values "b" and "c" associated with it.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public OrderedMultiDictionary(
    bool allowDuplicateValues,
    IComparer<TKey> keyComparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub New ( _
    allowDuplicateValues As Boolean, _
    keyComparer As IComparer(Of TKey) _
)
```

**Visual C++**

```cpp
public:
OrderedMultiDictionary ( 
    bool allowDuplicateValues,
    IComparer<TKey>^ keyComparer
)
```

### Parameters

**allowDuplicateValues**

**Boolean**

Can the same value be associated with a key multiple times?

**keyComparer**

**IComparer<TKey>**

An IComparer<TKey> instance that will be used to compare keys.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>TValue does not implement either IComparable&lt;TValue&gt; or IComparable.</td>
</tr>
</tbody>
</table>
See Also

**OrderedMultiDictionary(Of TKey, TValue)>** Class

**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
Create a new OrderedMultiDictionary. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key "foo" could have "a", "a", and "b" associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key "foo" could have "a" and "b" associated with it, which key "bar" has values "b" and "c" associated with it.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
- **Syntax**

  **C#**

  ```csharp
  public OrderedMultiDictionary(
      bool allowDuplicateValues,
      Comparison<TKey> keyComparison
  )
  ```

  **Visual Basic (Declaration)**

  ```vbnet
  Public Sub New (_
      allowDuplicateValues As Boolean, _
      keyComparison As Comparison(Of TKey) _
  )
  ```

  **Visual C++**

  ```cpp
  public:
  OrderedMultiDictionary ( _
      bool allowDuplicateValues, _
      Comparison<TKey>^ keyComparison
  )
  ```

  **Parameters**

  allowDuplicateValues
    **Boolean**
    Can the same value be associated with a key multiple times?

  keyComparison
    **Comparison(Of <TKey>)**
    A delegate to a method that will be used to compare keys.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::InvalidOperationException</td>
<td>TValue does not implement either IComparable&lt;TValue&gt; or IComparable.</td>
</tr>
</tbody>
</table>
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<Of <TKey, TValue>> Constructor (Boolean, IComparer<Of <TKey>>, IComparer<Of <TValue>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Create a new OrderedMultiDictionary. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key "foo" could have "a", "a", and "b" associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key "foo" could have "a" and "b" associated with it, which key "bar" has values "b" and "c" associated with it.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public OrderedMultiDictionary(
    bool allowDuplicateValues,
    IComparer<TKey> keyComparer,
    IComparer<TValue> valueComparer
)
```

Visual Basic (Declaration)

```vbnet
Public Sub New (
    allowDuplicateValues As Boolean, _
    keyComparer As IComparer(Of TKey), _
    valueComparer As IComparer(Of TValue) _
)
```

Visual C++

```cpp
public:
OrderedMultiDictionary (
    bool allowDuplicateValues,
    IComparer<TKey>* keyComparer,
    IComparer<TValue>* valueComparer
)
```

Parameters

allowDuplicateValues

*Boolean*
Can the same value be associated with a key multiple times?

keyComparer

*IComparer<TKey>*
An IComparer<TKey> instance that will be used to compare keys.

valueComparer

*IComparer<TValue>*

An IComparer<TValue> instance that will be used to compare values.
See Also

`OrderedMultiDictionary(Of TKey, TValue)` Class
`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
Create a new OrderedMultiDictionary. If duplicate values are allowed, multiple copies of the same value can be associated with the same key. For example, the key "foo" could have "a", "a", and "b" associated with it. If duplicate values are not allowed, only one copies of a given value can be associated with the same key, although different keys can have the same value. For example, the key "foo" could have "a" and "b" associated with it, which key "bar" has values "b" and "c" associated with it.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public OrderedMultiDictionary(
    bool allowDuplicateValues,
    Comparison<TKey> keyComparison,
    Comparison<TValue> valueComparison
)
```

### Visual Basic (Declaration)

```vbnet
Public Sub New (
    allowDuplicateValues As Boolean, _
    keyComparison As Comparison(Of TKey), _
    valueComparison As Comparison(Of TValue)
)
```

### Visual C++

```cpp
public:
OrderedMultiDictionary (  
    bool allowDuplicateValues,  
    Comparison<TKey>^ keyComparison,  
    Comparison<TValue>^ valueComparison
)
```

## Parameters

**allowDuplicateValues**

[Boolean]

Can the same value be associated with a key multiple times?

**keyComparison**

[Comparison]<(Of <TKey>)>  
A delegate to a method that will be used to compare keys.

**valueComparison**

[Comparison]<(Of <TValue>)>
A delegate to a method that will be used to compare values.
See Also

[OrderedMultiDictionary]<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)> Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class OrderedMultiDictionary<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute>
Public Class OrderedMultiDictionary(Of TKey, TValue)

Visual C++

[SerializableAttribute]
generic< typename TKey, typename TValue >
public ref class OrderedMultiDictionary
**Type Parameters**

TKey
TValue

The type exposes the following methods.
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td></td>
<td>Adds new values to be associated with a key. If duplicate values are permitted, this method always adds new key-value pairs to the dictionary.</td>
</tr>
<tr>
<td><strong>AddMany</strong></td>
<td>If duplicate values are not permitted, and key already has a value equal to one of values associated with it, then that value is replaced, and the number of values associate with key is unchanged.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from MultiDictionaryBase&lt;TKey, TValue&gt;).</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes.</td>
</tr>
<tr>
<td></td>
<td>Since the returned object is just a view, changes to the collection will be reflected in the view.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from CollectionBase&lt;T&gt;).</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all keys and values from the dictionary.</td>
</tr>
<tr>
<td></td>
<td>(Overrides MultiDictionaryBase&lt;TKey, TValue&gt;...::ClearQQ.)</td>
</tr>
<tr>
<td><strong>Clone</strong></td>
<td>Makes a shallow clone of this dictionary; i.e., if keys or values of the dictionary are reference types, then they are not cloned. If TKey or TValue is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td><strong>CloneContents</strong></td>
<td>Makes a deep clone of this dictionary. A new dictionary is created with a clone of each entry of this dictionary, by calling ICloneable.Clone on each element. If TKey or TValue is a value type, then each element is copied as if by simple assignment.</td>
</tr>
</tbody>
</table>
Contains

Overloaded.
Checks to see if the key is present in the dictionary and has
at least one value associated with it.
(Overrides MultiDictionaryBase<Of <TKey,
TValue>>::ContainsKey(TKey).)

ContainsKey

Convert this collection of items by applying a delegate to
each item in the collection. The resulting enumeration
contains the result of applying converter to each item in this
collection, in order.
(Inherited from CollectionBase<Of <T>>.)

ConvertAll<Of <TOutput>>

Copies all the items in the collection into an array.
Implemented by using the enumerator returned from
CopyTo

GetEnumerator to get all the items and copy them to the
provided array.
(Inherited from CollectionBase<Of <T>>.)

CountWhere

Counts the number of items in the collection that satisfy the
condition defined by predicate.
(Inherited from CollectionBase<Of <T>>.)

Equals

Determines whether the specified Object is equal to the
current Object.
(Inherited from Object.)

Exists

Determines if the collection contains any item that satisfies
the condition defined by predicate.
(Inherited from CollectionBase<Of <T>>.)

FindAll

Enumerates the items in the collection that satisfy the
condition defined by predicate.
(Inherited from CollectionBase<Of <T>>.)

ForEach

Performs the specified action on each item in this collection.
(Inherited from CollectionBase<Of <T>>.)

GetEnumerator

Enumerate all the keys in the dictionary, and for each key,
the collection of values for that key.
(Inherited from MultiDictionaryBase<Of <TKey,
TValue>>.)

GetHashCode

Serves as a hash function for a particular type.
GetHashCode() is suitable for use in hashing algorithms
and data structures like a hash table.
(Inherited from Object.)
- **GetType**
  Gets the `Type` of the current instance.
  (Inherited from `Object`.)
  Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are greater than from and less than to are included. The keys are enumerated in sorted order. Keys equal to the end points of the range can be included or excluded depending on the `fromInclusive` and `toInclusive` parameters.

- **Range**
  Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are greater than and optionally, equal to from are included. The keys are enumerated in sorted order. Keys equal to from can be included or excluded depending on the `fromInclusive` parameter.

- **RangeFrom**
  Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are greater than and optionally, equal to from are included. The keys are enumerated in sorted order. Keys equal to from can be included or excluded depending on the `fromInclusive` parameter.

- **RangeTo**
  Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are less than and optionally, equal to to are included. The keys are enumerated in sorted order. Keys equal to to can be included or excluded depending on the `toInclusive` parameter.

- **Remove**
  Overloaded.
  Removes all the items in the collection that satisfy the condition defined by predicate.
  (Inherited from `CollectionBase<(Of <T>)>`.)

- **RemoveAll**
  Overloaded.
  Removes all the items in the collection that satisfy the condition defined by predicate.
  (Inherited from `CollectionBase<(Of <T>)>`.)

- **RemoveMany**
  Overloaded.
  Replaces all values associated with key with the single value value.
  (Inherited from `MultiDictionaryBase<(Of <TKey, TValue>)>`.)

- **Replace**
  Replaces all values associated with key with a new collection of values. If the collection does not permit duplicate values, and values has duplicate items, then only the last of duplicates is added.
  (Inherited from `MultiDictionaryBase<(Of <TKey, TValue>)>`.)

- **ReplaceMany**
  Returns a View collection that can be used for enumerating the keys and values in the collection in reversed order.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ToArray</strong></td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>MultiDictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>TrueForAll</strong></td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
</tbody>
</table>
### Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountAllValues</td>
<td>Gets a total count of values in the collection.</td>
</tr>
<tr>
<td></td>
<td>(Overrides MultiDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;...::CountAllValues())</td>
</tr>
<tr>
<td>CountValues</td>
<td>Gets the number of values associated with a given key.</td>
</tr>
<tr>
<td></td>
<td>(Overrides MultiDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;...::CountValues(TKey).)</td>
</tr>
<tr>
<td>EnumerateKeys</td>
<td>Enumerate all of the keys in the dictionary.</td>
</tr>
<tr>
<td></td>
<td>(Overrides MultiDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;...::EnumerateKeys())</td>
</tr>
<tr>
<td>EqualValues</td>
<td>Determine if two values are equal.</td>
</tr>
<tr>
<td></td>
<td>(Overrides MultiDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;...::EqualValues(TValue, TValue).)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>TryEnumerateValuesForKey</td>
<td>Determines if this dictionary contains a key equal to key. If so, all the values associated with that key are returned through the values</td>
</tr>
<tr>
<td></td>
<td>(Overrides MultiDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&gt;...::TryEnumerateValuesForKey(TKey, IEnumerator&lt;(Of &lt;TValue&gt;)&gt;%).)</td>
</tr>
</tbody>
</table>
# Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDictionary(Of TKey, ICollection(Of TValue&gt;&gt;),::Add</td>
<td>(Inherited from MultiDictionaryBase(Of TKey, TValue&gt;).)</td>
</tr>
<tr>
<td>ICollection(Of TKey, ICollection(Of TValue&gt;&gt;),::TryGetValue</td>
<td>(Inherited from MultiDictionaryBase(Of TKey, TValue&gt;).)</td>
</tr>
<tr>
<td>ICollection,::CopyTo</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase(Of T).)</td>
</tr>
<tr>
<td>IEnumerable,::GetEnumerator</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from CollectionBase(Of T).)</td>
</tr>
<tr>
<td>ICloneable,::Clone</td>
<td>Implements ICloneable.Clone. Makes a shallow clone of this dictionary; i.e., if keys or values are reference types, then they are not cloned.</td>
</tr>
</tbody>
</table>
See Also

OrderedMultiDictionary<Of <TKey, TValue>>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<\(\text{Of }<\text{ TKey, TValue}>\)>::Add Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Add(KeyValuePair&lt;TKey, ICollection&lt;TValue&gt;&gt;)</code></td>
<td>Adds a key-value pair to the collection. The value part of the pair must be a collection of values to associate with the key. If values are already associated with the given key, the new values are added to the ones associated with that key. (Inherited from <code>MultiDictionaryBase&lt;TKey, TValue&gt;</code>.)</td>
</tr>
<tr>
<td><code>Add(TKey, TValue)</code></td>
<td>Adds a new value to be associated with a key. If duplicate values are permitted, this method always adds a new key-value pair to the dictionary. If duplicate values are not permitted, and key already has a value equal to value associated with it, then that value is replaced with value, and the number of values associate with key is unchanged. (Overrides <code>MultiDictionaryBase&lt;TKey, TValue&gt;</code>...:::<code>Add(TKey, TValue)</code>.)</td>
</tr>
</tbody>
</table>
See Also

OrderedMultiDictionary<(Of <TKey, TValue>)> Class
OrderedMultiDictionary<(Of <TKey, TValue>)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary(Of TKey, TValue)\t\t::\t\tAdd\tMethod\t(TKey, TValue)


See Also

[x]

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Add a new value to be associated with a key. If duplicate values are permitted, this method always adds a new key-value pair to the dictionary.

If duplicate values are not permitted, and key already has a value equal to value associated with it, then that value is replaced with value, and the number of values associate with key is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed void Add(
    TKey key,
    TValue value
)

Visual Basic (Declaration)

Public Overrides NotOverridable Sub Add ( 
    key As TKey, 
    value As TValue _
)

Visual C++

public:
    virtual void Add ( 
        TKey key, 
        TValue value 
    ) override sealed

Parameters

key
    TKey
    The key to associate with.

value
    TValue
    The value to associated with key.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>::Clear Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all keys and values from the dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed void Clear()

Visual Basic (Declaration)

Public Overrides NotOverridable Sub Clear

Visual C++

public:
virtual void Clear () override sealed
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>::Clone Method

See Also

☐ Visual Basic (Declaration) ☐ Visual Basic (Usage)
☐ C#
☐ Visual C++
☐ J#
☐ JScript
☐ XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a shallow clone of this dictionary; i.e., if keys or values of the dictionary are reference types, then they are not cloned. If TKey or TValue is a value type, then each element is copied as if by simple assignment.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedMultiDictionary<TKey, TValue> Clone()

Visual Basic (Declaration)

Public Function Clone As OrderedMultiDictionary(Of TKey, TValue)

Visual C++

public: OrderedMultiDictionary<TKey, TValue>* Clone ()

ReturnValue

The cloned dictionary.
Remarks

Cloning the dictionary takes time $O(N)$, where $N$ is the number of key-value pairs in the dictionary.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary(Of TKey, TValue)>::CloneContents Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a deep clone of this dictionary. A new dictionary is created with a clone of each entry of this dictionary, by calling ICloneable.Clone on each element. If TKey or TValue is a value type, then each element is copied as if by simple assignment.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedMultiDictionary<TKey, TValue> CloneContents()

Visual Basic (Declaration)

Public Function CloneContents As OrderedMultiDictionary(Of TKey, TValue)

Visual C++

public:
OrderedMultiDictionary<TKey, TValue>^ CloneContents ()

Return Value

The cloned dictionary.
**Remarks**

If TKey or TValue is a reference type, it must implement ICloneable. Otherwise, an InvalidOperationException is thrown.

Cloning the dictionary takes time $O(N \log N)$, where $N$ is the number of key-value pairs in the dictionary.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..:::InvalidOperationException</strong></td>
<td>TKey or TValue is a reference type that does not implement ICloneable.</td>
</tr>
</tbody>
</table>
See Also

(OrderedMultiDictionary(Of TKey, TValue)) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary(Of TKey, TValue).::Contains Method

See Also   Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains(KeyValuePair(Of TKey, ICollection(Of TValue)))</td>
<td>Determines if this dictionary contains the given key and all of the values associated with that key. (Inherited from MultiDictionaryBase(Of TKey, TValue).)</td>
</tr>
<tr>
<td>Contains(TKey, TValue)</td>
<td>Checks to see if value is associated with key in the dictionary. (Overrides MultiDictionaryBase(Of TKey, TValue).::Contains(TKey, TValue).)</td>
</tr>
</tbody>
</table>
See Also

OrderedMultiDictionary(Of TKey, TValue)> Class
OrderedMultiDictionary(Of TKey, TValue)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>::Contains Method (TKey, TValue)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Checks to see if value is associated with key in the dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public override sealed bool Contains(TKey key, TValue value)
```

Visual Basic (Declaration)

```vbnet
Public Overrides NotOverridable Function Contains ( _
key As TKey, _
value As TValue _
) As Boolean
```

Visual C++

```cpp
public:
virtual bool Contains (  
    TKey key,  
    TValue value
) override sealed
```

Parameters

key

- TKey
- The key to check.

value

- TValue
- The value to check.

Return Value

True if value is associated with key.
See Also

**OrderedMultiDictionary(Of TKey, TValue)>** Class
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>::ContainsKey Method

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Checks to see if the key is present in the dictionary and has at least one value associated with it.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public override sealed bool ContainsKey(
    TKey key
)
```

Visual Basic (Declaration)

```vbnet
Public Overrides NotOverridable Function ContainsKey ( _
    key As TKey _
) As Boolean
```

Visual C++

```cpp
public:
virtual bool ContainsKey (  
    TKey key
) override sealed
```

Parameters

key

TKey
The key to check.

Return Value

True if key is present and has at least one value associated with it. Returns false otherwise.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<Of <TKey, TValue>>...::CountAllValues Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets a total count of values in the collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

protected override sealed int CountAllValues()

Visual Basic (Declaration)

Protected Overrides NotOverridable Function CountAllValues As Integer

Visual C++

protected:
virtual int CountAllValues () override sealed

Return Value

The total number of values associated with all keys in the dictionary.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<Of <TKey, TValue>>...::CountValues Method

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets the number of values associated with a given key.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

protected override sealed int CountValues(
    TKey key
)

Visual Basic (Declaration)

Protected Overrides NotOverridable Function CountValues ( _
    key As TKey _
) As Integer

Visual C++

protected:
    virtual int CountValues (  
        TKey key
    ) override sealed

Parameters

key
    TKey
    The key to count values of.

Return Value

The number of values associated with key. If key is not present in the dictionary, zero is returned.
See Also

OrderedMultiDictionary(Of TKey, TValue)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Enumerate all of the keys in the dictionary.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

protected override sealed IEnumerator<TKey> EnumerateKeys()

**Visual Basic (Declaration)**

Protected Overrides NotOverridable Function EnumerateKeys As IEnumerator<TKey>

**Visual C++**

protected:
virtual IEnumerator<TKey>^ EnumerateKeys () override sealed

**Return Value**

An IEnumerator<TKey> of all of the keys in the dictionary.
See Also

**OrderedMultiDictionary(Of TKey, TValue)>** Class
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
OrderedMultiDictionary<Of <TKey, TValue>>::EqualValues Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determine if two values are equal.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

protected override sealed bool EqualValues(
    TValue value1,
    TValue value2
)

Visual Basic (Declaration)

Protected Overrides NotOverridable Function EqualValues ( _
    value1 As TValue, _
    value2 As TValue _
) As Boolean

Visual C++

protected:
    virtual bool EqualValues ( 
        TValue value1,
        TValue value2
    ) override sealed

Parameters

value1
    TValue
    First value to compare.

value2
    TValue
    Second value to compare.

Return Value

True if the values are equal.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary(Of TKey, TValue).::Range Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are greater than from and less than to are included. The keys are enumerated in sorted order. Keys equal to the end points of the range can be included or excluded depending on the fromInclusive and toInclusive parameters.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public OrderedMultiDictionary<TKey, TValue>.OrderedMultiDictionary<TKey, TValue>
    TKey from,
    bool fromInclusive,
    TKey to,
    bool toInclusive
)
```

Visual Basic (Declaration)

```vbnet
Public Function Range (  
    from As TKey,  
    fromInclusive As Boolean,  
    to As TKey,  
    toInclusive As Boolean  
) As OrderedMultiDictionary(Of <TKey, TValue>)..::View
```

Visual C++

```cpp
public:  
    OrderedMultiDictionary<(Of <TKey, TValue>)..::View^  
    TKey from,
    bool fromInclusive,
    TKey to,
    bool toInclusive
)
```

Parameters

from

TKey
The lower bound of the range.

fromInclusive

Boolean
If true, the lower bound is inclusive--keys equal to the lower bound will be included in the range. If false, the lower bound is exclusive--keys equal to
the lower bound will not be included in the range.

to

TKey
The upper bound of the range.

toInclusive

Boolean
If true, the upper bound is inclusive--keys equal to the upper bound will be included in the range. If false, the upper bound is exclusive--keys equal to the upper bound will not be included in the range.

Return Value

An OrderedMultiDictionary.View of key-value pairs in the given range.
Remarks

If from is greater than or equal to to, the returned collection is empty.

The sorted order of the keys is determined by the comparison instance or delegate used to create the dictionary.

Typically, this property is used in conjunction with a foreach statement. For example:

```csharp
foreach(KeyValuePair<TKey, TValue> pair in dictionary) {
    // process pair
}
```

Calling Range does not copy the data in the dictionary, and the operation takes constant time.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary(Of TKey, TValue).::RangeFrom Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a collection that can be used for enumerating some of the keys and values in the collection. Only keys that are greater than (and optionally, equal to) from are included. The keys are enumerated in sorted order. Keys equal to from can be included or excluded depending on the fromInclusive parameter.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public OrderedMultiDictionary<TKey, TValue>.RangeFrom(TKey from,
bool fromInclusive
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function RangeFrom ( _
from As TKey, _
fromInclusive As Boolean _
) As OrderedMultiDictionary(Of TKey, TValue)>.::View
```

**Visual C++**

```cpp
public:
OrderedMultiDictionary<(Of <TKey, TValue>)>::View^ RangeFrom ( 
TKey from,
bool fromInclusive
)
```

### Parameters

**from**

`TKey`

The lower bound of the range.

**fromInclusive**

`Boolean`

If true, the lower bound is inclusive--keys equal to the lower bound will be included in the range. If false, the lower bound is exclusive--keys equal to the lower bound will not be included in the range.

### Return Value
An OrderedMultiDictionary.View of key-value pairs in the given range.
Remarks

The sorted order of the keys is determined by the comparison instance or delegate used to create the dictionary.

Typically, this property is used in conjunction with a foreach statement. For example:

```csharp
foreach(KeyValuePair<TKey, TValue> pair in dictionary) // process pair
```

Calling RangeFrom does not copy of the data in the dictionary, and the operation takes constant time.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary(Of TKey, TValue).::RangeTo Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a collection that can be used for enumerating some of the keys and values in the collection. Only items that are less than (and optionally, equal to) to are included. The items are enumerated in sorted order. Items equal to to can be included or excluded depending on the toInclusive parameter.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedMultiDictionary<TKey, TValue>.OrderedMultiDictionary<TKey, TValue>.RangeTo(TKey to,
bool toInclusive
)

Visual Basic (Declaration)

Public Function RangeTo (_
    to As TKey, _
    toInclusive As Boolean _
) As OrderedMultiDictionary(Of <TKey, TValue>)..:::View

Visual C++

public:
    OrderedMultiDictionary(Of <TKey, TValue>)..:::View^ RangeTo (_
    TKey to,
    bool toInclusive
)

Parameters

to
    TKey
    The upper bound of the range.

toInclusive
    Boolean
    If true, the upper bound is inclusive--keys equal to the upper bound will be included in the range. If false, the upper bound is exclusive--keys equal to the upper bound will not be included in the range.

Return Value
An OrderedMultiDictionary.View of key-value pairs in the given range.
Remarks

The sorted order of the keys is determined by the comparison instance or delegate used to create the dictionary.

Typically, this property is used in conjunction with a foreach statement. For example:

```csharp
foreach(KeyValuePair<TKey, TValue> pair in dictionary."
  // process pair
}
```

Calling RangeTo does not copy the data in the dictionary, and the operation takes constant time.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>::Remove Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove(TKey)</td>
<td>Removes a key from the dictionary. This method must be overridden in the derived class. (Inherited from MultiDictionaryBase&lt;Of &lt;TKey, TValue&gt;).</td>
</tr>
<tr>
<td>Remove(TKey, TValue)</td>
<td>Removes a given value from the values associated with a key. If the last value is removed from a key, the key is removed also. (Overrides MultiDictionaryBase&lt;Of &lt;TKey, TValue&gt;),...::Remove(TKey, TValue).</td>
</tr>
</tbody>
</table>
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
OrderedMultiDictionary(Of TKey, TValue) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary(Of TKey, TValue).Remove Method (TKey, TValue)

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes a given value from the values associated with a key. If the last value is removed from a key, the key is removed also.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public override sealed bool Remove(
    TKey key,
    TValue value
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Function Remove ( _
    key As TKey, _
    value As TValue _
) As Boolean
```

**Visual C++**

```cpp
public:
virtual bool Remove ( 
    TKey key, 
    TValue value
) override sealed
```

**Parameters**

**key**

- **TKey**
  - A key to remove a value from.

**value**

- **TValue**
  - The value to remove.

**Return Value**

True if value was associated with key (and was therefore removed). False if value was not associated with key.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>::RemoveMany Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RemoveMany(IEnumerable&lt;Of &lt;TKey&gt;&gt;)</code></td>
<td>Remove all of the keys (and any associated values) in a collection of keys. If a key is not present in the dictionary, nothing happens. (Inherited from <code>MultiDictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>RemoveMany(TKey, IEnumerable&lt;Of &lt;TValue&gt;&gt;)</code></td>
<td>Removes a collection of values from the values associated with a key. If the last value is removed from a key, the key is removed also. (Inherited from <code>MultiDictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;</code>.)</td>
</tr>
</tbody>
</table>
See Also

OrderedMultiDictionary(Of TKey, TValue)> Class
OrderedMultiDictionary(Of TKey, TValue)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<
(OF<TKey, TValue>)>::Reversed Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a View collection that can be used for enumerating the keys and values in the collection in reversed order.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public OrderedMultiDictionary<TKey, TValue>.OrderedMultiDictionary<TKey, TValue>.Reversed()
```

**Visual Basic (Declaration)**

Public Function Reversed As OrderedMultiDictionary(Of TKey, TValue)

**Visual C++**

```cpp
public: OrderedMultiDictionary(Of TKey, TValue>).View^ Reversed ()
```

### Return Value

An OrderedDictionary.View of key-value pairs in reverse order.
Remarks

Typically, this method is used in conjunction with a foreach statement. For example:

```csharp
foreach(KeyValuePair<TKey, TValue> pair in dictionary.Reversed())
{
    // process pair
}
```

If an entry is added to or deleted from the dictionary while the View is being enumerated, then the enumeration will end with an InvalidOperationException.

Calling Reverse does not copy the data in the dictionary, and the operation takes constant time.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<Of <TKey, TValue>>...::System.ICloneable.Clone Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Implements ICloneable.Clone. Makes a shallow clone of this dictionary; i.e., if keys or values are reference types, then they are not cloned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

**C#**

```csharp
private Object ICloneable.Clone()
```

**Visual Basic (Declaration)**

```vbnet
Private Function System.ICloneable.Clone As Object Implements IClone
```

**Visual C++**

```cpp
private:
virtual Object^ System.ICloneable.Clone() sealed = ICloneable::Clor
```

## Return Value

The cloned dictionary.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary(Of TKey, TValue).::TryEnumerateValuesForKey

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this dictionary contains a key equal to key. If so, all the values associated with that key are returned through the values parameter.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

protected override sealed bool TryEnumerateValuesForKey(
    TKey key,
    out IEnumerator<TValue> values
)

Visual Basic (Declaration)

Protected Overrides NotOverridable Function TryEnumerateValuesForKey(ByVal key As TKey, _
    <OutAttribute> ByRef values As IEnumerable(Of TValue) _
) As Boolean

Visual C++

protected:
    virtual bool TryEnumerateValuesForKey ( 
    TKey key,
    [OutAttribute] IEnumerable<TValue>^ values
) override sealed

Parameters

key
    TKey
    The key to search for.

values
    IEnumerable<(Of TValue)>%
    Returns all values associated with key, if true was returned.

Return Value

True if the dictionary contains key. False if the dictionary does not contain key.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)> Properties

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class OrderedMultiDictionary<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class OrderedMultiDictionary(Of TKey, TValue)

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class OrderedMultiDictionary
Type Parameters

TKey
TValue

The type exposes the following properties.
# Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Gets the number of key-value pairs in the dictionary. Each value associated with a given key is counted. If duplicate values are permitted, each duplicate value is included in the count. (Overrides <code>MultiDictionaryBase&lt;TKey, TValue&gt;</code>::<code>Count</code>.)</td>
</tr>
<tr>
<td>Item</td>
<td>Returns a collection of all of the values in the dictionary associated with key, or changes the set of values associated with key. If the key is not present in the dictionary, an <code>Icollection</code> enumerating no values is returned. The returned collection of values is read-write, and can be used to modify the collection of values associated with the key. (Inherited from <code>MultiDictionaryBase&lt;TKey, TValue&gt;</code>.)</td>
</tr>
<tr>
<td>KeyComparer</td>
<td>Returns the <code>IComparer&lt;T&gt;</code> used to compare keys in this dictionary.</td>
</tr>
<tr>
<td>Keys</td>
<td>Gets a read-only collection all the keys in this dictionary. (Inherited from <code>MultiDictionaryBase&lt;TKey, TValue&gt;</code>.)</td>
</tr>
<tr>
<td>KeyValuePairs</td>
<td>Gets a read-only collection of all key-value pairs in the dictionary. If a key has multiple values associated with it, then a key-value pair is present for each value associated with the key. (Overrides <code>MultiDictionaryBase&lt;TKey, TValue&gt;</code>::<code>KeyValuePairs</code>.)</td>
</tr>
<tr>
<td>ValueComparer</td>
<td>Returns the <code>IComparer&lt;T&gt;</code> used to compare values in this dictionary.</td>
</tr>
<tr>
<td>Values</td>
<td>Gets a read-only collection of all the values in the dictionary. (Inherited from <code>MultiDictionaryBase&lt;TKey, TValue&gt;</code>.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection/(Of &lt;T&gt;),::IsReadOnly</td>
<td>(Inherited from CollectionBase/(Of &lt;T&gt;),.)</td>
</tr>
<tr>
<td>IDictionary/(Of &lt;TKey, TValue&gt;,),::Item</td>
<td>(Inherited from MultiDictionaryBase/(Of &lt;TKey, TValue&gt;,).)</td>
</tr>
<tr>
<td>ICollection/(Of &lt;TKey, TValue&gt;,),::Values</td>
<td>(Inherited from MultiDictionaryBase/(Of &lt;TKey, TValue&gt;,).)</td>
</tr>
<tr>
<td>ICollection,::IsSynchronized</td>
<td>Indicates whether the collection is synchronized. (Inherited from CollectionBase/(Of &lt;T&gt;),.)</td>
</tr>
<tr>
<td>ICollection,::SyncRoot</td>
<td>Indicates the synchronization object for this collection. (Inherited from CollectionBase/(Of &lt;T&gt;),.)</td>
</tr>
</tbody>
</table>
See Also

**OrderedMultiDictionary**(Of **TKey, TValue**)
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:) about this topic to Microsoft.
Gets the number of key-value pairs in the dictionary. Each value associated with a given key is counted. If duplicate values are permitted, each duplicate value is included in the count.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed int Count{ get; }

Visual Basic (Declaration)

Public Overrides NotOverridable ReadOnly Property Count As Integer

Visual C++

public:
virtual property int Count {
   int get () override sealed;
}

Field Value

The number of key-value pairs in the dictionary.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>:::KeyComparer Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the IComparer<T> used to compare keys in this dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Field Value

If the dictionary was created using a comparer, that comparer is returned. If the dictionary was created using a comparison delegate, then a comparer equivalent to that delegate is returned. Otherwise the default comparer for TKey (Comparer<TKey>.Default) is returned.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<Of <TKey, TValue>>::KeyValuePairs Property

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets a read-only collection of all key-value pairs in the dictionary. If a key has multiple values associated with it, then a key-value pair is present for each value associated with the key.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed ICollection<KeyValuePair<TKey, TValue>> KeyValuePairs

Visual Basic (Declaration)

Public Overrides NotOverridable ReadOnly Property KeyValuePairs As.NotNil

Visual C++

public:
virtual property ICollection<KeyValuePair<TKey, TValue>>^ KeyValuePairs

ICollection<KeyValuePair<TKey, TValue>>^ get () override sealed

}
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PoweCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>::ValueComparer Property

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the IComparer<T> used to compare values in this dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public IComparer<TValue> ValueComparer{ get; }

Visual Basic (Declaration)

Public ReadOnly Property ValueComparer As IComparer(Of TValue)

Visual C++

public:
property IComparer<TValue>^ ValueComparer {
    IComparer<TValue>^ get ();
}

Field Value

If the dictionary was created using a comparer, that comparer is returned. If the dictionary was created using a comparison delegate, then a comparer equivalent to that delegate is returned. Otherwise the default comparer for TValue (Comparer<TValue>.Default) is returned.
See Also

OrderedMultiDictionary(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
The OrderedMultiDictionary<`TKey,TValue> View class is used to look at a subset of the keys and values inside an ordered multi-dictionary. It is returned from the Range, RangeTo, RangeFrom, and Reversed methods.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

[SerializableAttribute]
public class View : MultiDictionaryBase<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class View _
    Inherits MultiDictionaryBase(Of TKey, TValue)

Visual C++

[SerializableAttribute]
public ref class View : public MultiDictionaryBase<TKey, TValue>
Remarks

Views are dynamic. If the underlying dictionary changes, the view changes in sync. If a change is made to the view, the underlying dictionary changes accordingly.

Typically, this class is used in conjunction with a foreach statement to enumerate the keys and values in a subset of the OrderedMultiDictionary. For example:

```csharp
foreach(KeyValuePair<TKey, TValue> pair in dictionary.Range(from, to))
{
    // process pair
}
```
Inheritance Hierarchy

System::Object
Wintellect.PowerCollections::CollectionBase<TKey, ICollection<TValue>>
Wintellect.PowerCollections::MultiDictionaryBase<TKey, TValue, KeyValuePair<TKey, TValue>>
Wintellect.PowerCollections::OrderedMultiDictionary<TKey, TValue>
See Also

OrderedMultiDictionary(Of TKey, TValue)>..::View Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>...::View Members

See Also  Methods  Properties  Explicit Interface Implementations

☐  Include Inherited Members  ☑ Include Protected Members
☐  .NET Compact Framework Members Only
☑  XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
The OrderedMultiDictionary class that associates values with a key. Unlike an OrderedDictionary, each key can have multiple values associated with it. When indexing an OrderedMultiDictionary, instead of a single value associated with a key, you retrieve an enumeration of values.

All of the key are stored in sorted order. Also, the values associated with a given key are kept in sorted order as well.

When constructed, you can chose to allow the same value to be associated with a key multiple times, or only one time.

The OrderedMultiDictionary<(Of <TKey, TValue>)> generic type exposes the following members.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Overloaded. Adds new values to be associated with a key. If duplicate values are permitted, this method always adds new key-value pairs to the dictionary.</td>
</tr>
<tr>
<td><strong>AddMany</strong></td>
<td>If duplicate values are not permitted, and key already has a value equal to one of values associated with it, then that value is replaced, and the number of values associate with key is unchanged. (Inherited from MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all the keys and values within this view from the underlying OrderedMultiDictionary. (Overrides MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;::Clear().)</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Overloaded. Tests if the key is present in the part of the dictionary being viewed.</td>
</tr>
<tr>
<td><strong>ContainsKey</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this</td>
</tr>
</tbody>
</table>
collection, in order.
(Inherited from `CollectionBase<Of <T>>`.)

Copies all the items in the collection into an array.
Implemented by using the enumerator returned from
GetEnumerator to get all the items and copy them to the
provided array.
(Inherited from `CollectionBase<Of <T>>`.)

Counts the number of items in the collection that satisfy the
condition defined by predicate.
(Inherited from `CollectionBase<Of <T>>`.)

Determines whether the specified `Object` is equal to the
current `Object`.
(Inherited from `Object`.)

Determines if the collection contains any item that satisfies
the condition defined by predicate.
(Inherited from `CollectionBase<Of <T>>`.)

Enumerates the items in the collection that satisfy the
condition defined by predicate.
(Inherited from `CollectionBase<Of <T>>`.)

Performs the specified action on each item in this collection.
(Inherited from `CollectionBase<Of <T>>`.)

Enumerate all the keys in the dictionary, and for each key,
the collection of values for that key.
(Inherited from `MultiDictionaryBase<Of <TKey, TValue>>`.)

Serves as a hash function for a particular type.

GetHashCode() is suitable for use in hashing algorithms
and data structures like a hash table.
(Inherited from `Object`.)

Gets the `Type` of the current instance.
(Inherited from `Object`.)

Overloaded.

Removes all the items in the collection that satisfy the
condition defined by predicate.
(Inherited from `CollectionBase<Of <T>>`.)

Overloaded.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replace</strong></td>
<td>Replaces all values associated with key with the single value value. (Inherited from <strong>MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;</strong>.)</td>
</tr>
<tr>
<td><strong>ReplaceMany</strong></td>
<td>Replaces all values associated with key with a new collection of values. If the collection does not permit duplicate values, and values has duplicate items, then only the last of duplicates is added. (Inherited from <strong>MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;</strong>.)</td>
</tr>
<tr>
<td><strong>Reversed</strong></td>
<td>Creates a new View that has the same keys and values as this, in the reversed order. Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from <strong>CollectionBase&lt;Of&lt;T&gt;&gt;</strong>.)</td>
</tr>
<tr>
<td><strong>ToArray</strong></td>
<td>Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary. (Inherited from <strong>MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;</strong>.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from <strong>CollectionBase&lt;Of&lt;T&gt;&gt;</strong>.)</td>
</tr>
</tbody>
</table>
# Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountAllValues</td>
<td>Gets a total count of values in the collection. This default implementation is slow; it enumerates all of the keys in the dictionary and calls CountValues on each. A derived class may be able to supply a more efficient implementation. (Inherited from MultiDictionaryBase(Of TKey, TValue&gt;).)</td>
</tr>
<tr>
<td>CountValues</td>
<td>Gets the number of values associated with a given key. (Overrides MultiDictionaryBase(Of TKey, TValue&gt;)::.CountValues(TKey).)</td>
</tr>
<tr>
<td>EnumerateKeys</td>
<td>Enumerate all the keys in the dictionary. (Overrides MultiDictionaryBase(Of TKey, TValue&gt;)::.EnumerateKeys().)</td>
</tr>
<tr>
<td>EqualValues</td>
<td>If the derived class does not use the default comparison for values, this methods should be overridden to compare two values for equality. This is used for the correct implementation of ICollection.Contains on the Values and KeyValuePairs collections. (Inherited from MultiDictionaryBase(Of TKey, TValue&gt;).)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

- `[CountAllValues]`  
- `[CountValues]`  
- `[EnumerateKeys]`  
- `[EqualValues]`  
- `[Finalize]`  
- `[MemberwiseClone]`
TryEnumerateValuesForKey values is returned throught values. If the key does not exist, false is returned.
(Overrides MultiDictionaryBase<(Of <TKey, TValue>))->...:TryEnumerateValuesForKey(TKey, IEnumerable<(Of <TValue>)>)%).
### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td>Number of keys in this view. (Overrides <code>MultiDictionaryBase(Of&lt;TKey, TValue&gt;...)::Count</code>.) Returns a collection of all of the values in the dictionary associated with key, or changes the set of values associated with key. If the key is not present in the dictionary, an ICollection enumerating no values is returned. The returned collection of values is read-write, and can be used to modify the collection of values associated with the key. (Inherited from <code>MultiDictionaryBase(Of&lt;TKey, TValue&gt;)</code>.)</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Gets a read-only collection all the keys in this dictionary. (Inherited from <code>MultiDictionaryBase(Of&lt;TKey, TValue&gt;)</code>.)</td>
</tr>
<tr>
<td><strong>Keys</strong></td>
<td>Gets a read-only collection of all key-value pairs in the dictionary. If a key has multiple values associated with it, then a key-value pair is present for each value associated with the key. (Inherited from <code>MultiDictionaryBase(Of&lt;TKey, TValue&gt;)</code>.)</td>
</tr>
<tr>
<td><strong>KeyValuePairs</strong></td>
<td>Gets a read-only collection of all the values in the dictionary. (Inherited from <code>MultiDictionaryBase(Of&lt;TKey, TValue&gt;)</code>.)</td>
</tr>
<tr>
<td><strong>Values</strong></td>
<td></td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `ICollection<(Of `T`)>..::IsReadOnly` | (Inherited from `CollectionBase<(Of `T`)>`.)
| `IDictionary<(Of `<TKey`, `TValue`)>..::Add` | (Inherited from `MultiDictionaryBase<(Of `<TKey`, `TValue`)>`.)
| `ICollection<(Of `<TValue`)>..::Item` | (Inherited from `MultiDictionaryBase<(Of `<TKey`, `TValue`)>`.)
| `IDictionary<(Of `<TKey`, `TValue`)>..::TryGetValue` | (Inherited from `MultiDictionaryBase<(Of `<TKey`, `TValue`)>`.)
| `ICollection<(Of `<TValue`)>..::Values` | (Inherited from `MultiDictionaryBase<(Of `<TKey`, `TValue`)>`.)
| `ICollection..::CopyTo` | Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from `CollectionBase<(Of `T`)>`.)
| `ICollection..::IsSynchronized` | Indicates whether the collection is synchronized. (Inherited from `CollectionBase<(Of `T`)>`.)
| `ICollection..::SyncRoot` | Indicates the synchronization object for this collection. (Inherited from `CollectionBase<(Of `T`)>`.)
| | Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the |
IEnumerable....::GetEnumerator

IEnumerator<T> that was overridden by the derived classes to enumerate the members of the collection. (Inherited from CollectionBase(Of <T>).)
See Also

OrderedMultiDictionary(Of TKey, TValue)>..::.View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>::View Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
**Syntax**

**C#**

```csharp
[SerializableAttribute]
public class View
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute> _
Public Class View
```

**Visual C++**

```cpp
[SerializableAttribute]
public ref class View
```

The `OrderedMultiDictionary(Of TKey, TValue)>` generic type exposes the following methods.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>Add</strong></td>
<td>Adds new values to be associated with a key. If duplicate values are permitted, this method always adds new key-value pairs to the dictionary.</td>
</tr>
<tr>
<td><strong>AddMany</strong></td>
<td>If duplicate values are not permitted, and key already has a value equal to one of values associated with it, then that value is replaced, and the number of values associate with key is unchanged.</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. The returned object is just a view, changes to the collection will be reflected in the view.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all the keys and values within this view from the underlying OrderedMultiDictionary.</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>ContainsKey</strong></td>
<td>Tests if the key is present in the part of the dictionary being viewed.</td>
</tr>
<tr>
<td><strong>ConvertAll&lt;Of</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection.</td>
</tr>
</tbody>
</table>

(Inherited from MultiDictionaryBase<Of TKey, TValue>).

(Inherited from CollectionBase<Of T>).

(Overrides MultiDictionaryBase<Of TKey, TValue>::.Clear().)

(Overrides ContainsKey<TKey>.)
collection, in order.  
(Inherited from `CollectionBase(Of <T>)`.)

Copies all the items in the collection into an array.  
Implemented by using the enumerator returned from  
`GetEnumerator` to get all the items and copy them to the  
provided array.  
(Inherited from `CollectionBase(Of <T>)`.)

Counts the number of items in the collection that satisfy the  
condition defined by predicate.  
(Inherited from `CollectionBase(Of <T>)`.)

Determines whether the specified `Object` is equal to the  
current `Object`.  
(Inherited from `Object`.)

Determines if the collection contains any item that satisfies  
the condition defined by predicate.  
(Inherited from `CollectionBase(Of <T>)`.)

Enumerates the items in the collection that satisfy the  
condition defined by predicate.  
(Inherited from `CollectionBase(Of <T>)`.)

Performs the specified action on each item in this collection.  
(Inherited from `CollectionBase(Of <T>)`.)

Enumerate all the keys in the dictionary, and for each key,  
the collection of values for that key.  
(Inherited from `MultiDictionaryBase(Of TKey, TValue)`.)

Serves as a hash function for a particular type.  
`GetHashCode()` is suitable for use in hashing algorithms  
and data structures like a hash table.  
(Inherited from `Object`.)

Gets the `Type` of the current instance.  
(Inherited from `Object`.)

Overloaded.  

Removes all the items in the collection that satisfy the  
condition defined by predicate.  
(Inherited from `CollectionBase(Of <T>)`.)

Overloaded.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replace</strong></td>
<td>Replaces all values associated with key with the single value value. (Inherited from MultiDictionaryBase&lt;Of TKey, TValue&gt;).</td>
</tr>
<tr>
<td><strong>ReplaceMany</strong></td>
<td>Replaces all values associated with key with a new collection of values. If the collection does not permit duplicate values, and values has duplicate items, then only the last of duplicates is added. (Inherited from MultiDictionaryBase&lt;Of TKey, TValue&gt;).</td>
</tr>
<tr>
<td><strong>Reversed</strong></td>
<td>Creates a new View that has the same keys and values as this, in the reversed order.</td>
</tr>
<tr>
<td><strong>ToArray</strong></td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from CollectionBase&lt;Of T&gt;).</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary. (Inherited from MultiDictionaryBase&lt;Of TKey, TValue&gt;).</td>
</tr>
<tr>
<td><strong>TrueForAll</strong></td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from CollectionBase&lt;Of T&gt;).</td>
</tr>
</tbody>
</table>
# Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountAllValues</td>
<td>Gets a total count of values in the collection. This default implementation is slow; it enumerates all of the keys in the dictionary and calls CountValues on each. A derived class may be able to supply a more efficient implementation. (Inherited from MultiDictionaryBase(Of TKey, TValue&gt;).)</td>
</tr>
<tr>
<td>CountValues</td>
<td>Gets the number of values associated with a given key. (Overrides MultiDictionaryBase(Of TKey, TValue)&gt;...: CountValues(TKey).)</td>
</tr>
<tr>
<td>EnumerateKeys</td>
<td>Enumerate all the keys in the dictionary. (Overrides MultiDictionaryBase(Of TKey, TValue)&gt;...: EnumerateKeys()().)</td>
</tr>
<tr>
<td>EqualValues</td>
<td>If the derived class does not use the default comparison for values, this methods should be overridden to compare two values for equality. This is used for the correct implementation of ICollection.Contains on the Values and KeyValuePairs collections. (Inherited from MultiDictionaryBase(Of TKey, TValue&gt;).)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

---
TryEnumerateValuesForKey values is returned through values. If the key does not exist, false is returned.
(Overrides MultiDictionaryBase<(Of <TKey, TValue>))...:: TryEnumerateValuesForKey(TKey, IEnumerator<(Of <TValue>))%).
### Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDictionary&lt;Of TKey, TValue&gt;&gt;&gt;&lt;::Add</td>
<td>(Inherited from MultiDictionaryBase&lt;Of TKey, TValue&gt;.)</td>
</tr>
<tr>
<td>ICollection&lt;Of TKey, TValue&gt;&gt;&gt;&lt;::TryGetValue</td>
<td>(Inherited from MultiDictionaryBase&lt;Of TKey, TValue&gt;.)</td>
</tr>
<tr>
<td>ICollection&lt;Of TKey, TValue&gt;&gt;&gt;&lt;::CopyTo</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;Of T&gt;.)</td>
</tr>
<tr>
<td>IEnumerable&lt;Of TKey, TValue&gt;&gt;&gt;&lt;::GetEnumerator</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from CollectionBase&lt;Of T&gt;.)</td>
</tr>
</tbody>
</table>
See Also

`OrderedMultiDictionary< Of <TKey, TValue> >`::View Class
`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary(Of TKey, TValue)->View::Add Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add(KeyValuePair&lt;TKey, ICollection&lt;TValue&gt;&gt;)</td>
<td>Adds a key-value pair to the collection. The value part of the pair must be a collection of values to associate with the key. If values are already associated with the given key, the new values are added to the ones associated with that key. (Inherited from MultiDictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;.)</td>
</tr>
<tr>
<td>Add(TKey, TValue)</td>
<td>Adds the given key-value pair to the underlying dictionary of this view. If key is not within the range of this view, an ArgumentException is thrown. (Overrides MultiDictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;::Add(TKey, TValue).)</td>
</tr>
</tbody>
</table>
See Also

OrderedMultiDictionary(Of TKey, TValue)>...::View Class
OrderedMultiDictionary(Of TKey, TValue)>...::View Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary&lt;(Of <TKey, TValue>&gt;)::View::Add Method (TKey, TValue)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds the given key-value pair to the underlying dictionary of this view. If key is not within the range of this view, an ArgumentException is thrown.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed void Add(
    TKey key,
    TValue value
)

Visual Basic (Declaration)

Public Overrides NotOverridable Sub Add (_
   key As TKey, _
   value As TValue _)  
)

Visual C++

public:
    virtual void Add ( 
    TKey key, 
    TValue value  
) override sealed

Parameters

key 
    TKey

value 
    TValue
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentException</code></td>
<td>key is not within the range of this view.</td>
</tr>
</tbody>
</table>
See Also

OrderedMultiDictionary(Of TKey, TValue)>...::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<Of <TKey, TValue>>>::View::Clear Method

See Also  Example

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all the keys and values within this view from the underlying OrderedMultiDictionary.

**Namespace:** Wintellect.PowerCollections
**Assembly:**  PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed void Clear() 

Visual Basic (Declaration)

Public Overrides NotOverridable Sub Clear

Visual C++

public:
virtual void Clear () override sealed
Examples

The following removes all the keys that start with "A" from an OrderedMultiDictionary.

```
Copy Code

dictionary.Range("A", "B").Clear();
```
See Also

OrderedMultiDictionary(Of TKey, TValue)> View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>...::View...::Contains Method
See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains(KeyValuePair(Of TKeyType, ICollection(Of TValue)&gt;)&gt; &gt;)</td>
<td>Determines if this dictionary contains the given key and all of the values associated with that key. (Inherited from MultiDictionaryBase(Of TKey, TValue&gt;).)</td>
</tr>
<tr>
<td>Contains(TKey, TValue)</td>
<td>Tests if the key-value pair is present in the part of the dictionary being viewed. (Overrides MultiDictionaryBase(Of TKey, TValue&gt;). ..::Contains(TKey, TValue).)</td>
</tr>
</tbody>
</table>
See Also

OrderedMultiDictionary<(Of <TKey, TValue>)> View Class
OrderedMultiDictionary<(Of <TKey, TValue>)> View Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>::View:::Contains Method (TKey, TValue)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Tests if the key-value pair is present in the part of the dictionary being viewed.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public override sealed bool Contains(
    TKey key,
    TValue value
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Function Contains ( _
    key As TKey, _
    value As TValue _
) As Boolean
```

**Visual C++**

```c++
public:
    virtual bool Contains (  
        TKey key,
        TValue value  
    ) override sealed
```

**Parameters**

**key**

- TKey
  - Key to check for.

**value**

- TValue
  - Value to check for.

**Return Value**

True if the key-value pair is within this view.
See Also

OrderedMultiDictionary(Of TKey, TValue)>...::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<
(Of <TKey, TValue>)>
::
::
View
::
::
ContainsKey
Method

See Also

Tests if the key is present in the part of the dictionary being viewed.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public override sealed bool ContainsKey(
    TKey key
)
```

Visual Basic (Declaration)

```vbnet
Public Overrides NotOverridable Function ContainsKey ( _
    key As TKey _
) As Boolean
```

Visual C++

```cpp
public:
    virtual bool ContainsKey ( 
        TKey key
    ) override sealed
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>TKey</td>
</tr>
</tbody>
</table>

Return Value

True if the key is within this view.
See Also

OrderedMultiDictionary(Of TKey, TValue)>..::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary(Of TKey, TValue)>...::View...::CountValues Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets the number of values associated with a given key.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

protected override sealed int CountValues(
    TKey key
)

Visual Basic (Declaration)

Protected Overrides NotOverridable Function CountValues ( _
    key As TKey _
) As Integer

Visual C++

protected:
    virtual int CountValues ( 
        TKey key
    ) override sealed

Parameters

key
    TKey
    The key to count values of.

Return Value

The number of values associated with key. If key is not present in this view, zero is returned.
See Also

`OrderedMultiDictionary<(Of <TKey, TValue>)>`, `::View` Class

`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<Of <TKey, TValue>>::View::::EnumerateKeys

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerate all the keys in the dictionary.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

protected override sealed IEnumerator<TKey> EnumerateKeys()

Visual Basic (Declaration)

Protected Overrides NotOverridable Function EnumerateKeys As IEnumerable

Visual C++

protected:
virtual IEnumerable<TKey>^ EnumerateKeys () override sealed

Return Value

An IEnumerator<TKey> that enumerates all of the keys in the collection that have at least one value associated with them.
See Also

OrderedMultiDictionary(Of TKey, TValue)>::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>...::View...::Remove Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove(TKey)</td>
<td>Removes a key from the dictionary. This method must be overridden in the derived class. (Inherited from MultiDictionaryBase&lt;Of &lt;TKey, TValue&gt;).</td>
</tr>
<tr>
<td>Remove(TKey, TValue)</td>
<td>Removes the key and value from the underlying dictionary of this view. that is equal to the passed in key. If no key in the view is equal to the passed key, or has the given value associated with it, the dictionary and view are unchanged. (Overrides MultiDictionaryBase&lt;Of &lt;TKey, TValue&gt;).</td>
</tr>
</tbody>
</table>
See Also

OrderedMultiDictionary(Of TKey, TValue)>...::View Class
OrderedMultiDictionary(Of<TKey, TValue>)...::View Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>::View::<::Remove Method (TKey, TValue)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes the key and value from the underlying dictionary of this view. that is equal to the passed in key. If no key in the view is equal to the passed key, or has the given value associated with it, the dictionary and view are unchanged.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public override sealed bool Remove(
    TKey key,
    TValue value
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Function Remove ( _
    key As TKey, _
    value As TValue _
) As Boolean
```

**Visual C++**

```cpp
public:
    virtual bool Remove ( 
        TKey key, 
        TValue value 
    ) override sealed
```

### Parameters

- **key**
  - TKey
  - The key to remove.

- **value**
  - TValue
  - The value to remove.

### Return Value

True if the key-value pair was found and removed. False if the key-value pair was not found.
See Also

OrderedMultiDictionary<Of<TKey, TValue>><..::View Class
WinTellecT.PowerCollect ions Namespace

Send comments about this topic to Microsoft.
View...::RemoveMany Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveMany(IEnumerable&lt;Of &lt;TKey&gt;&gt;)</td>
<td>Remove all of the keys (and any associated values) in a collection of keys. If a key is not present in the dictionary, nothing happens. (Inherited from MultiDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&amp;gt.;)</td>
</tr>
<tr>
<td>RemoveMany(TKey, IEnumerable&lt;(Of &lt;TValue&gt;)&gt;)</td>
<td>Removes a collection of values from the values associated with a key. If the last value is removed from a key, the key is removed also. (Inherited from MultiDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)&amp;gt.;)</td>
</tr>
</tbody>
</table>
See Also

OrderedMultidictionary(Of TKey, TValue)>..::.View Class
OrderedMultidictionary(Of TKey, TValue)>..::.View Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<TKey, TValue>::View::Reversed Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new View that has the same keys and values as this, in the reversed order.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedMultiDictionary<TKey, TValue>.OrderedMultiDictionary<R>(

Visual Basic (Declaration)

Public Function Reversed As OrderedMultiDictionary(Of TKey, TValue)

Visual C++

public:
OrderedMultiDictionary(Of TKey, TValue).::View^ Reversed ()

Return Value

A new View that has the reversed order of this view.
See Also

OrderedMultiDictionary<(Of <TKey, TValue>)>::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>::View::TryEnumerateValuesForKey Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerate all of the values associated with a given key. If the key exists and has values associated with it, an enumerator for those values is returned through \texttt{Values}. If the key does not exist, false is returned.

**Namespace**: Wintellect.PowerCollections

**Assembly**: PowerCollections (in PowerCollections.dll)
# Syntax

## C#

```csharp
protected override sealed bool TryEnumerateValuesForKey(TKey key,
            out IEnumerator<TValue> values)
```

## Visual Basic (Declaration)

```vbnet
Protected Overrides NotOverridable Function TryEnumerateValuesForKey(key As TKey, _
            <OutAttribute> ByRef values As IEnumerator(Of TValue) _) As Boolean
```

## Visual C++

```cpp
protected:
    virtual bool TryEnumerateValuesForKey ( TKey key,
            [OutAttribute] IEnumerator<TValue>^% values
    ) override sealed
```

### Parameters

- **key**
  - TKey
  - The key to get values for.

- **values**
  - `IEnumerator<Of <TValue>>%`
  - If true is returned, this parameter receives an enumerators that enumerates the values associated with that key.

### Return Value

True if the key exists and has values associated with it. False otherwise.
See Also

OrderedMultiDictionary(Of TKey, TValue)>...::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>...::View Properties

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class View

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class View

Visual C++

[SerializableAttribute]
public ref class View

The OrderedMultiDictionary<Of <TKey, TValue>> generic type exposes the following properties.
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>(Overrides MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;::Count) Returns a collection of all of the values in the dictionary associated with key, or changes the set of values associated with key. If the key is not present in the dictionary, an ICollection enumerating no values is returned. The returned collection of values is read-write, and can be used to modify the collection of values associated with the key. (Inherited from MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;.).</td>
</tr>
<tr>
<td>Item</td>
<td>Gets a read-only collection all the keys in this dictionary. (Inherited from MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;.).</td>
</tr>
<tr>
<td>Keys</td>
<td>Gets a read-only collection of all key-value pairs in the dictionary. If a key has multiple values associated with it, then a key-value pair is present for each value associated with the key. (Inherited from MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;.).</td>
</tr>
<tr>
<td>KeyValuePairs</td>
<td>Gets a read-only collection of all the values in the dictionary. (Inherited from MultiDictionaryBase&lt;Of&lt;TKey, TValue&gt;&gt;.).</td>
</tr>
<tr>
<td>Values</td>
<td></td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;Of &lt;T&gt;&gt;::IsReadOnly</code></td>
<td>(Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IDictionary&lt;Of &lt;TKey, TValue&gt;&gt;::Item</code></td>
<td>(Inherited from <code>MultiDictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection&lt;Of &lt;TKey, TValue&gt;&gt;::Values</code></td>
<td>(Inherited from <code>MultiDictionaryBase&lt;Of &lt;TKey, TValue&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection,::IsSynchronized</code></td>
<td>Indicates whether the collection is synchronized. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection,::SyncRoot</code></td>
<td>Indicates the synchronization object for this collection. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
</tbody>
</table>
See Also

OrderedMultiDictionary(Of TKey, TValue)> View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedMultiDictionary<(Of <TKey, TValue>)>::View::Count Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Number of keys in this view.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed int Count{ get; }

Visual Basic (Declaration)

Public Overrides NotOverridable ReadOnly Property Count As Integer

Visual C++

public:
virtual property int Count {
    int get () override sealed;
}

Field Value

Number of keys that lie within the bounds the view.
See Also

OrderedMultiDictionary<Of<TKey, TValue>>..::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>> Class

See Also  Members

[Visual Basic (Declaration)] [Visual Basic (Usage)]
[C#]
[Visual C++]
[J#]
[JScript]
[XAML]

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

OrderedSet<T> is a collection that contains items of type T. The items are maintained in a sorted order, and duplicate items are not allowed. Each item has an index in the set: the smallest item has index 0, the next smallest item has index 1, and so forth.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
[SerializableAttribute]
public class OrderedSet<T> : CollectionBase<T>, ICollection<T>, IEnumerable<T>, IEnumerable, ICloneable
```

Visual Basic (Declaration)

```vbnet
<SerializableAttribute>
Public Class OrderedSet(Of T) 
    Inherits CollectionBase(Of T) 
    Implements ICollection(Of T), IEnumerable(Of T), IEnumerable, ICloneable
```

Visual C++

```cpp
[SerializableAttribute]
generic<typename T>
public ref class OrderedSet : public CollectionBase<T>, ICollection<T>, IEnumerable<T>, IEnumerable, ICloneable
```
Type Parameters

T
Remarks

The items are compared in one of three ways. If T implements IComparable<TKey> or IComparable, then the CompareTo method of that interface will be used to compare items. Alternatively, a comparison function can be passed in either as a delegate, or as an instance of IComparer<TKey>.

OrderedSet is implemented as a balanced binary tree. Inserting, deleting, and looking up an element all are done in log(N) type, where N is the number of keys in the tree.

Set(Of T) is similar, but uses hashing instead of comparison, and does not maintain the items in sorted order.
Inheritance Hierarchy

System::Object
Wintellect.PowerCollections::CollectionBase<Of <T>>
Wintellect.PowerCollections::OrderedSet<Of <T>>
See Also

OrderedSet<(Of <T>)> Members
Wintellect.PowerCollections Namespace
Wintellect.PowerCollections...::Set<(Of <T>)>

Send comments about this topic to Microsoft.
OrderedSet<Of <T>> Members

See Also  Methods  Properties  Constructors  Explicit Interface Implementations

☐

☐ Include Inherited Members  ☐ Include Protected Members

☐ .NET Compact Framework Members Only

☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class OrderedSet<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class OrderedSet(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class OrderedSet
Type Parameters

T

The type exposes the following members.
## Public Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderedSet&lt;T&gt; &gt; OrderedSet&lt;T&gt; &gt; New</td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Add</code></td>
<td>Must be overridden to allow adding items to this collection. Add all the items in collection to the set. If the set already contains an item equal to one of the items in collection, that item will be replaced.</td>
</tr>
<tr>
<td><code>AddMany</code></td>
<td>Get a read-only list view of the items in this ordered set. The items in the list are in sorted order, with the smallest item at index 0. This view does not copy any data, and reflects any changes to the underlying OrderedSet.</td>
</tr>
<tr>
<td><code>AsList</code></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><code>AsReadOnly</code></td>
<td>Makes a shallow clone of this set; i.e., if items of the set are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td><code>Clear</code></td>
<td>Removes all items from the set. (Overides CollectionBase&lt;Of&lt;T&gt;&gt;.:..:Clear().) Makes a deep clone of this set. A new set is created with a clone of each element of this set, by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment. Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable&lt;T&gt;.Equals or Object.Equals.</td>
</tr>
</tbody>
</table>
ConvertAll(Of TOutput)> Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.
(Inherited from CollectionBase(Of T>.)

CopyTo Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.
(Inherited from CollectionBase(Of T>.)

CountWhere Counts the number of items in the collection that satisfy the condition defined by predicate.
(Inherited from CollectionBase(Of T>.)

Difference Computes the difference of this set with another set. The difference of these two sets is all items that appear in this set, but not in otherSet. A new set is created with the difference of the sets and is returned. This set and the other set are unchanged.

DifferenceWith Computes the difference of this set with another set. The difference of these two sets is all items that appear in this set, but not in otherSet. This set receives the difference of the two sets; the other set is unchanged.

Equals Determines whether the specified Object is equal to the current Object.
(Inherited from Object.)

Exists Determines if the collection contains any item that satisfies the condition defined by predicate.
(Inherited from CollectionBase(Of T>.)

FindAll Enumerates the items in the collection that satisfy the condition defined by predicate.
(Inherited from CollectionBase(Of T>.)

ForEach Performs the specified action on each item in this collection.
(Inherited from CollectionBase(Of T>.)

GetEnumerator Returns an enumerator that enumerates all the items in the set. The items are enumerated in sorted order.
(Overrides CollectionBase<(Of
<T>)>...::GetEnumerator();

- **GetFirst**
  Returns the first item in the set: the item that would appear first if the set was enumerated. This is also the smallest item in the set.

- **GetHashCode**
  Serves as a hash function for a particular type. 
  GetHashCode() is suitable for use in hashing algorithms and data structures like a hash table. 
  (Inherited from **Object**.)

- **GetLast**
  Returns the last item in the set: the item that would appear last if the set was enumerated. This is also the largest item in the set.

- **GetType**
  Gets the **Type** of the current instance. 
  (Inherited from **Object**.)

- **IndexOf**
  Get the index of the given item in the sorted order. The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1.

- **Intersection**
  Computes the intersection of this set with another set. 
  The intersection of two sets is all items that appear in both of the sets. A new set is created with the intersection of the sets and is returned. This set and the other set are unchanged.

- **IntersectionWith**
  Computes the intersection of this set with another set. 
  The intersection of two sets is all items that appear in both of the sets. This set receives the intersection of the two sets, the other set is unchanged.

- **IsDisjointFrom**
  Determines if this set is disjoint from another set. Two sets are disjoint if no item from one set is equal to any item in the other set.

- **IsEqualTo**
  Determines if this set is equal to another set. This set is equal to otherSet if they contain the same items.

- **IsProperSubsetOf**
  Determines if this set is a proper subset of another set.
  Neither set is modified. This set is a subset of otherSet if every element in this set is also in otherSet. Additionally, this set must have strictly fewer items than otherSet.

- **IsProperSupersetOf**
  Determines if this set is a proper superset of another set. 
  Neither set is modified. This set is a proper superset of otherSet if every element in otherSet is also in this set.
- **IsProperSupersetOf**: Additionally, this set must have strictly more items than otherSet.

- **IsSubsetOf**: Determines if this set is a subset of another set. Neither set is modified. This set is a subset of otherSet if every element in this set is also in otherSet.

- **IsSupersetOf**: Determines if this set is a superset of another set. Neither set is modified. This set is a superset of otherSet if every element in otherSet is also in this set.
Remarks

IsSupersetOf is computed in time O(M log N), where M is the size of the otherSet, and N is the size of the this set.

Range

Returns a View collection that can be used for enumerating a range of the items in the set. Only items that are greater than from and less than to are included. The items are enumerated in sorted order. Items equal to the end points of the range can be included or excluded depending on the fromInclusive and toInclusive parameters.

RangeFrom

Returns a View collection that can be used for enumerating a range of the items in the set. Only items that are greater than (and optionally, equal to) from are included. The items are enumerated in sorted order. Items equal to from can be included or excluded depending on the fromInclusive parameter.
RangeTo

RangeTo (and optionally, equal to) to are included. The items are enumerated in sorted order. Items equal to to can be included or excluded depending on the toInclusive parameter.

Remove

Remove

Remove all the items in the collection that satisfy the condition defined by predicate. (Inherited from CollectionBase<Of<T>>.)

RemoveAll

RemoveAll

Removes all the items in the collection that satisfy the condition defined by predicate.

RemoveFirst

RemoveFirst

Removes the first item in the set. This is also the smallest item in the set.

RemoveLast

RemoveLast

Removes the last item in the set. This is also the largest item in the set.

RemoveMany

RemoveMany

Removes all the items in collection from the set. Items not present in the set are ignored.

Reversed

Reversed

Returns a View collection that can be used for enumerating the items in the set in reversed order.

SymmetricDifference

SymmetricDifference

Computes the symmetric difference of this set with another set. The symmetric difference of two sets is all items that appear in either of the sets, but not both. A new set is created with the symmetric difference of the sets and is returned. This set and the other set are unchanged.

Computes the symmetric difference of this set with
SymmetricDifferenceWith

The symmetric difference of two sets is all items that appear in either of the sets, but not both. This set receives the symmetric difference of the two sets; the other set is unchanged.

ToArray

Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.
(Inherited from CollectionBase(Of T>.)

ToString

Shows the string representation of the collection. The string representation contains a list of the items in the collection.
(Contained collections (except string) are expanded recursively.)
(Inherited from CollectionBase(Of T>.)

TrueForAll

Determines if all of the items in the collection satisfy the condition defined by predicate.
(Inherited from CollectionBase(Of T>.)

TryGetItem

Determines if this set contains an item equal to item, according to the comparison mechanism that was used when the set was created. The set is not changed.

If the set does contain an item equal to item, then the item from the set is returned.

Computes the union of this set
Union

The union of two sets is all items that appear in either or both of the sets. A new set is created with the union of the sets and is returned. This set and the other set are unchanged.

UnionWith

Computes the union of this set with another set. The union of two sets is all items that appear in either or both of the sets. This set receives the union of the two sets, the other set is unchanged.
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize</td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>)</td>
</tr>
</tbody>
</table>
### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comparer</strong></td>
<td>Returns the IComparer&lt;T&gt; used to compare items in this set.</td>
</tr>
<tr>
<td><strong>Count</strong></td>
<td>Returns the number of items in the set. (Overrides <code>CollectionBase&lt;Of &lt;T&gt;&gt;::.Count</code>.)</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Get the item by its index in the sorted order. The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;(Of &lt;T&gt;)&gt;</code>.Add</td>
<td>(Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>ICollection&lt;(Of &lt;T&gt;)&gt;</code>.IsReadOnly</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>ICollection</code>.CopyTo</td>
<td>Indicates whether the collection is synchronized. (Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>ICollection</code>.IsSynchronized</td>
<td>Indicates the synchronization object for this collection. (Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>ICollection</code>.SyncRoot</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from <code>CollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>IEnumerable</code>.GetEnumerator</td>
<td>Makes a shallow clone of this set; i.e., if items of the set are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td><code>ICloneable</code>.Clone</td>
<td></td>
</tr>
</tbody>
</table>
See Also

**OrderedSet(Of T)**

**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
OrderedSet<Of <T>> Constructor

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OrderedSet</strong>&lt;<strong>(Of</strong>&lt;br&gt;<strong>T</strong>&gt;)</td>
<td>Creates a new OrderedSet. The <strong>T</strong> must implement IComparable&lt;<strong>T</strong>&gt; or IComparable. The CompareTo method of this interface will be used to compare items in this set.</td>
</tr>
<tr>
<td><strong>OrderedSet</strong>&lt;<strong>(Of</strong>&lt;br&gt;<strong>T</strong>&gt;)&lt;br&gt;New()</td>
<td>Creates a new OrderedSet. The Compare method of the passed comparison object will be used to compare items in this set.</td>
</tr>
<tr>
<td><strong>OrderedSet</strong>&lt;<strong>(Of</strong>&lt;br&gt;<strong>T</strong>&gt;)&lt;br&gt;New(IComparer&lt;<strong>(Of</strong>&lt;br&gt;<strong>T</strong>&gt;)&gt;)</td>
<td>Creates a new OrderedSet. The <strong>T</strong> must implement IComparable&lt;<strong>T</strong>&gt; or IComparable. The CompareTo method of this interface will be used to compare items in this set. The set is initialized with all the items in the given collection.</td>
</tr>
<tr>
<td><strong>OrderedSet</strong>&lt;<strong>(Of</strong>&lt;br&gt;<strong>T</strong>&gt;)&lt;br&gt;New(IEnumerable&lt;<strong>(Of</strong>&lt;br&gt;<strong>T</strong>&gt;)&gt;, IComparer&lt;<strong>(Of</strong>&lt;br&gt;<strong>T</strong>&gt;)&gt;)</td>
<td>Creates a new OrderedSet. The passed delegate will be used to compare items in this set.</td>
</tr>
<tr>
<td><strong>OrderedSet</strong>&lt;<strong>(Of</strong>&lt;br&gt;<strong>T</strong>&gt;)&lt;br&gt;New(Comparer&lt;<strong>(Of</strong>&lt;br&gt;<strong>T</strong>&gt;)&gt;)</td>
<td>Creates a new OrderedSet. The Compare method of the passed comparison object will be used to compare items in this set. The set is initialized with all the items in the given collection.</td>
</tr>
<tr>
<td><strong>OrderedSet</strong>&lt;<strong>(Of</strong>&lt;br&gt;<strong>T</strong>&gt;)&lt;br&gt;New(IEnumerable&lt;<strong>(Of</strong>&lt;br&gt;<strong>T</strong>&gt;)&gt;, Comparator&lt;<strong>(Of</strong>&lt;br&gt;<strong>T</strong>&gt;)&gt;)</td>
<td>Creates a new OrderedSet. The passed delegate will be used to compare items in this set. The set is initialized with all the items in the given collection.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) Class
OrderedSet(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>> Constructor

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedSet. The T must implement IComparable<T> or IComparable. The CompareTo method of this interface will be used to compare items in this set.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedSet() 

Visual Basic (Declaration)

Public Sub New

Visual C++

public:
OrderedSet ();
Remarks

Items that are null are permitted, and will be sorted before all other items.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::InvalidOperationException</td>
<td>T does not implement IComparable&lt;TKey&gt;.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>> Constructor (IComparer<Of <T>>)

**See Also**

- Visual Basic (Declaration) — Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedSet. The Compare method of the passed comparison object will be used to compare items in this set.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedSet(
    IComparer<T> comparer
)

Visual Basic (Declaration)

Public Sub New ( _
    comparer As IComparer(Of T) _
)

Visual C++

public:
OrderedSet ( _
    IComparer<T>^ comparer
)

Parameters

comparer
    IComparer<(Of <T>)> An instance of IComparer<T> that will be used to compare items.
Remarks

The GetHashCode and Equals methods of the provided IComparer<T> will never be called, and need not be implemented.
See Also

**OrderedSet(Of** **T)** Class
**Wintellect.PowerCollections** Namespace

Send comments about this topic to Microsoft.
OrderedSet(Of T) Constructor (IEnumerable(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedSet. The T must implement IComparable<T> or IComparable. The CompareTo method of this interface will be used to compare items in this set. The set is initialized with all the items in the given collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public OrderedSet(
    IEnumerable<T> collection
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub New (_
    collection As IEnumerable(Of T) _
)
```

**Visual C++**

```cpp
public: 
OrderedSet (
    IEnumerable<T>^ collection
)
```

**Parameters**

collection  
**IEnumerable<(Of <T>)>**  
A collection with items to be placed into the OrderedSet.
Remarks

Items that are null are permitted, and will be sorted before all other items.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::InvalidOperationException</code></td>
<td>T does not implement <code>IComparable&lt;TKey&gt;</code></td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet(Of T) Constructor (Comparison(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedSet. The passed delegate will be used to compare items in this set.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public OrderedSet(
    Comparison<T> comparison
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub New ( _
    comparison As Comparison(Of T) _
)
```

**Visual C++**

```cpp
public:
OrderedSet ( 
    Comparison<T>^ comparison
)
```

**Parameters**

`comparison`

`(Comparison(Of <T>)>
A delegate to a method that will be used to compare items.`
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>> Constructor (IEnumerable<Of <T>>, IComparer<Of <T>>)>

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedSet. The Compare method of the passed comparison object will be used to compare items in this set. The set is initialized with all the items in the given collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
# Syntax

**C#**

```csharp
public OrderedSet(
    IEnumerable<T> collection,
    IComparer<T> comparer
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub New ( _
    collection As IEnumerable(Of T), _
    comparer As IComparer(Of T) _
)
```

**Visual C++**

```csharp
public: 
OrderedSet ( 
    IEnumerable<T>^ collection, 
    IComparer<T>^ comparer 
)
```

## Parameters

**collection**

- `IEnumerable<T>`
  - A collection with items to be placed into the OrderedSet.

**comparer**

- `IComparer<T>`
  - An instance of IComparer<T> that will be used to compare items.
Remarks

The GetHashCode and Equals methods of the provided IComparer<T> will never be called, and need not be implemented.
See Also

OrderedSet<Of<T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet(Of <T>) Constructor (IEnumerable(Of <T>), Comparison(Of <T>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new OrderedSet. The passed delegate will be used to compare items in this set. The set is initialized with all the items in the given collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedSet(
    IEnumerable<T> collection,
    Comparison<T> comparison
)

Visual Basic (Declaration)

Public Sub New ( _
    collection As IEnumerable(Of T), _
    comparison As Comparison(Of T) _
)

Visual C++

public:
    OrderedSet ( 
        IEnumerable<T>^ collection,
        Comparison<T>^ comparison
    )

Parameters

collection
    IEnumerable<(Of <T>)> 
    A collection with items to be placed into the OrderedSet.

comparison
    Comparison<(Of <T>)> 
    A delegate to a method that will be used to compare items.
See Also

OrderedSet(Of T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet(Of `<T`)> Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

```
[SerializableAttribute]
public class OrderedSet<T>
```

Visual Basic (Declaration)

```
<SerializableAttribute>
Public Class OrderedSet(Of T)
```

Visual C++

```
[SerializableAttribute]
generic<typename T>
public ref class OrderedSet
```
Type Parameters

T

The type exposes the following methods.
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Must be overridden to allow adding items to this collection.</td>
</tr>
<tr>
<td><strong>AddMany</strong></td>
<td>Adds all the items in collection to the set. If the set already contains an item equal to one of the items in collection, that item will be replaced.</td>
</tr>
<tr>
<td><strong>AsList</strong></td>
<td>Get a read-only list view of the items in this ordered set. The items in the list are in sorted order, with the smallest item at index 0. This view does not copy any data, and reflects any changes to the underlying OrderedSet.</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase(Of &lt;T&gt;).)</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all items from the set.</td>
</tr>
<tr>
<td><strong>Clone</strong></td>
<td>Makes a shallow clone of this set; i.e., if items of the set are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td><strong>CloneContents</strong></td>
<td>Makes a deep clone of this set. A new set is created with a clone of each element of this set, by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable&lt;T&gt;.Equals or Object.Equals.</td>
</tr>
</tbody>
</table>
ConvertAll(Of TOutput>)

Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.
(Inherited from CollectionBase(Of T).)

CopyTo

Copies all the items in the collection into an array.
Implemented by using the enumerator returned from GetEnumerato to get all the items and copy them to the provided array.
(Inherited from CollectionBase(Of T).)

CountWhere

Counts the number of items in the collection that satisfy the condition defined by predicate.
(Inherited from CollectionBase(Of T).)

Difference

Computes the difference of this set with another set. The difference of these two sets is all items that appear in this set, but not in otherSet. A new set is created with the difference of the sets and is returned. This set and the other set are unchanged.

DifferenceWith

Computes the difference of this set with another set. The difference of these two sets is all items that appear in this set, but not in otherSet. This set receives the difference of the two sets; the other set is unchanged.

Equals

Determines whether the specified Object is equal to the current Object.
(Inherited from Object.)

Exists

Determines if the collection contains any item that satisfies the condition defined by predicate.
(Inherited from CollectionBase(Of T).)

FindAll

Enumerates the items in the collection that satisfy the condition defined by predicate.
(Inherited from CollectionBase(Of T).)

ForEach

Performs the specified action on each item in this collection.
(Inherited from CollectionBase(Of T).)

GetEnumerator

Returns an enumerator that enumerates all the items in the set. The items are enumerated in sorted order.
(Overrides CollectionBase(Of
<T>\rangle\ldots::\texttt{GetEnumerator()}\ldots\).

- **GetFirst**
  Returns the first item in the set: the item that would appear first if the set was enumerated. This is also the smallest item in the set.

- **GetHashCode**
  Serves as a hash function for a particular type. \texttt{GetHashCode()} is suitable for use in hashing algorithms and data structures like a hash table. (Inherited from \texttt{Object}.)

- **GetLast**
  Returns the last item in the set: the item that would appear last if the set was enumerated. This is also the largest item in the set.

- **GetType**
  Gets the \texttt{Type} of the current instance. (Inherited from \texttt{Object}.)

- **IndexOf**
  Get the index of the given item in the sorted order. The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1.

- **Intersection**
  Computes the intersection of this set with another set. The intersection of two sets is all items that appear in both of the sets. A new set is created with the intersection of the sets and is returned. This set and the other set are unchanged.

- **IntersectionWith**
  Computes the intersection of this set with another set. The intersection of two sets is all items that appear in both of the sets. This set receives the intersection of the two sets, the other set is unchanged.

- **IsDisjointFrom**
  Determines if this set is disjoint from another set. Two sets are disjoint if no item from one set is equal to any item in the other set.

- **IsEqualTo**
  Determines if this set is equal to another set. This set is equal to \texttt{otherSet} if they contain the same items.

- **IsProperSubsetOf**
  Determines if this set is a proper subset of another set. Neither set is modified. This set is a subset of \texttt{otherSet} if every element in this set is also in \texttt{otherSet}. Additionally, this set must have strictly fewer items than \texttt{otherSet}.

- **IsProperSupersetOf**
  Determines if this set is a proper superset of another set. Neither set is modified. This set is a proper superset of \texttt{otherSet} if every element in \texttt{otherSet} is also in this set.
**IsProperSupersetOf** Additionally, this set must have strictly more items than otherSet.

**IsSubsetOf** Determines if this set is a subset of another set. Neither set is modified. This set is a subset of otherSet if every element in this set is also in otherSet.

**IsSupersetOf** Determines if this set is a superset of another set. Neither set is modified. This set is a superset of otherSet if every element in otherSet is also in this set.
Remarks

IsSupersetOf is computed in time $O(M \log N)$, where $M$ is the size of the otherSet, and $N$ is the size of the this set.

Returns a View collection that can be used for enumerating a range of the items in the set. Only items that are greater than from and less than to are included. The items are enumerated in sorted order. Items equal to the end points of the range can be included or excluded depending on the fromInclusive and toInclusive parameters.

Returns a View collection that can be used for enumerating a range of the items in the set. Only items that are greater than (and optionally, equal to) from are included. The items are enumerated in sorted order. Items equal to from can be included or excluded depending on the fromInclusive parameter.

Returns a View collection that can be used for enumerating a range of the items in the set. Only items that are less than
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RangeTo</strong></td>
<td>(and optionally, equal to) to are included. The items are enumerated in sorted order. Items equal to to can be included or excluded depending on the toInclusive parameter.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Must be overridden to allow removing items from this collection.</td>
</tr>
<tr>
<td><strong>RemoveAll</strong></td>
<td>Removes all the items in the collection that satisfy the condition defined by predicate. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;.</code>)</td>
</tr>
<tr>
<td><strong>RemoveFirst</strong></td>
<td>Removes the first item in the set. This is also the smallest item in the set.</td>
</tr>
<tr>
<td><strong>RemoveLast</strong></td>
<td>Removes the last item in the set. This is also the largest item in the set.</td>
</tr>
<tr>
<td><strong>RemoveMany</strong></td>
<td>Removes all the items in collection from the set. Items not present in the set are ignored.</td>
</tr>
<tr>
<td><strong>Reversed</strong></td>
<td>Returns a View collection that can be used for enumerating the items in the set in reversed order.</td>
</tr>
<tr>
<td><strong>SymmetricDifference</strong></td>
<td>Computes the symmetric difference of this set with another set. The symmetric difference of two sets is all items that appear in either of the sets, but not both. A new set is created with the symmetric difference of the sets and is returned. This set and the other set are unchanged.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SymmetricDifferenceWith</strong></td>
<td>The symmetric difference of two sets is all items that appear in either of the sets, but not both. This set receives the symmetric difference of the two sets; the other set is unchanged.</td>
</tr>
<tr>
<td><strong>ToArray</strong></td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from CollectionBase(Of&lt;T&gt;).)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Shows the string representation of the collection. The string representation contains a list of the items in the collection. Contained collections (except string) are expanded recursively. (Inherited from CollectionBase(Of&lt;T&gt;).)</td>
</tr>
<tr>
<td><strong>TrueForAll</strong></td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from CollectionBase(Of&lt;T&gt;).)</td>
</tr>
<tr>
<td><strong>TryGetItem</strong></td>
<td>Determines if this set contains an item equal to item, according to the comparison mechanism that was used when the set was created. The set is not changed.</td>
</tr>
<tr>
<td></td>
<td>If the set does contain an item equal to item, then the item from the set is returned.</td>
</tr>
<tr>
<td></td>
<td>Computes the union of this set</td>
</tr>
</tbody>
</table>
Union

with another set. The union of two sets is all items that appear in either or both of the sets. A new set is created with the union of the sets and is returned. This set and the other set are unchanged.

Computes the union of this set with another set. The union of two sets is all items that appear in either or both of the sets. This set receives the union of the two sets, the other set is unchanged.

UnionWith
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>)</td>
</tr>
</tbody>
</table>
# Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection(Of T)&gt;::Add</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase(Of T&gt;).)</td>
</tr>
<tr>
<td>ICollection::CopyTo</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from CollectionBase(Of T&gt;.)</td>
</tr>
<tr>
<td>IEnumerable::GetEnumerator</td>
<td>Makes a shallow clone of this set; i.e., if items of the set are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td>ICloneable::Clone</td>
<td></td>
</tr>
</tbody>
</table>


See Also

OrderedSet<Of<T>>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>.AddMany Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds all the items in collection to the set. If the set already contains an item equal to one of the items in collection, that item will be replaced.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public void AddMany(
    IEnumerable<T> collection
)
```

#### Visual Basic (Declaration)

```vbnet
Public Sub AddMany (_
    collection As IEnumerable(Of T) _
)
```

#### Visual C++

```cpp
public:
void AddMany (  
    IEnumerable<T>^ collection
)
```

### Parameters

`collection`  
`IEnumerable<>Of<>T>`  
A collection of items to add to the set.
Remarks

Equality between items is determined by the comparison instance or delegate used to create the set.

Adding the collection takes time $O(M \log N)$, where $N$ is the number of items in the set, and $M$ is the number of items in collection.
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<(Of <T>)>::AsList Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Get a read-only list view of the items in this ordered set. The items in the list are in sorted order, with the smallest item at index 0. This view does not copy any data, and reflects any changes to the underlying OrderedSet.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public IList<T> AsList()
```

**Visual Basic (Declaration)**

```vbnet
Public Function AsList As IList(Of T)
```

**Visual C++**

```c++
public:
    IList<T>^ AsList ()
```

**Return Value**

A read-only IList<T> view onto this OrderedSet.
See Also

**OrderedSet(Of T)** Class  
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:submitmerge@microsoft.com) about this topic to Microsoft.
OrderedSet<Of <T>>...::Clear Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all items from the set.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#
public override sealed void Clear()

Visual Basic (Declaration)
Public Overrides NotOverridable Sub Clear

Visual C++
public:
virtual void Clear () override sealed
Remarks

Clearing the sets takes a constant amount of time, regardless of the number of items in it.
See Also

**OrderedSet(Of T)** Class  
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
OrderedSet(Of T).::Clone Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a shallow clone of this set; i.e., if items of the set are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedSet<T> Clone()

Visual Basic (Declaration)

Public Function Clone As OrderedSet(Of T)

Visual C++

public: 
OrderedSet<T>^ Clone ()

Return Value

The cloned set.
Remarks

Cloning the set takes time $O(N)$, where $N$ is the number of items in the set.
See Also

`OrderedSet<Of `T`)> Class`  
`Wintellect.PowerCollections Namespace`

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>::CloneContents Method

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a deep clone of this set. A new set is created with a clone of each element of this set, by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public OrderedSet<T> CloneContents()
```

**Visual Basic (Declaration)**

```vbnet
Public Function CloneContents As OrderedSet(Of T)
```

**Visual C++**

```cpp
public: 
OrderedSet<T>& CloneContents ()
```

**Return Value**

The cloned set.
Remarks

If T is a reference type, it must implement ICloneable. Otherwise, an InvalidOperationException is thrown.

Cloning the set takes time $O(N \log N)$, where $N$ is the number of items in the set.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>T is a reference type that does not implement <code>ICloneable</code>.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>::Difference Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the difference of this set with another set. The difference of these two sets is all items that appear in this set, but not in otherSet. A new set is created with the difference of the sets and is returned. This set and the other set are unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedSet<T> Difference(
    OrderedSet<T> otherSet
)

Visual Basic (Declaration)

Public Function Difference ( _
    otherSet As OrderedSet(Of T) _
) As OrderedSet(Of T)

Visual C++

public:
OrderedSet<T>^ Difference (  
    OrderedSet<T>^ otherSet
)

Parameters

otherSet
    OrderedSet<Of <T>>
    Set to difference with.

Return Value

The difference of the two sets.
Remarks

The difference of two sets is computed in time $O(M + N \log M)$, where $M$ is the size of the larger set, and $N$ is the size of the smaller set.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet(Of T)>...:::DifferenceWith Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the difference of this set with another set. The difference of these two sets is all items that appear in this set, but not in otherSet. This set receives the difference of the two sets; the other set is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

**C#**

```csharp
public void DifferenceWith(
    OrderedSet<T> otherSet
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub DifferenceWith (_
    otherSet As OrderedSet(Of T) _
)
```

**Visual C++**

```cpp
public:
void DifferenceWith ( 
    OrderedSet<T>* otherSet
)
```

### Parameters

- **otherSet**
  - `OrderedSet<(Of <T>)>`
  - Set to difference with.
Remarks

The difference of two sets is computed in time $O(M + N \log M)$, where $M$ is the size of the larger set, and $N$ is the size of the smaller set.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>..::GetEnumerator Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns an enumerator that enumerates all the items in the set. The items are enumerated in sorted order.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed IEnumerable<T> GetEnumerator()

Visual Basic (Declaration)

Public Overrides NotOverridable Function GetEnumerator As IEnumerable

Visual C++

public: virtual IEnumerable<T> GetEnumerator () override sealed

Return Value

An enumerator for enumerating all the items in the OrderedSet.
Remarks

Typically, this method is not called directly. Instead the "foreach" statement is used to enumerate the items, which uses this method implicitly.

If an item is added to or deleted from the set while it is being enumerated, then the enumeration will end with an InvalidOperationException.

Enumeration all the items in the set takes time $O(N \log N)$, where $N$ is the number of items in the set.
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet(Of T)::.GetFirst Method

See Also

Visual Basic (Declaration) ▶ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the first item in the set: the item that would appear first if the set was enumerated. This is also the smallest item in the set.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public T GetFirst()
```

**Visual Basic (Declaration)**

```vbnet
Public Function GetFirst As T
```

**Visual C++**

```cpp
public:
T GetFirst ()
```

**Return Value**

The first item in the set.
Remarks

GetFirst() takes time $O(\log N)$, where $N$ is the number of items in the set.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>The set is empty.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<(Of <T>)>...::GetLast Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the last item in the set: the item that would appear last if the set was enumerated. This is also the largest item in the set.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public T GetLast()

Visual Basic (Declaration)

Public Function GetLast As T

Visual C++

public:
T GetLast ()

Return Value

The last item in the set.
Remarks

GetLast() takes time $O(\log N)$, where $N$ is the number of items in the set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>The set is empty.</td>
</tr>
</tbody>
</table>
See Also

`OrderedSet(Of T)>` Class  
`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
Get the index of the given item in the sorted order. The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public  int  IndexOf(
    T  item
)

Visual Basic (Declaration)

Public Function IndexOf ( _
    item As T _
) As  Integer

Visual C++

public:
    int  IndexOf ( _
        T  item
    )

Parameters

item
    T
    The item to get the index of.

Return Value

The index of the item in the sorted set, or -1 if the item is not present in the set.
Remarks

Finding the index takes time $O(\log N)$, which $N$ is the number of items in the set.
See Also

OrderedSet<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet(Of `T`)::Intersection Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the intersection of this set with another set. The intersection of two sets is all items that appear in both of the sets. A new set is created with the intersection of the sets and is returned. This set and the other set are unchanged.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedSet<T> Intersection(
    OrderedSet<T> otherSet
)

Visual Basic (Declaration)

Public Function Intersection ( _
    otherSet As OrderedSet(Of T) _
) As OrderedSet(Of T)

Visual C++

public:
    OrderedSet<T>^ Intersection ( _
    OrderedSet<T>^ otherSet
    )

Parameters

otherSet
    OrderedSet<(Of <T>)>  
    Set to intersection with.

Return Value

The intersection of the two sets.
Remarks

When equal items appear in both sets, the intersection will include an arbitrary choice of one of the two equal items.

The intersection of two sets is computed in time $O(N \log M)$, where $M$ is the size of the larger set, and $N$ is the size of the smaller set.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..:::InvalidOperationException</td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

**OrderedSet(Of T)** Class
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
OrderedSet<(Of <T>)>...::IntersectionWith Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the intersection of this set with another set. The intersection of two sets is all items that appear in both of the sets. This set receives the intersection of the two sets, the other set is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public void IntersectionWith(
    OrderedSet<T> otherSet
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub IntersectionWith ( _
    otherSet As OrderedSet(Of T) _
)
```

**Visual C++**

```c++
public:
void IntersectionWith ( 
    OrderedSet<T>^ otherSet
)
```

**Parameters**

`otherSet`  
`OrderedSet(Of T)`  
Set to intersection with.
Remarks

When equal items appear in both sets, the intersection will include an arbitrary choice of one of the two equal items.

The intersection of two sets is computed in time $O(N \log M)$, where $M$ is the size of the larger set, and $N$ is the size of the smaller set.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::InvalidOperationException</td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet<Of <T>> Class  
_Wintellect.PowerCollections_ Namespace

Send [comments](mailto:comments@microsoft.com) about this topic to Microsoft.
OrderedSet(Of <T>)...::IsDisjointFrom Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this set is disjoint from another set. Two sets are disjoint if no item from one set is equal to any item in the other set.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public bool IsDisjointFrom(
    OrderedSet<T> otherSet
)
```

Visual Basic (Declaration)

```vbnet
Public Function IsDisjointFrom ( _
    otherSet As OrderedSet(Of T) _
) As Boolean

Visual C++

```c++
public:
bool IsDisjointFrom ( 
    OrderedSet<T>& otherSet
)
```

Parameters

otherSet

`OrderedSet<Of <T>>`

Set to check disjointness with.

Return Value

True if the two sets are disjoint, false otherwise.
Remarks

The answer is computed in time $O(N \log M)$, where $M$ is the size of the larger set, and $N$ is the size of the smaller set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

**OrderedSet(Of T)** Class
`Wintellect.PowerCollections` Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
OrderedSet<Of <T>>::isEqualTo Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this set is equal to another set. This set is equal to otherSet if they contain the same items.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
- **Syntax**

  **C#**

  ```csharp
  public bool IsEqualTo(
      OrderedSet<T> otherSet
  )
  ```

  **Visual Basic (Declaration)**

  ```vbnet
  Public Function IsEqualTo (_
      otherSet As OrderedSet(Of T) _
  ) As Boolean
  ```

  **Visual C++**

  ```cpp
  public:
  bool IsEqualTo ( 
      OrderedSet<T>^ otherSet
  )
  ```

  **Parameters**

  `otherSet`  
  `OrderedSet<(Of <T>)>`  
  Set to compare to

  **Return Value**

  True if this set is equal to `otherSet`, false otherwise.
Remarks

IsEqualTo is computed in time $O(N)$, where $N$ is the number of items in this set.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet<Of<T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>..::IsProperSubsetOf Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this set is a proper subset of another set. Neither set is modified. This set is a subset of otherSet if every element in this set is also in otherSet. Additionally, this set must have strictly fewer items than otherSet.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public bool IsProperSubsetOf(
    OrderedSet<T> otherSet
)
```

Visual Basic (Declaration)

```vbnet
Public Function IsProperSubsetOf (_
    otherSet As OrderedSet(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
bool IsProperSubsetOf ( _
    OrderedSet<T>* otherSet
)
```

Parameters

otherSet

    OrderedSet<Of<T>>

    Set to compare to.

Return Value

True if this is a proper subset of otherSet.
Remarks

IsSubsetOf is computed in time $O(N \log M)$, where $M$ is the size of the otherSet, and $N$ is the size of the this set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

**OrderedSet<(Of <T>))** Class

**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
OrderedSet<Of <T>>.::IsProperSupersetOf Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this set is a proper superset of another set. Neither set is modified. This set is a proper superset of otherSet if every element in otherSet is also in this set. Additionally, this set must have strictly more items than otherSet.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public bool IsProperSupersetOf(
    OrderedSet<T> otherSet
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function IsProperSupersetOf (_
    otherSet As OrderedSet(Of T) _
) As Boolean
```

**Visual C++**

```cpp
public:
bool IsProperSupersetOf (
    OrderedSet<T>& otherSet
)
```

### Parameters

otherSet
`OrderedSet<Of <T>>`
OrderedSet to compare to.

### Return Value

True if this is a proper superset of otherSet.
Remarks

IsProperSupersetOf is computed in time $O(M \log N)$, where $M$ is the number of unique items in otherSet.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..:::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet(Of Type)::.IsSubsetOf Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this set is a subset of another set. Neither set is modified. This set is a subset of otherSet if every element in this set is also in otherSet.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public bool IsSubsetOf(
    OrderedSet<T> otherSet
)
```

Visual Basic (Declaration)

```vbnet
Public Function IsSubsetOf ( _
    otherSet As OrderedSet(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
    bool IsSubsetOf ( 
        OrderedSet<T>^ otherSet
    )
```

Parameters

otherSet

OrderedSet<(Of <T>)>
Set to compare to.

Return Value

True if this is a subset of otherSet.
Remarks

IsSubsetOf is computed in time $O(N \log M)$, where $M$ is the size of the otherSet, and $N$ is the size of the this set.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..:::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet<(Of T)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<(Of <T>)>..:::IsSupersetOf Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this set is a superset of another set. Neither set is modified. This set is a superset of otherSet if every element in otherSet is also in this set.
Remarks

IsSupersetOf is computed in time $O(M \log N)$, where $M$ is the size of the otherSet, and $N$ is the size of the this set.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public bool IsSupersetOf(
    OrderedSet<T> otherSet
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function IsSupersetOf ( _
    otherSet As OrderedSet(Of T) _
) As Boolean
```

**Visual C++**

```cpp
public:
bool IsSupersetOf ( 
    OrderedSet<T>^ otherSet
)
```

**Parameters**

otherSet

*OrderedSet<Of <T>>*
OrderedSet to compare to.

**Return Value**

True if this is a superset of otherSet.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet(Of <T>)::Range Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a View collection that can be used for enumerating a range of the items in the set. Only items that are greater than from and less than to are included. The items are enumerated in sorted order. Items equal to the end points of the range can be included or excluded depending on the fromInclusive and toInclusive parameters.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public OrderedSet<T>.OrderedSet<Of <T>>..::View Range(  
    T from,
    bool fromInclusive,
    T to,
    bool toInclusive
)
```

Visual Basic (Declaration)

```vbnet
Public Function Range ( _  
    from As T, _  
    fromInclusive As Boolean, _  
    to As T, _  
    toInclusive As Boolean _  
) As OrderedSet<Of <T>>..::View
```

Visual C++

```cpp
public: 
    OrderedSet<Of <T>>..::View^ Range (  
        T from,
        bool fromInclusive,
        T to,
        bool toInclusive
    )
```

Parameters

from

T
The lower bound of the range.

fromInclusive

Boolean
If true, the lower bound is inclusive--items equal to the lower bound will be included in the range. If false, the lower bound is exclusive--items equal to
the lower bound will not be included in the range.

to

T
The upper bound of the range.

toInclusive

Boolean
If true, the upper bound is inclusive--items equal to the upper bound will be included in the range. If false, the upper bound is exclusive--items equal to the upper bound will not be included in the range.

Return Value

An OrderedSet.View of items in the given range.
Remarks

If from is greater than to, the returned collection is empty.

Typically, this method is used in conjunction with a foreach statement. For example:

```csharp
foreach(T item in set.Range(from, true, to, false)) {
    // process item
}
```

If an item is added to or deleted from the set while the View is being enumerated, then the enumeration will end with an InvalidOperationException.

Calling Range does not copy the data in the tree, and the operation takes constant time.
See Also

**OrderedSet(Of T)>** Class
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:) about this topic to Microsoft.
OrderedSet<(Of <T>)>::RangeFrom Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a View collection that can be used for enumerating a range of the items in the set. Only items that are greater than (and optionally, equal to) from are included. The items are enumerated in sorted order. Items equal to from can be included or excluded depending on the fromInclusive parameter.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public OrderedSet<T>.OrderedSet<Of <T>>.:::View RangeFrom(
    T from,
    bool fromInclusive
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function RangeFrom ( _
    from As T, _
    fromInclusive As Boolean _
) As OrderedSet<Of <T>>.:::View
```

**Visual C++**

```cpp
public: 
    OrderedSet<Of <T>>.:::View^ RangeFrom ( 
    T from,
    bool fromInclusive
    )
```

**Parameters**

`from`

`T`

The lower bound of the range.

`fromInclusive`

`Boolean`

If true, the lower bound is inclusive--items equal to the lower bound will be included in the range. If false, the lower bound is exclusive--items equal to the lower bound will not be included in the range.

**Return Value**
An OrderedSet.View of items in the given range.
Remarks

Typically, this method is used in conjunction with a foreach statement. For example:

```csharp
foreach(T item in set.RangeFrom(from, true)) {
    // process item
}
```

If an item is added to or deleted from the set while the View is being enumerated, then the enumeration will end with an InvalidOperationException.

Calling RangeFrom does not copy the data in the tree, and the operation takes constant time.
See Also

`OrderedSet<(Of <T>)>` Class
`Wintellect.PowerCollections` Namespace

Send [comments](mailto:) about this topic to Microsoft.
OrderedSet<Of <T>্>::RangeTo Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a View collection that can be used for enumerating a range of the items in the set. Only items that are less than (and optionally, equal to) to are included. The items are enumerated in sorted order. Items equal to to can be included or excluded depending on the toInclusive parameter.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public OrderedSet<T>.OrderedSet<OF_<T>>::.View RangeTo(T to,
    bool toInclusive
)
```

Visual Basic (Declaration)

```vbnet
Public Function RangeTo ( _
    to As T, _
    toInclusive As Boolean _
) As OrderedSet<OF_<T>>::.View
```

Visual C++

```cpp
public:
    OrderedSet<OF_<T>>::.View^ RangeTo ( T to,
        bool toInclusive
    )
```

Parameters

to

T
The upper bound of the range.

toInclusive

Boolean
If true, the upper bound is inclusive--items equal to the upper bound will be included in the range. If false, the upper bound is exclusive--items equal to the upper bound will not be included in the range.

Return Value
An OrderedSet.View of items in the given range.
Remarks

Typically, this method is used in conjunction with a foreach statement. For example:

```csharp
foreach(T item in set.RangeTo(to, false)) {
    // process item
}
```

If an item is added to or deleted from the set while the View is being enumerated, then the enumeration will end with an InvalidOperationException.

Calling RangeTo does not copy the data in the tree, and the operation takes constant time.
See Also

**OrderedSet<(Of <T>)>** Class

**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
OrderedSet<Of <T>>.::RemoveFirst Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes the first item in the set. This is also the smallest item in the set.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public T RemoveFirst()

Visual Basic (Declaration)

Public Function RemoveFirst As T

Visual C++

public:
T RemoveFirst ()

Return Value

The item that was removed, which was the smallest item in the set.
Remarks

RemoveFirst() takes time $O(\log N)$, where $N$ is the number of items in the set.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::InvalidOperationException</code></td>
<td>The set is empty.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<
(Of <T>)>::RemoveLast Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes the last item in the set. This is also the largest item in the set.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public T RemoveLast()

Visual Basic (Declaration)

Public Function RemoveLast As T

Visual C++

public:
T RemoveLast ()

Return Value

The item that was removed, which was the largest item in the set.
Remarks

RemoveLast() takes time $O(\log N)$, where $N$ is the number of items in the set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>The set is empty.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<(Of <T)>).::RemoveMany Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all the items in collection from the set. Items not present in the set are ignored.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public int RemoveMany(
    IEnumerable<T> collection
)
```

### Visual Basic (Declaration)

```vbnet
Public Function RemoveMany ( 
    collection As IEnumerable(Of T) _
) As Integer
```

### Visual C++

```csharp
public:
    int RemoveMany ( 
        IEnumerable<T>^ collection
    )
```

## Parameters

- **collection**
  - `IEnumerable<(Of <T>)>`
  - A collection of items to remove from the set.

## Return Value

- The number of items removed from the set.
Remarks

Equality between items is determined by the comparison instance or delegate used to create the set.

Removing the collection takes time $O(M \log N)$, where $N$ is the number of items in the set, and $M$ is the number of items in collection.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::ArgumentNullException</code></td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

**OrderedSet(Of T)** Class
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
OrderedSet(Of T).Reversed Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a View collection that can be used for enumerating the items in the set in reversed order.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public OrderedSet<T>.OrderedSet(Of <T>).::View Reversed()

Visual Basic (Declaration)

Public Function Reversed As OrderedSet(Of <T>).::View

Visual C++

public:
OrderedSet(Of <T>).::View^ Reversed()

Return Value

An OrderedSet.View of items in reverse order.
Remarks

Typically, this method is used in conjunction with a foreach statement. For example:

```csharp
Copy Code

foreach(T item in set.Reversed()) {
    // process item
}
```

If an item is added to or deleted from the set while the View is being enumerated, then the enumeration will end with an InvalidOperationException.

Calling Reverse does not copy the data in the tree, and the operation takes constant time.
See Also

OrderedSet<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>...::SymmetricDifference Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the symmetric difference of this set with another set. The symmetric difference of two sets is all items that appear in either of the sets, but not both. A new set is created with the symmetric difference of the sets and is returned. This set and the other set are unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public OrderedSet<T> SymmetricDifference(
    OrderedSet<T> otherSet
)
```

Visual Basic (Declaration)

```vbnet
Public Function SymmetricDifference ( _
    otherSet As OrderedSet(Of T) _
) As OrderedSet(Of T)
```

Visual C++

```cpp
public:
    OrderedSet<T>^ SymmetricDifference ( 
        OrderedSet<T>^ otherSet
    )
```

Parameters

otherSet

`OrderedSet<Of <T>>`

Set to symmetric difference with.

Return Value

The symmetric difference of the two sets.
The symmetric difference of two sets is computed in time $O(M + N \log M)$, where $M$ is the size of the larger set, and $N$ is the size of the smaller set.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>::SymmetricDifferenceWith Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the symmetric difference of this set with another set. The symmetric difference of two sets is all items that appear in either of the sets, but not both. This set receives the symmetric difference of the two sets; the other set is unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public void SymmetricDifferenceWith(
   OrderedSet<T> otherSet
)

Visual Basic (Declaration)

Public Sub SymmetricDifferenceWith ( _
   otherSet As OrderedSet(Of T) _
)

Visual C++

public:
void SymmetricDifferenceWith ( 
   OrderedSet<T>* otherSet
)

Parameters

otherSet
   OrderedSet<(Of <T>)>
   Set to symmetric difference with.
The symmetric difference of two sets is computed in time $O(M + N \log M)$, where $M$ is the size of the larger set, and $N$ is the size of the smaller set.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..:::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private void ICollection<T>.Add(T item)

Visual Basic (Declaration)

Private Sub System.Collections.Generic.ICollection<T>.Add(_
    item As T _)
    Implements ICollection(Of T).Add

Visual C++

private:
    virtual void System.Collections.Generic.ICollection<T>.Add(T item)
    ) sealed = ICollection<T>::Add

Parameters

item
    T
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Makes a shallow clone of this set; i.e., if items of the set are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
private Object ICloneable.Clone()
```

**Visual Basic (Declaration)**

```vbnet
Private Function System.ICloneable.Clone As Object Implements IClone
```

**Visual C++**

```cpp
private:
virtual Object ^ System.ICloneable.Clone () sealed = ICloneable::Clone
```

**Return Value**

The cloned set.
Remarks

Cloning the set takes time $O(N)$, where $N$ is the number of items in the set.
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this set contains an item equal to item, according to the comparison mechanism that was used when the set was created. The set is not changed.

If the set does contain an item equal to item, then the item from the set is returned.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public bool TryGetItem(
    T item,
    out T foundItem
)

Visual Basic (Declaration)

Public Function TryGetItem (_
    item As T,_
    <OutAttribute> ByRef foundItem As T _
) As Boolean

Visual C++

public:
bool TryGetItem ( _
    T item,
    [OutAttribute] T% foundItem
)

Parameters

item
    T
    The item to search for.

foundItem
    T%
    Returns the item from the set that was equal to item.

Return Value

True if the set contains item. False if the set does not contain item.
Remarks

Searching the set for an item takes time $O(\log N)$, where $N$ is the number of items in the set.
Examples

In the following example, the set contains strings which are compared in a case-insensitive manner.

```csharp
OrderedSet<string> set = new OrderedSet<string>(StringComparer.CurrentCultureIgnoreCase);
set.Add("HELLO");
string s;
bool b = set.TryGetItem("Hello", out s);  // b receives true, s receives "HELLO".
```
See Also

**OrderedSet(Of T)** Class
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
OrderedSet<Of <T>>...::Union Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the union of this set with another set. The union of two sets is all items that appear in either or both of the sets. A new set is created with the union of the sets and is returned. This set and the other set are unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public OrderedSet<T> Union(
    OrderedSet<T> otherSet
)
```

Visual Basic (Declaration)

```vbnet
Public Function Union (_
    otherSet As OrderedSet(Of T) _
) As OrderedSet(Of T)
```

Visual C++

```cpp
public:
    OrderedSet<T>^ Union ( 
        OrderedSet<T>^ otherSet
    )
```

Parameters

otherSet

    `OrderedSet<(Of <T>)>`

    Set to union with.

Return Value

The union of the two sets.
Remarks

If equal items appear in both sets, the union will include an arbitrary choice of one of the two equal items.

The union of two sets is computed in time $O(M + N \log M)$, where $M$ is the size of the larger set, and $N$ is the size of the smaller set.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..:::InvalidOperationException</td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

**OrderedSet(Of T)** Class
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
OrderedSet(Of T)::.UnionWith Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the union of this set with another set. The union of two sets is all items that appear in either or both of the sets. This set receives the union of the two sets, the other set is unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public void UnionWith(
    OrderedSet<T> otherSet
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub UnionWith ( _
    otherSet As OrderedSet(Of T) _
)
```

**Visual C++**

```cpp
public:
    void UnionWith (
        OrderedSet<T>^ otherSet
    )
```

#### Parameters

**otherSet**

```csharp
OrderedSet<(Of <T>)>
```

Set to union with.
Remarks

If equal items appear in both sets, the union will include an arbitrary choice of one of the two equal items.

The union of two sets is computed in time $O(M + N \log M)$, where $M$ is the size of the larger set, and $N$ is the size of the smaller set.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet<(Of <T>)) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>?> Properties

See Also

☐ Include Inherited Members ☑ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class OrderedSet<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class OrderedSet(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class OrderedSet
Type Parameters

T

The type exposes the following properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comparer</strong></td>
<td>Returns the IComparer&lt;T&gt; used to compare items in this set.</td>
</tr>
<tr>
<td><strong>Count</strong></td>
<td>Returns the number of items in the set. (Overrides CollectionBase&lt;Of T&gt;::Count.)</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Get the item by its index in the sorted order. The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;of T&gt;::IsReadOnly</code></td>
<td>(Inherited from <code>CollectionBase&lt;of T&gt;::</code>) Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td><code>ICollection::IsSynchronized</code></td>
<td>(Inherited from <code>CollectionBase&lt;of T&gt;::</code>) Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td><code>ICollection::SyncRoot</code></td>
<td>(Inherited from <code>CollectionBase&lt;of T&gt;::</code>)</td>
</tr>
</tbody>
</table>
See Also

OrderedSet<(Of <T>))
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>...::Comparer Property

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the IComparer<T> used to compare items in this set.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public IComparer<T> Comparer{ get; }

Visual Basic (Declaration)

Public ReadOnly PropertyComparer As IComparer(Of T)

Visual C++

public:

property IComparer<T>^Comparer {

IComparer<T>^ get ();
}

Field Value

If the set was created using a comparer, that comparer is returned. If the set was created using a comparison delegate, then a comparer equivalent to that delegate is returned. Otherwise the default comparer for T (Comparer<T>.Default) is returned.
See Also

**OrderedSet(Of T)** Class  
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
OrderedSet<$T>$::Count Property

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the number of items in the set.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public override sealed int Count{ get; }
```

**Visual Basic (Declaration)**

```
Public Overrides NotOverridable ReadOnly Property Count As Integer
```

**Visual C++**

```c++
public:
virtual property int Count {
    int get () override sealed;
}
```

**Field Value**

The number of items in the set.
Remarks

The size of the set is returned in constant time.
See Also

**OrderedSet(Of T)>** Class
**Wintellect.PowerCollections** Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>::Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Get the item by its index in the sorted order. The smallest item has index 0, the next smallest item has index 1, and the largest item has index Count-1.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public T this[
    int index
]{ get;}

Visual Basic (Declaration)

Public ReadOnly Default Property Item ( _
    index As Integer _
) As T

Visual C++

public:
property T default[int index] {
    T get (int index);
}

Parameters

index
    Int32
    The index to get the item by.

Return Value

The item at the given index.
Remarks

The indexer takes time $O(\log N)$, which $N$ is the number of items in the set.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>index is less than zero or greater than or equal to Count.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>.View Class

See Also Members

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

The OrderedSet<Of <T>>.View class is used to look at a subset of the Items inside an ordered set. It is returned from the Range, RangeTo, RangeFrom, and Reversed methods.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

[SerializableAttribute]
public class View : CollectionBase<T>, ICollection<T>, IEnumerable<T>, IEnumerable

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class View _
    Inherits CollectionBase(Of T) _
    Implements ICollection(Of T), IEnumerable(Of T), _
    IEnumerable

Visual C++

[SerializableAttribute]
public ref class View : public CollectionBase<T>, ICollection<T>, IEnumerable<T>, IEnumerable
Remarks

Views are dynamic. If the underlying set changes, the view changes in sync. If a change is made to the view, the underlying set changes accordingly.

Typically, this class is used in conjunction with a foreach statement to enumerate the items in a subset of the OrderedSet. For example:

```csharp
foreach(T item in set.Range(from, to)) {
    // process item
}
```
Inheritance Hierarchy

- System::Object
- Wintellect.PowerCollections::CollectionBase<Of <T>>
- Wintellect.PowerCollections::OrderedSet<Of <T>>::View
See Also

OrderedSet<Of T>...: View Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<\(\text{Of } \langle T \rangle\)>...::View Members

See Also  Methods  Properties  Explicit Interface Implementations

- Include Inherited Members  - Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Syntax

**C#**

```csharp
[SerializableAttribute]
public class View
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute> _
Public Class View
```

**Visual C++**

```cpp
[SerializableAttribute]
public ref class View
```

OrderedSet<T> is a collection that contains items of type T. The item are maintained in a sorted order, and duplicate items are not allowed. Each item has an index in the set: the smallest item has index 0, the next smallest item has index 1, and so forth.

The **OrderedSet(Of T)** generic type exposes the following members.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Must be overridden to allow adding items to this collection. Get a read-only list view of the items in this view. The items in the list are in sorted order, with the smallest item at index 0. This view does not copy any data, and reflects any changes to the underlying OrderedSet.</td>
</tr>
<tr>
<td><strong>AsList</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Removes all the items within this view from the underlying set. (Overrides CollectionBase&lt;Of&lt;T&gt;&gt;::Clear().)</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable&lt;T&gt;.Equals or Object.Equals.</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>ConvertAll&lt;Of&lt;TOutput&gt;&gt;</strong></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate. (Inherited from <strong>CollectionBase&lt;Of &lt;T&gt;&gt;</strong>.)</td>
</tr>
<tr>
<td><strong>FindAll</strong></td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from <strong>CollectionBase&lt;Of &lt;T&gt;&gt;</strong>.)</td>
</tr>
<tr>
<td><strong>ForEach</strong></td>
<td>Performs the specified action on each item in this collection. (Inherited from <strong>CollectionBase&lt;Of &lt;T&gt;&gt;</strong>.)</td>
</tr>
<tr>
<td><strong>GetEnumerator</strong></td>
<td>Enumerate all the items in this view. (Overrides <strong>CollectionBase&lt;Of &lt;T&gt;&gt;</strong>::&lt;GetEnumerator() ()).</td>
</tr>
<tr>
<td><strong>GetFirst</strong></td>
<td>Returns the first item in this view: the item that would appear first if the view was enumerated.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. <strong>GetHashCode()</strong> is suitable for use in hashing algorithms and data structures like a hash table. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetLast</strong></td>
<td>Returns the last item in the view: the item that would appear last if the view was enumerated.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>IndexOf</strong></td>
<td>Get the index of the given item in the view. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count-1.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Must be overridden to allow removing items from this collection.</td>
</tr>
<tr>
<td><strong>RemoveAll</strong></td>
<td>Removes all the items in the collection that satisfy the condition defined by predicate. (Inherited from <strong>CollectionBase&lt;Of &lt;T&gt;&gt;</strong>.)</td>
</tr>
<tr>
<td><strong>Reversed</strong></td>
<td>Creates a new View that has the same items as this view, in the reversed order.</td>
</tr>
<tr>
<td><strong>ToArray</strong></td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.</td>
</tr>
</tbody>
</table>
(Inherited from `CollectionBase<Of<'T>>`.)

**ToString**

Shows the string representation of the collection. The string representation contains a list of the items in the collection.

(Contained collections (except string) are expanded recursively.
(Inherited from `CollectionBase<Of<'T>>`.)

**TrueForAll**

Determines if all of the items in the collection satisfy the condition defined by predicate.
(Inherited from `CollectionBase<Of<'T>>`.)
### Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Count]</td>
<td>Number of items in this view. (Overrides <code>CollectionBase&lt;Of&lt;T&gt;&gt;::Count</code>.)</td>
</tr>
<tr>
<td>![Item]</td>
<td>Get the item by its index in the sorted order. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count-1.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection(Of Of T)&gt;..::Add</td>
<td>(Inherited from CollectionBase(Of Of T)&gt;.)</td>
</tr>
<tr>
<td>ICollection(Of Of T&gt;),..::IsReadOnly</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase(Of Of T&gt;.)</td>
</tr>
<tr>
<td>ICollection,,::CopyTo</td>
<td>Indicates whether the collection is synchronized. (Inherited from CollectionBase(Of Of T&gt;.)</td>
</tr>
<tr>
<td>ICollection,,::IsSynchronized</td>
<td>Indicates the synchronization object for this collection. (Inherited from CollectionBase(Of Of T&gt;.)</td>
</tr>
<tr>
<td>ICollection,,::SyncRoot</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from CollectionBase(Of Of T&gt;.)</td>
</tr>
<tr>
<td>IEnumerable,,::GetEnumerator</td>
<td></td>
</tr>
</tbody>
</table>
See Also

OrderedSet<(Of <T>)>::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>...::View Methods

See Also

☐ Include Inherited Members ☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class View

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class View

Visual C++

[SerializableAttribute]
public ref class View

The OrderedSet(Of T) generic type exposes the following methods.
### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Must be overridden to allow adding items to this collection.</td>
</tr>
<tr>
<td><strong>AsList</strong></td>
<td>Get a read-only list view of the items in this view. The items in the list are in sorted order, with the smallest item at index 0. This view does not copy any data, and reflects any changes to the underlying OrderedSet.</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all the items within this view from the underlying set. (Overrides CollectionBase&lt;Of&lt;T&gt;&gt;.::.Clear().)</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable&lt;T&gt;.Equals or Object.Equals.</td>
</tr>
<tr>
<td><strong>ConvertAll&lt;Of&lt;TOutput&gt;&gt;</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>CountWhere</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>FindAll</strong></td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>ForEach</strong></td>
<td>Performs the specified action on each item in this collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>GetEnumerator</strong></td>
<td>Enumerate all the items in this view.</td>
</tr>
<tr>
<td></td>
<td>(Overrides CollectionBase&lt;Of &lt;T&gt;&gt;...::GetEnumerator())</td>
</tr>
<tr>
<td><strong>GetFirst</strong></td>
<td>Returns the first item in this view: the item that would appear first if the view was enumerated.</td>
</tr>
<tr>
<td></td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>GetHashCode() is suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetLast</strong></td>
<td>Returns the last item in the view: the item that would appear last if the view was enumerated.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td><strong>IndexOf</strong></td>
<td>Get the index of the given item in the view. The smallest item in the view has index 0, the next</td>
</tr>
<tr>
<td></td>
<td>smallest item has index 1, and the largest item has index Count-1.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Must be overridden to allow removing items from this collection.</td>
</tr>
<tr>
<td></td>
<td>Removes all the items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from CollectionBase&lt;Of &lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>RemoveAll</strong></td>
<td>Creates a new View that has the same items as this view, in the reversed order.</td>
</tr>
<tr>
<td><strong>Reversed</strong></td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.</td>
</tr>
</tbody>
</table>
(Inherited from `CollectionBase<Of <T>>`.)

Shows the string representation of the collection. The string representation contains a list of the items in the collection.

**ToString**

Contained collections (except string) are expanded recursively.
(Inherited from `CollectionBase<Of <T>>`.)

**TrueForAll**

Determines if all of the items in the collection satisfy the condition defined by predicate.
(Inherited from `CollectionBase<Of <T>>`.)
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
### Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;Of &lt;T&gt;&gt;::Add</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection::CopyTo</code></td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T)> View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Get a read-only list view of the items in this view. The items in the list are in sorted order, with the smallest item at index 0. This view does not copy any data, and reflects any changes to the underlying OrderedSet.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public IList<T> AsList()

Visual Basic (Declaration)

Public Function AsList As IList(Of T)

Visual C++

public:
IList<T>^ AsList ()

Return Value

A read-only IList<T> view onto this view.
See Also

OrderedSet<Of T> View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>><::View::<::Clear Method
See Also  Example

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all the items within this view from the underlying set.

Namespace:  Wintellect.PowerCollections
Assembly:  PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed void Clear()

Visual Basic (Declaration)

Public Overrides NotOverridable Sub Clear

Visual C++

public:
virtual void Clear () override sealed
Examples

The following removes all the items that start with "A" from an OrderedSet.

```
set.Range("A", "B").Clear();
```
See Also

OrderedSet<OF T>...::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>...::View...::GetEnumerator Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerate all the items in this view.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
- Syntax

**C#**

```csharp
public override sealed IEnumerator<T> GetEnumerator()
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Function GetEnumerator As IEnumerator
```

**Visual C++**

```cpp
public:
virtual IEnumerator<T>^ GetEnumerator() override sealed
```

**Return Value**

An IEnumerator<T> with the items in this view.
See Also

OrderedSet<(Of <T>)>::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>...::View...::GetFirst Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the first item in this view: the item that would appear first if the view was enumerated.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public T GetFirst()
```

**Visual Basic (Declaration)**

```vbnet
Public Function GetFirst As T
```

**Visual C++**

```cpp
public:
T GetFirst ()
```

**Return Value**

The first item in the view.
Remarks

GetFirst() takes time $O(\log N)$, where $N$ is the number of items in the set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::InvalidOperationException</code></td>
<td>The view has no items in it.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet< Of <T>>...::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Returns the last item in the view: the item that would appear last if the view was enumerated.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public T GetLast()
```

**Visual Basic (Declaration)**

```vbnet
Public Function GetLast As T
```

**Visual C++**

```cpp
public:
T GetLast ()
```

**Return Value**

The last item in the view.
Remarks

GetLast() takes time $O(\log N)$, where $N$ is the number of items in the set.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>The view has no items in it.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Get the index of the given item in the view. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count - 1.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public int IndexOf(
    T item
)

Visual Basic (Declaration)

Public Function IndexOf (_
    item As T _
) As Integer

Visual C++

public:
    int IndexOf (
        T item
    )

Parameters

item
    T
    The item to get the index of.

Return Value

The index of the item in the view, or -1 if the item is not present in the view.
Remarks

Finding the index takes time $O(\log N)$, which $N$ is the number of items in the set.
See Also

OrderedSet<Of <T>>...: View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>> View Reversed Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new View that has the same items as this view, in the reversed order.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public OrderedSet<T>.OrderedSet(Of T).::View Reversed()
```

#### Visual Basic (Declaration)

```vbnet
Public Function Reversed As OrderedSet(Of T).::View
```

#### Visual C++

```cpp
public:
OrderedSet(Of T).::View^ Reversed()
```

### Return Value

A new View that has the reversed order of this view, with the same upper and lower bounds.
See Also

OrderedSet(Of T) View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<(Of <T>)...::View...::System.Collections.Generic.ICollection<T>.Add Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
private void ICollection<T>.Add(T item)
```

**Visual Basic (Declaration)**

```vbnet
Private Sub System.Collections.Generic.ICollection<T>.Add(_
    item As T _
) Implements ICollection(Of T).Add
```

**Visual C++**

```cpp
private:
virtual void System.Collections.Generic.ICollection<T>.Add(T item)
sealed = ICollection<T>::Add
```

### Parameters

- **item**: T
See Also

OrderedSet<Of T>::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<Of <T>>...::View Properties

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#  

[SerializableAttribute]  
public class View

Visual Basic (Declaration)  

<SerializableAttribute> _  
Public Class View

Visual C++  

[SerializableAttribute]  
public ref class View

The OrderedSet(Of T) generic type exposes the following properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Number of items in this view. (Overrides CollectionBase&lt;Of &lt;T&gt;&gt;&lt;::Count).</td>
</tr>
<tr>
<td>Item</td>
<td>Get the item by its index in the sorted order. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count-1.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;Of &lt;T&gt;&gt;::IsReadOnly</code></td>
<td>(Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.) Indicates whether the collection is synchronized. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.) Indicates the synchronization object for this collection. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection::IsSynchronized</code></td>
<td></td>
</tr>
<tr>
<td><code>ICollection::SyncRoot</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

OrderedSet<(Of <T>)>::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
OrderedSet<(Of <T>))::.View::.Count Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Number of items in this view.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed int Count{ get;}

Visual Basic (Declaration)

Public Overrides NotOverridable ReadOnly Property Count As Integer

Visual C++

public:
virtual property int Count {
    int get () override sealed;
}

Field Value

Number of items that lie within the bounds the view.
See Also

OrderedSet<Of T>...::View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Get the item by its index in the sorted order. The smallest item in the view has index 0, the next smallest item has index 1, and the largest item has index Count-1.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public T this[
    int index
] { get; }
```

Visual Basic (Declaration)

```vbnet
Public ReadOnly Default Property Item ( _
    index As Integer _
) As T
```

Visual C++

```cpp
public:
property T default[int index] {_
    T get (int index);
}
```

Parameters

index  
Int32  
The index to get the item by.

Return Value

The item at the given index.
Remarks

The indexer takes time $O(\log N)$, which $N$ is the number of items in the set.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>index is less than zero or greater than or equal to Count.</td>
</tr>
</tbody>
</table>
See Also

OrderedSet(Of T) View Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Stores a pair of objects within a single struct. This struct is useful to use as the T of a collection, or as the TKey or TValue of a dictionary.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

[SerializableAttribute]
public struct Pair<TFirst, TSecond> : IComparable,
    IComparable<Pair<TFirst, TSecond>>

Visual Basic (Declaration)

<SerializableAttribute>
Public Structure Pair(Of TFirst, TSecond)
    Implements IComparable, IComparable(Of Pair(Of TFirst, TSecond))

Visual C++

[SerializableAttribute]
generic<typename TFirst, typename TSecond>
public value class Pair : IComparable, IComparable<Pair<TFirst, TSecond>>
Type Parameters

TFirst
TSecond
See Also

`Pair<Of <TFirst, TSecond>>` Members

`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
Pair<Of <TFirst, TSecond>> Members

See Also  Methods  Constructors  Fields  Explicit Interface Implementations

Include Inherited Members  Include Protected Members

.NET Compact Framework Members Only

XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
C#

[SerializableAttribute]
public struct Pair<TFirst, TSecond>

Visual Basic (Declaration)

<SerializableAttribute>
Public Structure Pair(Of TFirst, TSecond)

Visual C++

[SerializableAttribute]
generic< typename TFirst,  typename TSecond>
public value class Pair
Type Parameters

TFirst
TSecond

The type exposes the following members.
**Public Constructors**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Pair&lt;\langle TFirst, TSecond\rangle&gt;</code>&amp; <code>Pair&lt;\langle TFirst, TSecond\rangle&gt;</code> &amp; <code>New</code></td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CompareTo</strong></td>
<td>Compares this pair to another pair of the same type. The pairs are compared by using the IComparable(&lt;T&gt;) or IComparable interface on TFirst and TSecond. The pairs are compared by their first elements first, if their first elements are equal, then they are compared by their second elements.</td>
</tr>
<tr>
<td><strong>Equality</strong></td>
<td>Determines if two pairs are equal. Two pairs are equal if the first and second elements both compare equal using IComparable(&lt;T&gt;).Equals or object.Equals.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>Explicit</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Returns a hash code for the pair, suitable for use in a hash-table or other hashed collection. Two pairs that compare equal (using Equals) will have the same hash code. The hash code for the pair is derived by combining the hash codes for each of the two elements of the pair. (Overrides ValueType..::.GetHashCode().)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Inequality</strong></td>
<td>Determines if two pairs are not equal. Two pairs are equal if the first and second elements both compare equal using IComparable(&lt;T&gt;).Equals or object.Equals.</td>
</tr>
<tr>
<td><strong>ToKeyValuePair</strong></td>
<td>Converts this Pair to a KeyValuePair. The Key part of the KeyValuePair gets the First element, and the Value part of the KeyValuePair gets the Second elements.</td>
</tr>
</tbody>
</table>
`ToString` Returns a string representation of the pair. The string representation of the pair is of the form: First: {0}, Second: {1} where {0} is the result of First.ToString(), and {1} is the result of Second.ToString() (or "null" if they are null.) (Overrides `ValueType::ToString()`.)
# Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
## Public Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>First</strong></td>
<td>The first element of the pair.</td>
</tr>
<tr>
<td><strong>Second</strong></td>
<td>The second element of the pair.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>::CompareTo</code></td>
<td>Compares this pair to another pair of the same type. The pairs are compared by using the <code>IComparable&lt;T&gt;</code> or <code>IComparable</code> interface on <code>TFirst</code> and <code>TSecond</code>. The pairs are compared by their first elements first, if their first elements are equal, then they are compared by their second elements.</td>
</tr>
<tr>
<td><code>IComparable::::CompareTo</code></td>
<td>If either <code>TFirst</code> or <code>TSecond</code> does not implement <code>IComparable&lt;T&gt;</code> or <code>IComparable</code>, then an <code>NotSupportedException</code> is thrown, because the pairs cannot be compared.</td>
</tr>
</tbody>
</table>
See Also

Pair(Of TFirst, TSecond)>  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<(Of <TFirst, TSecond>)> Constructor

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Pair&lt;(Of &lt;TFirst, TSecond&gt;)&gt;Pair&lt;(Of &lt;TFirst, TSecond&gt;)&gt;</code></td>
<td>Creates a new pair using elements from a <code>KeyValuePair</code> structure. The First element gets the Key, and the Second elements gets the Value.</td>
</tr>
<tr>
<td><code>Pair&lt;(Of &lt;TFirst, TSecond&gt;)&gt;Pair&lt;(Of &lt;TFirst, TSecond&gt;)&gt;</code></td>
<td>Creates a new pair with given first and second elements.</td>
</tr>
<tr>
<td><code>New(KeyValuePair&lt;(Of &lt;TFirst, TSecond&gt;)&gt;)</code></td>
<td></td>
</tr>
<tr>
<td><code>New(TFirst, TSecond)</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

Pair<(Of <TFirst, TSecond>> Structure
Pair<(Of <TFirst, TSecond>> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair(Of <TFirst, TSecond>)> Constructor (KeyValuePair(Of <TFirst, TSecond>))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new pair using elements from a KeyValuePair structure. The First element gets the Key, and the Second elements gets the Value.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public Pair(
    KeyValuePair<TFirst, TSecond> keyAndValue
)

Visual Basic (Declaration)

Public Sub New (_
    keyAndValue As KeyValuePair(Of TFirst, TSecond) _
)

Visual C++

public:
    Pair ( 
    KeyValuePair<TFirst, TSecond> keyAndValue
    )

Parameters

keyAndValue
    KeyValuePair<(Of <TFirst, TSecond>)>
    The KeyValuePair to initialize the Pair with.
See Also

Pair(Of TFirst, TSecond)> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<(Of <TFirst, TSecond>)> Constructor (TFirst, TSecond)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new pair with given first and second elements.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

public Pair(
    TFirst first,
    TSecond second
)

**Visual Basic (Declaration)**

Public Sub New (
    first As TFirst, _
    second As TSecond _
)

**Visual C++**

public:
Pair (   TFirst first,
    TSecond second
)

**Parameters**

first
    TFirst
    The first element of the pair.

second
    TSecond
    The second element of the pair.
See Also

Pair<TFirst, TSecond> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<(Of <TFirst, TSecond>)> Fields

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

**C#**

```csharp
[SerializableAttribute]
public struct Pair<TFirst, TSecond>
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute> _
Public Structure Pair(Of TFirst, TSecond)
```

**Visual C++**

```cpp
[SerializableAttribute]
generic<typename TFirst, typename TSecond>
public value class Pair
```
Type Parameters

TFirst
TSecond

The type exposes the following fields.
## Public Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>The first element of the pair.</td>
</tr>
<tr>
<td>Second</td>
<td>The second element of the pair.</td>
</tr>
</tbody>
</table>
See Also

Pair(Of TFirst, TSecond)>  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<(Of <TFirst, TSecond>)>::First Field

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

The first element of the pair.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

public TFirst First

**Visual Basic (Declaration)**

Public First As TFirst

**Visual C++**

public:
TFirst First
See Also

Pair<Of TFirst, TSecond> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<(Of <TFirst, TSecond>)>...: Second Field

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

The second element of the pair.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public TSecond Second

Visual Basic (Declaration)

Public Second As TSecond

Visual C++

public:
TSecond Second
See Also

Pair(Of TFirst, TSecond) Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair(Of TFirst, TSecond) Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

**C#**

```csharp
[SerializableAttribute]
public struct Pair<TFirst, TSecond>
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute> __
Public Structure Pair(Of TFirst, TSecond)
```

**Visual C++**

```c++
[SerializableAttribute]
generic<typename TFirst, typename TSecond>
public value class Pair
```
Type Parameters

TFirst
TSecond

The type exposes the following methods.
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CompareTo</strong></td>
<td>Compares this pair to another pair of the same type. The pairs are compared by using the IComparable&lt;T&gt; or IComparable interface on TFirst and TSecond. The pairs are compared by their first elements first, if their first elements are equal, then they are compared by their second elements. If either TFirst or TSecond does not implement IComparable&lt;T&gt; or IComparable, then an NotSupportedException is thrown, because the pairs cannot be compared.</td>
</tr>
<tr>
<td><strong>Equality</strong></td>
<td>Determines if two pairs are equal. Two pairs are equal if the first and second elements both compare equal using IComparable&lt;T&gt;.Equals or object.Equals.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>Explicit</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Returns a hash code for the pair, suitable for use in a hash-table or other hashed collection. Two pairs that compare equal (using Equals) will have the same hash code. The hash code for the pair is derived by combining the hash codes for each of the two elements of the pair. (Overrides ValueType..::.GetHashCode().)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Inequality</strong></td>
<td>Determines if two pairs are not equal. Two pairs are equal if the first and second elements both compare equal using IComparable&lt;T&gt;.Equals or object.Equals. Converting this Pair to a KeyValuePair. The Key part of the KeyValuePair gets the First element, and the Value part of the KeyValuePair gets the Second elements.</td>
</tr>
<tr>
<td><strong>ToKeyValuePair</strong></td>
<td></td>
</tr>
</tbody>
</table>
ToString

Returns a string representation of the pair. The string representation of the pair is of the form: First: {0}, Second: {1} where {0} is the result of First.ToString(), and {1} is the result of Second.ToString() (or "null" if they are null.)
(Overrides ValueType...::ToString().)
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize</td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
**Explicit Interface Implementations**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IComparable..:::CompareTo</td>
<td>Compares this pair to another pair of the some type. The pairs are compared by using the IComparable&lt;T&gt; or IComparable interface on TFirst and TSecond. The pairs are compared by their first elements first, if their first elements are equal, then they are compared by their second elements.</td>
</tr>
<tr>
<td></td>
<td>If either TFirst or TSecond does not implement IComparable&lt;T&gt; or IComparable, then an NotSupportedException is thrown, because the pairs cannot be compared.</td>
</tr>
</tbody>
</table>
See Also

Pair(Of <TFirst, TSecond>)
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<(Of <TFirst, TSecond>)>::CompareTo Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Compares this pair to another pair of the same type. The pairs are compared by using the IComparable<T> or IComparable interface on TFirst and TSecond. The pairs are compared by their first elements first, if their first elements are equal, then they are compared by their second elements.

If either TFirst or TSecond does not implement IComparable<T> or IComparable, then a NotSupportedException is thrown, because the pairs cannot be compared.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
- Syntax

**C#**

```csharp
public int CompareTo(
    Pair<TFirst, TSecond> other
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function CompareTo (_
    other As Pair(Of TFirst, TSecond) _
) As Integer
```

**Visual C++**

```cpp
public:
virtual int CompareTo (
    Pair<TFirst, TSecond> other
) sealed
```

- **Parameters**

  `other`  
  `Pair<Of <TFirst, TSecond>>`  
  The pair to compare to.

- **Return Value**

  An integer indicating how this pair compares to other. Less than zero indicates this pair is less than other. Zero indicate this pair is equals to other. Greater than zero indicates this pair is greater than other.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::NotSupportedException</code></td>
<td>Either FirstSecond or TSecond is not comparable via the IComparable&lt;T&gt; or IComparable interfaces.</td>
</tr>
</tbody>
</table>
See Also

Pair(Of TFirst, TSecond) Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<Of <TFirst, TSecond>>...::Equality Operator

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if two pairs are equal. Two pairs are equal if the first and second elements both compare equal using IComparable<T>.Equals or object.Equals.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static bool operator ==(
    Pair<TFirst, TSecond> pair1,
    Pair<TFirst, TSecond> pair2
)

Visual Basic (Declaration)

Public Shared Operator = ( _
    pair1 As Pair(Of TFirst, TSecond), _
    pair2 As Pair(Of TFirst, TSecond) _
) As Boolean

Visual C++

public:
static bool operator == ( 
    Pair<TFirst, TSecond> pair1,
    Pair<TFirst, TSecond> pair2
)

Parameters

pair1
    Pair<(Of <TFirst, TSecond>)>  
    First pair to compare.

pair2
    Pair<(Of <TFirst, TSecond>)>  
    Second pair to compare.

Return Value

True if the pairs are equal. False if the pairs are not equal.
See Also

Pair<(Of TFirst, TSecond)> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<Of <TFirst, TSecond>>...::Equals Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals(Object)</strong></td>
<td>Determines if this pair is equal to another object. The pair is equal to another object if that object is a Pair, both element types are the same, and the first and second elements both compare equal using object.Equals. (Overrides <code>ValueType::Equals(Object)</code>.)</td>
</tr>
<tr>
<td><strong>Equals(Pair&lt;TFirst, TSecond&gt;)</strong></td>
<td>Determines if this pair is equal to another pair. The pair is equal if the first and second elements both compare equal using <code>IComparable&lt;T&gt;.Equals</code> or <code>object.Equals</code>.</td>
</tr>
</tbody>
</table>
See Also

Pair(Of {TFirst, TSecond})> Structure
Pair(Of {TFirst, TSecond})> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<(Of <TFirst, TSecond>)>...::Equals Method (Object)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this pair is equal to another object. The pair is equal to another object if that object is a Pair, both element types are the same, and the first and second elements both compare equal using object.Equals.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public override bool Equals(
    Object obj
)
```

Visual Basic (Declaration)

```vbnet
Public Overrides Function Equals ( _
    obj As Object _
) As Boolean
```

Visual C++

```cpp
public:
virtual bool Equals (
    Object^ obj
) override
```

Parameters

obj

`Object`
Object to compare for equality.

Return Value

True if the objects are equal. False if the objects are not equal.
See Also

Pair<(Of TFirst, TSecond)> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this pair is equal to another pair. The pair is equal if the first and second elements both compare equal using IComparable<T>.Equals or object.Equals.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public bool Equals(
        Pair<TFirst, TSecond> other
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function Equals( _
        other As Pair(Of TFirst, TSecond) _
) As Boolean
```

**Visual C++**

```cpp
public:
  bool Equals ( 
        Pair<TFirst, TSecond> other
  )
```

**Parameters**

other

`Pair<((Of <TFirst, TSecond>))>`

Pair to compare with for equality.

**Return Value**

True if the pairs are equal. False if the pairs are not equal.
See Also

Pair(Of TFirst, TSecond>) Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<(Of <TFirst, TSecond>)>...::Explicit Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Explicit(KeyValuePair&lt;Of &lt;TFirst, TSecond&gt;&gt;)</code></td>
<td>Converts a KeyValuePair structure into a Pair. The First element gets the Key, and the Second element gets the Value.</td>
</tr>
<tr>
<td><code>Explicit(Pair&lt;Of &lt;TFirst, TSecond&gt;&gt;)</code></td>
<td>Converts a Pair to a KeyValuePair. The Key part of the KeyValuePair gets the First element, and the Value part of the KeyValuePair gets the Second elements.</td>
</tr>
</tbody>
</table>
See Also

Pair(Of TFirst, TSecond>) Structure
Pair(Of TFirst, TSecond>) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair(Of TFirst, TSecond)>:::Explicit Operator

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Converts a KeyValuePair structure into a Pair. The First element gets the Key, and the Second element gets the Value.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public static explicit operator Pair<TFirst, TSecond> (KeyValuePair<TFirst, TSecond> keyAndValue)
```

#### Visual Basic (Declaration)

```vbnet
Public Shared Narrowing Operator CType ( _
    keyAndValue As KeyValuePair(Of TFirst, TSecond) _
) As Pair(Of TFirst, TSecond)
```

#### Visual C++

```cpp
static explicit operator Pair<TFirst, TSecond> ( KeyValuePair<TFirst, TSecond> keyAndValue)
```

### Parameters

**keyAndValue**

`KeyValuePair<T>(Of TFirst, TSecond)>`  
The KeyValuePair to convert.

### Return Value

The Pair created by converted the KeyValuePair into a Pair.
See Also

Pair(Of TFirst, TSecond)> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<OF TFirst, TSecond>::Explicit Operator

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Converts a Pair to a KeyValuePair. The Key part of the KeyValuePair gets the First element, and the Value part of the KeyValuePair gets the Second elements.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

public static explicit operator KeyValuePair<TFirst, TSecond> (Pair<TFirst, TSecond> pair)

### Visual Basic (Declaration)

Public Shared Narrowing Operator CType ( _
    pair As Pair(Of TFirst, TSecond) _
) As KeyValuePair(Of TFirst, TSecond)

### Visual C++

static explicit operator KeyValuePair<TFirst, TSecond> ( Pair<TFirst, TSecond> pair)

## Parameters

pair

Pair<(Of <TFirst, TSecond>)> Pair to convert.

## Return Value

The KeyValuePair created from pair.
See Also

Pair(Of TFirst, TSecond)> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<(Of <TFirst, TSecond>)>...::GetHashCode Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a hash code for the pair, suitable for use in a hash-table or other hashed collection. Two pairs that compare equal (using Equals) will have the same hash code. The hash code for the pair is derived by combining the hash codes for each of the two elements of the pair.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override int GetHashCode()

Visual Basic (Declaration)

Public Overrides Function GetHashCode As Integer

Visual C++

public:
virtual int GetHashCode () override

Return Value

The hash code.
See Also

Pair<(Of TFirst, TSecond)> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<Of <TFirst, TSecond>>...::Inequality Operator

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if two pairs are not equal. Two pairs are equal if the first and second elements both compare equal using IComparable<T>.Equals or object.Equals.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public static bool operator !=(
    Pair<TFirst, TSecond> pair1,
    Pair<TFirst, TSecond> pair2
)

Visual Basic (Declaration)

Public Shared Operator <> ( _
    pair1 As Pair(Of TFirst, TSecond), _
    pair2 As Pair(Of TFirst, TSecond) _
) As Boolean

Visual C++

public:
static bool operator != ( 
    Pair<TFirst, TSecond> pair1,
    Pair<TFirst, TSecond> pair2
)

Parameters

pair1

    Pair<(Of <TFirst, TSecond>)> 
First pair to compare.

pair2

    Pair<(Of <TFirst, TSecond>)> 
Second pair to compare.

Return Value

True if the pairs are not equal. False if the pairs are equal.
See Also

**Pair(Of TFirst, TSecond)>** Structure  
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
Pair<(Of <TFirst, TSecond>)>...::System.IComparable.CompareTo Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Compares this pair to another pair of the same type. The pairs are compared by using the IComparable<T> or IComparable interface on TFirst and TSecond. The pairs are compared by their first elements first, if their first elements are equal, then they are compared by their second elements.

If either TFirst or TSecond does not implement IComparable<T> or IComparable, then a NotSupportedException is thrown, because the pairs cannot be compared.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private int IComparable.CompareTo(
    Object obj
)

Visual Basic (Declaration)

Private Function System.IComparable.CompareTo ( _
    obj As Object _
) As Integer Implements IComparable.CompareTo

Visual C++

private:
    virtual int System.IComparable.CompareTo ( _
        Object^ obj
    ) sealed = IComparable::CompareTo

Parameters

obj
    Object
    The pair to compare to.

Return Value

An integer indicating how this pair compares to obj. Less than zero indicates this pair is less than obj. Zero indicate this pair is equals to obj. Greater than zero indicates this pair is greater than obj.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentException</code></td>
<td>obj is not of the correct type.</td>
</tr>
<tr>
<td><code>System::NotSupportedException</code></td>
<td>Either FirstSecond or TSecond is not comparable via the IComparable&lt;T&gt; or IComparable interfaces.</td>
</tr>
</tbody>
</table>
See Also

Pair<(Of TFirst, TSecond)> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<Of <TFirst, TSecond>>::.ToKeyValuePair Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Converts this Pair to a KeyValuePair. The Key part of the KeyValuePair gets the First element, and the Value part of the KeyValuePair gets the Second elements.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public KeyValuePair<TFirst, TSecond> ToKeyValuePair()
```

**Visual Basic (Declaration)**

```vbnet
Public Function ToKeyValuePair As KeyValuePair(Of TFirst, TSecond)
```

**Visual C++**

```cpp
public:
KeyValuePair<TFirst, TSecond> ToKeyValuePair ()
```

### Return Value

The `KeyValuePair` created from this Pair.
See Also

Pair<(Of <TFirst, TSecond>)> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Pair<(Of <TFirst, TSecond>)>...::ToString Method

See Also
- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a string representation of the pair. The string representation of the pair is of the form: First: {0}, Second: {1} where {0} is the result of First.ToString(), and {1} is the result of Second.ToString() (or "null" if they are null.)

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override string ToString()

Visual Basic (Declaration)

Public Overrides Function ToString As String

Visual C++

public:
virtual String^ ToString () override

ReturnValue

The string representation of the pair.
See Also

Pair(Of TFirst, TSecond) Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<Of <T>> Class

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

ReadOnlyCollectionBase is a base class that can be used to more easily implement the generic ICollection<T> and non-generic ICollection interfaces for a read-only collection: a collection that does not allow adding or removing elements.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

[SerializableAttribute]
public abstract class ReadOnlyCollectionBase<T> : ICollection<T>, IEnumerable<T>, ICollection, IEnumerable

Visual Basic (Declaration)

<SerializableAttribute> _
Public MustInherit Class ReadOnlyCollectionBase(Of T) _
Implements ICollection(Of T), IEnumerable(Of T), _
ICollection, IEnumerable

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class ReadOnlyCollectionBase abstract : ICollection<T>, IEnumerable<T>, ICollection, IEnumerable
Type Parameters

T

The item type of the collection.
Remarks

To use ReadOnlyCollectionBase as a base class, the derived class must override the Count and GetEnumerator methods.

ICollection<T>.Contains need not be implemented by the derived class, but it should be strongly considered, because the ReadOnlyCollectionBase implementation may not be very efficient.
Inheritance Hierarchy

System...::Object
 Wintellect.PowerCollections...::ReadOnlyCollectionBase<(Of <T)>)
  Wintellect.PowerCollections...::ReadOnlyListBase<(Of <T)>)
  Wintellect.PowerCollections...::ReadOnlyDictionaryBase<(Of <TKey, TValue>)>
  Wintellect.PowerCollections...::ReadOnlyMultiDictionaryBase<(Of <TKey, TValue)>

See Also

ReadOnlyCollectionBase<(Of <T>)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<Of <T>> Members

See Also  Methods  Constructors  Properties  Explicit Interface Implementations

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class ReadOnlyCollectionBase<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public MustInherit Class ReadOnlyCollectionBase(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class ReadOnlyCollectionBase abstract
Type Parameters

T

The type exposes the following members.
## Protected Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ReadOnlyCollectionBase&lt;Of T&gt;</code> &gt; New</td>
<td></td>
</tr>
</tbody>
</table>
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Contains</strong></td>
<td>Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable&lt;T&gt;.Equals or Object.Equals.</td>
</tr>
<tr>
<td><strong>ConvertAll(Of TOutput&gt;)</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.</td>
</tr>
<tr>
<td><strong>CountWhere</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate.</td>
</tr>
<tr>
<td><strong>FindAll</strong></td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td><strong>ForEach</strong></td>
<td>Performs the specified action on each item in this collection.</td>
</tr>
<tr>
<td><strong>GetEnumerator</strong></td>
<td>Must be overridden to enumerate all the members of the collection.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. <strong>GetHashCode()</strong> is suitable for use in hashing algorithms and data structures like a hash table. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

Creates an array of the correct size, and copies all the items.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToArray</td>
<td>in the collection into the array, by calling CopyTo.</td>
</tr>
<tr>
<td>ToString</td>
<td>Shows the string representation of the collection. The string representation contains a list of the items in the collection. (Overrrides <code>Object.ToString()</code>.)</td>
</tr>
<tr>
<td>TrueForAll</td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate.</td>
</tr>
</tbody>
</table>
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Must be overridden to provide the number of items in the collection.</td>
</tr>
</tbody>
</table>
### Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IICollection&lt;Of &lt;T&gt;&gt;::Add</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.</td>
</tr>
<tr>
<td><code>IICollection&lt;Of &lt;T&gt;&gt;::Clear</code></td>
<td>Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td><code>IICollection&lt;Of &lt;T&gt;&gt;::IsReadOnly</code></td>
<td>Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td><code>IICollection&lt;Of &lt;T&gt;&gt;::Remove</code></td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection.</td>
</tr>
<tr>
<td><code>IEnumerable::GetEnumerator</code></td>
<td></td>
</tr>
<tr>
<td><code>IICollection::CopyTo</code></td>
<td></td>
</tr>
<tr>
<td><code>IICollection::IsSynchronized</code></td>
<td></td>
</tr>
<tr>
<td><code>IICollection::SyncRoot</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

ReadOnlyCollectionBase<(Of <T>>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase(Of T>) Constructor

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

protected ReadOnlyCollectionBase()

Visual Basic (Declaration)

Protected Sub New

Visual C++

protected:
ReadOnlyCollectionBase ()
See Also

ReadOnlyCollectionBase(Of <T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase(Of T) Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class ReadOnlyCollectionBase<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public MustInherit Class ReadOnlyCollectionBase(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class ReadOnlyCollectionBase abstract
Type Parameters

T

The type exposes the following methods.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains</td>
<td>Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable{T}.Equals or Object.Equals.</td>
</tr>
<tr>
<td>ConvertAll(Of &lt;TOutput&gt;)</td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.</td>
</tr>
<tr>
<td>CountWhere</td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Exists</td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate.</td>
</tr>
<tr>
<td>FindAll</td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td>ForEach</td>
<td>Performs the specified action on each item in this collection.</td>
</tr>
<tr>
<td>GetEnumerator</td>
<td>Must be overridden to enumerate all the members of the collection.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. GetHashCode() is suitable for use in hashing algorithms and data structures like a hash table. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

Creates an array of the correct size, and copies all the items
| **ToArray** | in the collection into the array, by calling CopyTo. |
| **ToString** | Shows the string representation of the collection. The string representation contains a list of the items in the collection. *(Overrides Object...ToString().)* |
| **TrueForAll** | Determines if all of the items in the collection satisfy the condition defined by predicate. |
### Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize</td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection(Of&lt;T&gt;)::.Add</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.</td>
</tr>
<tr>
<td>ICollection(Of&lt;T&gt;)::.Clear</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection.</td>
</tr>
<tr>
<td>ICollection::.Remove</td>
<td></td>
</tr>
<tr>
<td>ICollection::.CopyTo</td>
<td></td>
</tr>
<tr>
<td>IEnumerable::.GetEnumerator</td>
<td></td>
</tr>
</tbody>
</table>
See Also

ReadOnlyCollectionBase<(Of <T>)>  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<Of <T>>...::Contains Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable<T>.Equals or Object.Equals.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual bool Contains(
    T item
)

Visual Basic (Declaration)

Public Overridable Function Contains ( _
    item As T _
) As Boolean

Visual C++

public:
    virtual bool Contains ( T item
    )

Parameters

item
    T
    The item to check for in the collection.

Return Value

True if the collection contains item, false otherwise.
Remarks

You should strongly consider overriding this method to provide a more efficient implementation.
See Also

ReadOnlyCollectionBase< Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public virtual IEnumerable<TOutput> ConvertAll<TOutput>(
Converter<T, TOutput> converter
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function ConvertAll(Of TOutput) ( _
    converter As Converter(Of T, TOutput) _
) As IEnumerable(Of TOutput)
```

**Visual C++**

```csharp
public:
    generic<
type TOutput>
    virtual IEnumerable<TOutput>^ ConvertAll ( 
        Converter<T, TOutput>^ converter
    )
```

### Parameters

`converter`  
`Converter<(Of <T, TOutput>)>`

A delegate to the method to call, passing each item in this collection.

### Return Value

An `IEnumerable<TOutput>` that enumerates the resulting collection from applying `converter` to each item in this collection in order.
Type Parameters

TOutput
   The type each item is being converted to.
### Exceptions

<table>
<thead>
<tr>
<th><strong>Exception</strong></th>
<th><strong>Condition</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>converter is null.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyCollectionBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase(Of T)::.CopyTo Method

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual void CopyTo(
    T[] array,
    int arrayIndex
)

Visual Basic (Declaration)

Public Overridable Sub CopyTo ( _
    array As T(), _
    arrayIndex As Integer _
)

Visual C++

public:
    virtual void CopyTo ( 
        array<T>^ array, 
        int arrayIndex
    )

Parameters

array
    array<T>[]()
    Array to copy to.

arrayIndex
    Int32
    Starting index in array to copy to.
See Also

ReadOnlyCollectionBase<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<(Of <T>)>...::CountWhere Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Counts the number of items in the collection that satisfy the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public virtual int CountWhere(
    Predicate<T> predicate
)
```

### Visual Basic (Declaration)

```vbnet
Public Overridable Function CountWhere ( _
    predicate As Predicate(Of T) _
) As Integer
```

### Visual C++

```cpp
public:
    virtual int CountWhere ( int predicate
    )
```  

### Parameters

**predicate**

`Predicate<(Of <T>)>`

A delegate that defines the condition to check for.

### Return Value

The number of items in the collection that satisfy predicate.
See Also

ReadOnlyCollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<Of <T>>...::Exists Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if the collection contains any item that satisfies the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual bool Exists(
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function Exists ( _
    predicate As Predicate(Of T) _
) As Boolean
```

**Visual C++**

```cpp
public:
    virtual bool Exists ( 
        Predicate<T>^ predicate
    )
```

**Parameters**

`predicate`  
`Predicate<(Of <T>)>`  
A delegate that defines the condition to check for.

**Return Value**

True if the collection contains one or more items that satisfy the condition defined by `predicate`. False if the collection does not contain an item that satisfies `predicate`. 
See Also

ReadOnlyCollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase(Of T)>...::FindAll Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerates the items in the collection that satisfy the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public IEnumerable<T> FindAll(
    Predicate<T> predicate
)
```

### Visual Basic (Declaration)

```vbnet
Public Function FindAll (_
    predicate As Predicate(Of T) _
) As IEnumerable(Of T)
```

### Visual C++

```cpp
public:
    IEnumerable<T>^ FindAll (  
        Predicate<T>^ predicate
    )
```

## Parameters

**predicate**

`Predicate(Of <T>)`  
A delegate that defines the condition to check for.

## Return Value

An `IEnumerable<T>` that enumerates the items that satisfy the condition.
See Also

ReadOnlyCollectionBase`<Of <T>>` Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Perform `%operands%` on each item in this collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
---Syntax---

**C#**

```csharp
public virtual void ForEach(
    Action<T> action
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Sub ForEach ( action As Action(Of T) )
```

**Visual C++**

```cpp
public:
virtual void ForEach ( 
    Action<T>^ action
)
```

**Parameters**

`action`  

*Action(Of T)*

An Action delegate which is invoked for each item in this collection.
See Also

ReadOnlyCollectionBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<(Of <T>)>.::.GetEnumerator Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Must be overridden to enumerate all the members of the collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public abstract IEnumerator<T> GetEnumerator()

Visual Basic (Declaration)

Public MustOverride Function GetEnumerator As IEnumerator(Of T)

Visual C++

public:
virtual IEnumerator<T>^ GetEnumerator () abstract

Return Value

A generic IEnumerator<T> that can be used to enumerate all the items in the collection.
See Also

ReadOnlyCollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<(Of <T>)>...::System.Collections.Generic.ICollection<T>.Add Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

private void ICollection<T>.Add(
    T item
)

**Visual Basic (Declaration)**

Private Sub System.Collections.Generic.ICollection<T>.Add ( _
    item As T _
) Implements ICollection(Of T).Add

**Visual C++**

private:
    virtual void System.Collections.Generic.ICollection<T>.Add ( T item )
    sealed = ICollection<T>::Add

**Parameters**

item
    T
See Also

ReadOnlyCollectionBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
private void ICollection<T>.Clear()
```

### Visual Basic (Declaration)

```vbnet
```

### Visual C++

```cpp
private:
```
See Also

ReadOnlyCollectionBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase(Of \(T\))\ldots \text{::} \text{System.Collections.Generic.ICollection}\(\langle T\rangle\).Remove Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private bool ICollection<T>.Remove(
    T item
)

Visual Basic (Declaration)

Private Function System.Collections.Generic.ICollection<T>.Remove (  
    item As T  
) As Boolean Implements ICollection(Of T).Remove

Visual C++

private:
virtual bool System.Collections.Generic.ICollection<T>.Remove (  
    T item
) sealed = ICollection<T>::Remove

Parameters

item
    T
See Also

ReadOnlyCollectionBase<(Of '<T>')> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase(Of T)::.System.Collections.ICollection.CopyTo Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
private void ICollection.CopyTo(
    Array array,
    int index
)
```

**Visual Basic (Declaration)**

```vbnet
Private Sub System.Collections.ICollection.CopyTo ( _
    array As System.Collections.ICollection.CopyTo (_
    index As Integer _
) Implements System.Collections.ICollection.CopyTo
```

**Visual C++**

```cpp
private:
virtual void System.Collections.ICollection.CopyTo ( _
    Array array,
    int index
) sealed = ICollection::CopyTo
```

**Parameters**

- **array**
  - `Array` — Array to copy to.

- **index**
  - `Int32` — Starting index in array to copy to.
See Also

ReadOnlyCollectionBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Read Only Collection Base

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator<T> that was overridden by the derived classes to enumerate the members of the collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private IEnumerable IEnumerable.GetEnumerator();

Visual Basic (Declaration)

Private Function System.Collections.IEnumerable.GetEnumerator As IEnumerable

Visual C++

private:
virtual IEnumerable^ System.Collections.IEnumerable.GetEnumerator ();

Return Value

An IEnumerator that can be used to iterate the collection.
See Also

ReadOnlyCollectionBase<(Of <T>>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<(Of <T>)>...::ToArray Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public virtual T[] ToArray()
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function ToArray As T()
```

**Visual C++**

```cpp
public:
virtual array<T>^ ToArray ()
```

**Return Value**

An array containing all the elements in the collection, in order.
See Also

ReadOnlyCollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<(Of <T>)>...::ToString Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Shows the string representation of the collection. The string representation contains a list of the items in the collection.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public override string ToString()
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides Function ToString As String
```

**Visual C++**

```cpp
public:
virtual String^ ToString () override
```

**Return Value**

The string representation of the collection.
See Also

ReadOnlyCollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase(Of T) TrueForAll Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if all of the items in the collection satisfy the condition defined by predicate.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```
public virtual bool TrueForAll(
    Predicate<T> predicate
)
```

Visual Basic (Declaration)

```
Public Overridable Function TrueForAll ( _
    predicate As Predicate(Of T) _
) As Boolean
```

Visual C++

```
public:
    virtual bool TrueForAll ( 
        Predicate<T>^ predicate
    )
```

Parameters

predicate

```
    Predicate(Of <T>)
```

A delegate that defines the condition to check for.

Return Value

True if all of the items in the collection satisfy the condition defined by predicate, or if the collection is empty. False if one or more items in the collection do not satisfy predicate.
See Also

ReadOnlyCollectionBase<OF T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<(Of <T>)> Properties

See Also

☑ Include Inherited Members ☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class ReadOnlyCollectionBase<T>

Visual Basic (Declaration)

<SerializableAttribute>
Public MustInherit Class ReadOnlyCollectionBase(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class ReadOnlyCollectionBase abstract
Type Parameters

T

The type exposes the following properties.
# Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>📊 <strong>Count</strong></td>
<td>Must be overridden to provide the number of items in the collection.</td>
</tr>
</tbody>
</table>
# Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection<code>&lt;T&gt;</code>::IsReadOnly</td>
<td>Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td>ICollection::IsSynchronized</td>
<td>Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td>ICollection::SyncRoot</td>
<td></td>
</tr>
</tbody>
</table>
See Also

ReadOnlyCollectionBase<(Of <T>))
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<(Of <T>)>::Count Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Must be overridden to provide the number of items in the collection.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public abstract int Count { get; }
```

**Visual Basic (Declaration)**

```
Public MustOverride ReadOnly Property Count As Integer
```

**Visual C++**

```cpp
public:
virtual property int Count {
    int get () abstract;
}
```

Field Value

The number of items in the collection.
See Also

ReadOnlyCollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<Of
<T>>...: System.Collections.Generic.ICollection<T>.IsReadOnly Property
See Also

[This topic is pre-release documentation and is subject to change in future
releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

private bool ICollection<T>.IsReadOnly{ get; }

**Visual Basic (Declaration)**

Private ReadOnly Property System.Collections.Generic.ICollection<T>.IsReadOnly

**Visual C++**

private:
virtual property bool System.Collections.Generic.ICollection<T>.IsReadOnly

bool get () sealed = ICollection<T>::IsReadOnly::get;
See Also

ReadOnlyCollectionBase<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Indicates whether the collection is synchronized.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private bool ICollection.IsSynchronized { get; }

Visual Basic (Declaration)

Private ReadOnly Property System.Collections.ICollection.IsSynchronized

Visual C++

private:
virtual property bool System.Collections.ICollection.IsSynchronized
    bool get () sealed = ICollection::IsSynchronized::get;

Field Value

Always returns false, indicating that the collection is not synchronized.
See Also

ReadOnlyCollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyCollectionBase<(Of <T>))>...::System.Collections.ICollection.SyncRoot Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Indicates the synchronization object for this collection.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private Object ICollection.SyncRoot { get; }

Visual Basic (Declaration)

Private ReadOnly Property System.Collections.ICollection.SyncRoot As

Visual C++

private:
virtual property Object^ System.Collections.ICollection.SyncRoot {
          Object^ get () sealed = ICollection::SyncRoot::get;
}

Field Value

Always returns this.
See Also

ReadOnlyCollectionBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase(Of <TKey, TValue>) Class

See Also  Members

Syntax

[SerializableAttribute]
public abstract class ReadOnlyDictionaryBase<TKey, TValue> : ReadOnlyCollection<KeyValuePair<TKey, TValue>>, IDictionary<TKey, TValue>, ICollection<KeyValuePair<TKey, TValue>>, IEnumerable<KeyValuePair<TKey, TValue>>, IReadOnlyList<KeyValuePair<TKey, TValue>>, IReadOnlyDictionary<TKey, TValue>, IEnumerable

Visual Basic (Declaration)

<SerializableAttribute>
Public MustInherit Class ReadOnlyDictionaryBase(Of TKey, TValue)  Inherits ReadOnlyCollectionBase(Of KeyValuePair(Of TKey, TValue)) Implements IDictionary(Of TKey, TValue), ICollection(Of KeyValuePair(Of TKey, TValue)), IEnumerable<KeyValuePair(Of TKey, TValue)>, IReadOnlyList<KeyValuePair(Of TKey, TValue)>, IReadOnlyDictionary(Of TKey, TValue), IEnumerable

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class ReadOnlyDictionaryBase abstract : public IReadOnlyList<KeyValuePair<TKey, TValue>>, IReadOnlyDictionary<TKey, TValue>, ICollection<KeyValuePair<TKey, TValue>>, IEnumerable<KeyValuePair<TKey, TValue>>, IDictionary<TKey, TValue>, ICollection, IEnumerable
## Type Parameters

**TKey**
- The key type of the dictionary.

**TValue**
- The value type of the dictionary.
Remarks

To use ReadOnlyDictionaryBase as a base class, the derived class must override Count, TryGetValue, GetEnumerator.
Inheritance Hierarchy

```
System::Object
  Wintellect.PowerCollections::ReadOnlyCollectionBase<(Of
  KeyValuePair<(Of<TKey, TValue>)>)
  Wintellect.PowerCollections::ReadOnlyDictionaryBase<(Of<TKey,
  TValue>)>
```
See Also

ReadOnlyDictionaryBase<"Of<TKey, TValue>> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase(Of TKey, TValue)> Members

See Also  Methods  Properties  Constructors  Explicit Interface Implementations

☐  Include Inherited Members  ☑ Include Protected Members
☐  .NET Compact Framework Members Only
☐  XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
**Syntax**

**C#**

```csharp
[SerializableAttribute]
public abstract class ReadOnlyDictionaryBase<TKey, TValue>
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute>
Public MustInherit Class ReadOnlyDictionaryBase(Of TKey, TValue)
```

**Visual C++**

```cpp
[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class ReadOnlyDictionaryBase abstract
```
Type Parameters

TKey
TValue

The type exposes the following members.
## Protected Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ReadOnlyDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)</code></td>
<td><code>ReadOnlyDictionaryBase&lt;(Of &lt;TKey, TValue&gt;)</code> New</td>
</tr>
</tbody>
</table>
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Contains</code></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><code>ContainsKey</code></td>
<td>Determines whether a given key is found in the dictionary.</td>
</tr>
<tr>
<td><code>ConvertAll(Of &lt;TOutput&gt;?)</code></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from <code>ReadOnlyCollectionBase(Of &lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><code>CopyTo</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from <code>GetEnumerator</code> to get all the items and copy them to the provided array. (Inherited from <code>ReadOnlyCollectionBase(Of &lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><code>CountWhere</code></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from <code>ReadOnlyCollectionBase(Of &lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><code>Equals</code></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>Exists</code></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate. (Inherited from <code>ReadOnlyCollectionBase(Of &lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><code>FindAll</code></td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from <code>ReadOnlyCollectionBase(Of &lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><code>ForEach</code></td>
<td>Performs the specified action on each item in this collection. (Inherited from <code>ReadOnlyCollectionBase(Of &lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><code>GetEnumerator</code></td>
<td>Must be overridden to enumerate all the members of the collection. (Inherited from <code>ReadOnlyCollectionBase(Of &lt;T&gt;)</code>.)</td>
</tr>
<tr>
<td><code>GetHashCode()</code></td>
<td>Serve as a hash function for a particular type. <code>GetHashCode()</code> is suitable for use in hashing algorithms.</td>
</tr>
</tbody>
</table>
- **GetHashCode** and data structures like a hash table.
  (Inherited from **Object**.)

- **GetType**
  Gets the **Type** of the current instance.
  (Inherited from **Object**.)

- **Remove**
  Removes a key from the dictionary. Always throws an exception indicating that this method is not supported in a read-only dictionary.

- **ToArray**
  Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.
  (Inherited from **ReadOnlyCollectionBase**<**(Of **<T>**))>.)

- **ToString**
  Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.
  (Overrides **ReadOnlyCollectionBase**<**(Of **<KeyValuePair**<**(Of **<TKey>**, **TValue>**))>)>**::ToString()**.)

- **TrueForAll**
  Determines if all of the items in the collection satisfy the condition defined by predicate.
  (Inherited from **ReadOnlyCollectionBase**<**(Of **<T>**))>.)

- **TryGetValue**
  Determines if this dictionary contains a key equal to key. If so, the value associated with that key is returned through the value parameter. This method must be overridden in the derived class.
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
# Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Must be overridden to provide the number of items in the collection. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td>Item</td>
<td>The indexer of the dictionary. The set accessor throws an NotSupportedException stating the dictionary is read-only.</td>
</tr>
<tr>
<td>Keys</td>
<td>Returns a collection of the keys in this dictionary.</td>
</tr>
<tr>
<td>Values</td>
<td>Returns a collection of the values in this dictionary. The ordering of values in this collection is the same as that in the Keys collection.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection(Of <code>&lt;T&gt;</code>),::Add</td>
<td>(Inherited from ReadOnlyCollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>ICollection(Of <code>&lt;T&gt;</code>),::Clear</td>
<td>(Inherited from ReadOnlyCollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>ICollection(Of <code>&lt;T&gt;</code>),::IsReadOnly</td>
<td>(Inherited from ReadOnlyCollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>ICollection(Of <code>&lt;T&gt;</code>),::Remove</td>
<td>(Inherited from ReadOnlyCollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>IDictionary(Of <code>&lt;TKey, TValue&gt;</code>),::Add</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from ReadOnlyCollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>ICollection,::CopyTo</td>
<td>Indicates whether the collection is synchronized. (Inherited from ReadOnlyCollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>ICollection,::IsSynchronized</td>
<td>Indicates the synchronization object for this collection. (Inherited from ReadOnlyCollectionBase(Of <code>&lt;T&gt;</code>).)</td>
</tr>
<tr>
<td>ICollection,::SyncRoot</td>
<td>Adds a key-value pair to the collection.</td>
</tr>
<tr>
<td>IDictionary,::Add</td>
<td>Always throws an exception indicating that this method is not supported in a read-only dictionary.</td>
</tr>
<tr>
<td>IDictionary,::Clear</td>
<td>Clears this dictionary. Always throws an exception indicating that this method is not supported in a read-only dictionary.</td>
</tr>
</tbody>
</table>
IDictionary::Contains

Determines if this dictionary contains a key equal to key. The dictionary is not changed. Calls the (overridden) ContainsKey method. If key is not of the correct TKey for the dictionary, false is returned.

Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method.

IDictionary::GetEnumerator

Returns whether this dictionary is fixed size.

IDictionary::IsFixedSize

Returns if this dictionary is read-only.

IDictionary::IsReadOnly

Gets the value associated with a given key. When getting a value, if this key is not found in the collection, then null is returned. If the key is not of the correct type for this dictionary, null is returned.

Returns a collection of all the keys in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method.

IDictionary::Keys

Removes the key (and associated value) from the collection that is equal to the passed in key. Always throws an exception indicating that this method is not supported in a read-only dictionary.

IDictionary::Remove

Returns a collection of all the values in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method.

IDictionary::Values

IEnumerable::GetEnumerator

Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method.
See Also

ReadOnlyDictionaryBase(Of TKey, TValue)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase(Of TKey, TValue) Constructor

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections

Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

protected ReadOnlyDictionaryBase();

Visual Basic (Declaration)

Protected Sub New

Visual C++

protected:
ReadOnlyDictionaryBase();
See Also

**ReadOnlyDictionaryBase(Of TKey, TValue)** Class  
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
ReadOnlyDictionaryBase<Of <TKey, TValue>> Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class ReadOnlyDictionaryBase<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute> _
Public MustInherit Class ReadOnlyDictionaryBase(Of TKey, TValue)

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class ReadOnlyDictionaryBase abstract
Type Parameters

TKey
TValue

The type exposes the following methods.
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Contains</code></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><code>ContainsKey</code></td>
<td>Determines whether a given key is found in the dictionary.</td>
</tr>
</tbody>
</table>
| `ConvertAll<(<TOutput>)>` | Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from `ReadOnlyCollectionBase<(<T>)>`.)
| `CopyTo`            | Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from `ReadOnlyCollectionBase<(<T>)>`.)
| `CountWhere`        | Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from `ReadOnlyCollectionBase<(<T>)>`.)
| `Equals`            | Determines whether the specified `Object` is equal to the current `Object`. (Inherited from `Object`.)                                           |
| `Exists`            | Determines if the collection contains any item that satisfies the condition defined by predicate. (Inherited from `ReadOnlyCollectionBase<(<T>)>`.)
| `FindAll`           |Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from `ReadOnlyCollectionBase<(<T>)>`.)
| `ForEach`           | Performs the specified action on each item in this collection. (Inherited from `ReadOnlyCollectionBase<(<T>)>`.)
| `GetEnumerator`    | Must be overridden to enumerate all the members of the collection. (Inherited from `ReadOnlyCollectionBase<(<T>)>`.)
| `GetHashCode()`     | Serves as a hash function for a particular type. `GetHashCode()` is suitable for use in hashing algorithms |
- **GetHashCode** and data structures like a hash table.  
  (Inherited from **Object**.)

- **GetType**  
  Gets the **Type** of the current instance.  
  (Inherited from **Object**.)

- **Remove**  
  Removes a key from the dictionary. Always throws an exception indicating that this method is not supported in a read-only dictionary.

- **ToArray**  
  Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.  
  (Inherited from **ReadOnlyCollectionBase**<**T**>.)

- **ToString**  
  Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.  
  (Overrides **ReadOnlyCollectionBase**<**T**>..::**ToString**().)

- **TrueForAll**  
  Determines if all of the items in the collection satisfy the condition defined by predicate.  
  (Inherited from **ReadOnlyCollectionBase**<**T**>.)

- **TryGetValue**  
  Determines if this dictionary contains a key equal to key. If so, the value associated with that key is returned through the value parameter. This method must be overridden in the derived class.
# Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;Of&lt;T&gt;&gt;::Add</code></td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;Of&lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection&lt;Of&lt;T&gt;&gt;::Clear</code></td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;Of&lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection&lt;Of&lt;T&gt;&gt;::Remove</code></td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;Of&lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>IDictionary&lt;Of&lt;TKey, TValue&gt;&gt;::Add</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>ReadOnlyCollectionBase&lt;Of&lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection::CopyTo</code></td>
<td>Adds a key-value pair to the collection. Always throws an exception indicating that this method is not supported in a read-only dictionary.</td>
</tr>
<tr>
<td><code>IDictionary::Add</code></td>
<td>Clears this dictionary. Always throws an exception indicating that this method is not supported in a read-only dictionary.</td>
</tr>
<tr>
<td><code>IDictionary::Clear</code></td>
<td>Determines if this dictionary contains a key equal to key. The dictionary is not changed. Calls the (overridden) ContainsKey method. If key is not of the correct TKey for the dictionary, false is returned.</td>
</tr>
<tr>
<td><code>IDictionary::Contains</code></td>
<td>Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method.</td>
</tr>
</tbody>
</table>
**IDictionary::Remove**

Removes the key (and associated value) from the collection that is equal to the passed in key. Always throws an exception indicating that this method is not supported in a read-only dictionary.

Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method.

**IEnumerable::GetEnumerator**
See Also

ReadOnlyDictionaryBase(Of TKey, TValue>)
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase<\(\text{Of}\ <\text{TKKey, TValue}>\)>...::Contains Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains(T)</td>
<td>Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable&lt;T&gt;.Equals or Object.Equals. (Inherited from ReadOnlyCollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>Contains(KeyValuePair&lt;Of&lt;TKey, TValue&gt;&gt;&gt;)</td>
<td>Determines if a dictionary contains a given KeyValuePair. This implementation checks to see if the dictionary contains the given key, and if the value associated with the key is equal to (via object.Equals) the value.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyDictionaryBase(Of TKey, TValue) Class
ReadOnlyDictionaryBase(Of TKey, TValue) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if a dictionary contains a given KeyValuePair. This implementation checks to see if the dictionary contains the given key, and if the value associated with the key is equal to (via object.Equals) the value.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
- **Syntax**

  **C#**

  ```csharp
  public virtual bool Contains(
      KeyValuePair<TKey, TValue> item
  )
  ```

  **Visual Basic (Declaration)**

  ```vbnet
  Public Overridable Function Contains ( _
       item As KeyValuePair(Of TKey, TValue) _
  ) As Boolean
  ```

  **Visual C++**

  ```cpp
  public:
  virtual bool Contains ( 
      KeyValuePair<TKey, TValue> item
  )
  ```

  **Parameters**

  item

  `KeyValuePair<(Of <TKey, TValue>)>`

  A KeyValuePair containing the Key and Value to check for.

  **Return Value**
See Also

ReadOnlyDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines whether a given key is found in the dictionary.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
# Syntax

**C#**

```csharp
public virtual bool ContainsKey(
    TKey key
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function ContainsKey ( _
key As TKey _
) As Boolean
```

**Visual C++**

```cpp
public:
virtual bool ContainsKey ( 
    TKey key
)
```

## Parameters

**key**

`TKey`  
Key to look for in the dictionary.

## Return Value

True if the key is present in the dictionary.
Remarks

The default implementation simply calls TryGetValue and returns what it returns.
See Also

ReadOnlyDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Removes a key from the dictionary. Always throws an exception indicating that this method is not supported in a read-only dictionary.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual bool Remove(TKey key)

Visual Basic (Declaration)

Public Overridable Function Remove ( _
    key As TKey _
) As Boolean

Visual C++

public:
virtual bool Remove ( _
    TKey key
)

Parameters

key
    TKey
    Key to remove from the dictionary.

Return Value

True if the key was found, false otherwise.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::NotSupportedException</code></td>
<td>Always thrown.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyDictionaryBase<Of <TKey, TValue>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase<
(Of <TKey,
TValue>)>...: System.Collections.Generic.IDictionary<TKey,TValue>.Add
Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
private void IDictionary<TKey, TValue>.Add(
    TKey key,
    TValue value
)
```

**Visual Basic (Declaration)**

```vbnet
Private Sub System.Collections.Generic.IDictionary<TKey, TValue>.Add(
    key As TKey,
    value As TValue
) Implements IDictionary(Of TKey, TValue).Add
```

**Visual C++**

```cpp
private:
virtual void System.Collections.Generic.IDictionary<TKey,TValue>.Add(
    TKey key,
    TValue value
) sealed = IDictionary<TKey, TValue>::Add
```

**Parameters**

- **key**
  - TKey

- **value**
  - TValue
See Also

ReadOnlyDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase<
(Of <TKey,
TValue>)>...:::System.Collections.IDictionary.Add Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds a key-value pair to the collection. Always throws an exception indicating that this method is not supported in a read-only dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
private void IDictionary.Add(
    Object key,
    Object value
)
```

**Visual Basic (Declaration)**

```vbnet
Private Sub System.Collections.IDictionary.Add (
    key As Object, _
    value As Object _
) Implements IDictionary.Add
```

**Visual C++**

```cpp
private:
virtual void System.Collections.IDictionary.Add ( 
    Object^ key, 
    Object^ value 
) sealed = IDictionary::Add
```

**Parameters**

**key**

- **Object**
  - Key to add to the dictionary.

**value**

- **Object**
  - Value to add to the dictionary.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::NotSupportedException</code></td>
<td>Always thrown.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase(Of TKey, TValue)>...: System.Collections.IDictionary.Clear Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Clears this dictionary. Always throws an exception indicating that this method is not supported in a read-only dictionary.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private void IDictionary.Clear()

Visual Basic (Declaration)

Private Sub System.Collections.IDictionary.Clear Implements IDictionary

Visual C++

private:
virtual void System.Collections.IDictionary.Clear () sealed = IDictionary
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::NotSupportedException</code></td>
<td>Always thrown.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this dictionary contains a key equal to key. The dictionary is not changed. Calls the (overridden) ContainsKey method. If key is not of the correct TKey for the dictionary, false is returned.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
private bool IDictionary.Contains(Object key)
```

**Visual Basic (Declaration)**

```vbnet
Private Function System.Collections.IDictionary.Contains ( _
    key As Object _
) As Boolean Implements IDictionary.Contains
```

**Visual C++**

```cpp
private:
virtual bool System.Collections.IDictionary.Contains ( _
    Object^ key _
) sealed = IDictionary::Contains
```

### Parameters

- **key**
  - `Object`
  - The key to search for.

### Return Value

- True if the dictionary contains key. False if the dictionary does not contain key.
See Also

ReadOnlyDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private IDictionaryEnumerator IDictionary.GetEnumerator()

Visual Basic (Declaration)

Private Function System.Collections.IDictionary.GetEnumerator As IDictionaryEnumerator

Visual C++

private:
virtual IDictionaryEnumerator^ System.Collections.IDictionary.GetEnumerator

Return Value

An enumerator for enumerating all the elements in the OrderedDictionary.
See Also

ReadOnlyDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Removes the key (and associated value) from the collection that is equal to the passed in key. Always throws an exception indicating that this method is not supported in a read-only dictionary.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
- **Syntax**

  **C#**

  private void IDictionary.Remove(
    Object key
  )

  **Visual Basic (Declaration)**

  Private Sub System.Collections.IDictionary.Remove ( _
    key As Object _
  ) Implements IDictionary.Remove

  **Visual C++**

  private:
  virtual void System.Collections.IDictionary.Remove ( _
    Object^ key
  ) sealed = IDictionary::Remove

  **Parameters**

  key
    Object
    The key to remove.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::NotSupportedException</code></td>
<td>Always thrown.</td>
</tr>
</tbody>
</table>
See Also

&ensp;ReadOnlyDictionaryBase<(Of &LT;TKey, TValue&GT;)> Class
&ensp;Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase<
(TKey, TValue)>::System.Collections.IEnumerable.GetEnumerator Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns an enumerator that enumerates all the entries in the dictionary. Each entry is returned as a DictionaryEntry. The entries are enumerated in the same orders as the (overridden) GetEnumerator method.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
private IEnumerator IEnumerable.GetEnumerator()
```

**Visual Basic (Declaration)**

```vbnet
Private Function System.Collections.IEnumerable.GetEnumerator As IEnumerable
```

**Visual C++**

```cpp
private: 
virtual IEnumerable^ System.Collections.IEnumerable.GetEnumerator ()
```

**Return Value**

An enumerator for enumerating all the elements in the OrderedDictionary.
See Also

ReadOnlyDictionaryBase<,(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override string ToString()

Visual Basic (Declaration)

Public Overrides Function ToString As String

Visual C++

public:
virtual String^ ToString () override

ReturnValue

The string representation of the dictionary.
See Also

ReadOnlyDictionaryBase<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determine if this dictionary contains a key equal to key. If so, the value associated with that key is returned through the value parameter. This method must be overridden in the derived class.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

C#

```csharp
public abstract bool TryGetValue(
    TKey key,
    out TValue value
)
```

**Visual Basic (Declaration)**

```vbnet
Public MustOverride Function TryGetValue ( _
    key As TKey, _
    _<OutAttribute> ByRef value As TValue _
) As Boolean
```

**Visual C++**

```csharp
public:
    virtual bool TryGetValue ( 
        TKey key,
        [OutAttribute] TValue% value
    ) abstract
```

**Parameters**

**key**

TKey
The key to search for.

**value**

TValue%
Returns the value associated with key, if true was returned.

**Return Value**

True if the dictionary contains key. False if the dictionary does not contain key.
See Also

ReadOnlyDictionaryBase<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase<Of <TKey, TValue>> Properties

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class ReadOnlyDictionaryBase<TKey, TValue>

Visual Basic (Declaration)

<SerializableAttribute> _
Public MustInherit Class ReadOnlyDictionaryBase(Of TKey, TValue)

Visual C++

[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class ReadOnlyDictionaryBase abstract
Type Parameters

TKey
TValue

The type exposes the following properties.
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Must be overridden to provide the number of items in the collection. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;*&gt;</code>.)</td>
</tr>
<tr>
<td>Item</td>
<td>The indexer of the dictionary. The set accessor throws an <code>NotSupportedException</code> stating the dictionary is read-only.</td>
</tr>
<tr>
<td>Keys</td>
<td>Returns a collection of the keys in this dictionary.</td>
</tr>
<tr>
<td>Values</td>
<td>Returns a collection of the values in this dictionary. The ordering of values in this collection is the same as that in the Keys collection.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;</code>Of <code>T&gt;</code>::IsReadOnly</td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;</code>Of <code>T&gt;</code>.) Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td><code>ICollection</code>::IsSynchronized</td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;</code>Of <code>T&gt;</code>.) Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td><code>ICollection</code>::SyncRoot</td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;</code>Of <code>T&gt;</code>.)</td>
</tr>
<tr>
<td><code>IDictionary</code>::IsFixedSize</td>
<td>Returns whether this dictionary is fixed size.</td>
</tr>
<tr>
<td><code>IDictionary</code>::IsReadOnly</td>
<td>Returns if this dictionary is read-only.</td>
</tr>
<tr>
<td><code>IDictionary</code>::Item</td>
<td>Gets the value associated with a given key. When getting a value, if this key is not found in the collection, then null is returned. If the key is not of the correct type for this dictionary, null is returned.</td>
</tr>
<tr>
<td><code>IDictionary</code>::Keys</td>
<td>Returns a collection of all the keys in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method.</td>
</tr>
<tr>
<td><code>IDictionary</code>::Values</td>
<td>Returns a collection of all the values in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyDictionaryBase(Of TKey, TValue)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase<(Of <TKey, TValue>)>::Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

The indexer of the dictionary. The set accessor throws a NotSupportedException stating the dictionary is read-only.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual TValue this[
    TKey key
]{ get; set;}

Visual Basic (Declaration)

Public Overridable Default Property Item ( _
    key As TKey _
) As TValue

Visual C++

public:
    virtual property TValue default[TKey key] { 
    TValue get (TKey key); 
    void set (TKey key, TValue value); 
}

Parameters

key
    TKey
    Key to find in the dictionary.

Return Value

The value associated with the key.
Remarks

The get accessor is implemented by calling TryGetValue.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::NotSupportedException</strong></td>
<td>Always thrown from the set accessor, indicating that the dictionary is read only.</td>
</tr>
<tr>
<td><strong>System.Collections.Generic..::KeyNotFoundException</strong></td>
<td>Thrown from the get accessor if the key was not found.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase<
(Of <TKey, TValue>)>::Keys Property
See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a collection of the keys in this dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual ICollection<TKey> Keys{ get; }
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable ReadOnly Property Keys As ICollection(Of TKey)
```

**Visual C++**

```cpp
public:
virtual property ICollection<TKey>^ Keys {
    ICollection<TKey>^ get ();
}
```

**Field Value**

A read-only collection of the keys in this dictionary.
See Also

ReadOnlyDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase<
(Of <TKey,
TValue>)...::System.Collections.IDictionary.IsFixedSize Property
See Also
[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns whether this dictionary is fixed size.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

C#

```csharp
private bool IDictionary.IsFixedSize { get; }
```

**Visual Basic (Declaration)**

```
Private ReadOnly Property System.Collections.IDictionary.IsFixedSize
```

**Visual C++**

```cpp
private:
virtual property bool System.Collections.IDictionary.IsFixedSize {
    bool get () sealed = IDictionary::IsFixedSize::get;
}
```

**Field Value**

Always returns true.
See Also

ReadOnlyDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase<(Of <TKey, TValue>)>...::System.Collections.IDictionary.IsReadOnly Property

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns if this dictionary is read-only.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private bool IDictionary.IsReadOnly { get; }

Visual Basic (Declaration)

Private ReadOnly Property System.Collections.IDictionary.IsReadOnly

Visual C++

private:
    virtual property bool System.Collections>IDictionary.IsReadOnly { bool get () sealed = IDictionary::IsReadOnly::get; }

Field Value

Always returns true.
See Also

ReadOnlyDictionaryBase(Of KeyValuePair(Of TKey, TValue)) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase<TKey, TValue>...::System.Collections.IDictionary.Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets the value associated with a given key. When getting a value, if this key is not found in the collection, then null is returned. If the key is not of the correct type for this dictionary, null is returned.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private Object IDictionary.Item[
    Object key
]{ get; set;}

Visual Basic (Declaration)

Private Property System.Collections.IDictionary.Item ( _
key As Object _
) As Object Implements IDictionary.Item

Visual C++

private:
virtual property Object^ System.Collections.IDictionary.Item[Object/
    Object^ get (Object^ key) sealed = IDictionary::Item::get;
    void set (Object^ key, Object^ value) sealed = IDictionary::
}

Parameters

key
    Object

Field Value

The value associated with the key, or null if the key was not present.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::NotSupportedException</code></td>
<td>Always thrown from the set accessor, indicating that the dictionary is read only.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyDictionaryBase<(Of <TKey, TValue>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase(Of <TKey, TValue>)...::System.Collections.IDictionary.Keys Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a collection of all the keys in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
private ICollection IDictionary.Keys { get; }
```

**Visual Basic (Declaration)**

```vbnet
Private ReadOnly Property System.Collections.IDictionary.Keys As ICollection
```

**Visual C++**

```cpp
private:
virtual property ICollection^ System.Collections.IDictionary.Keys { 
ICollection^ get () sealed = IDictionary::Keys::get; }
```

**Field Value**

The collection of keys.
See Also

ReadOnlyDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Returns a collection of all the values in the dictionary. The values in this collection will be enumerated in the same order as the (overridden) GetEnumerator method.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private ICollection IDictionary.Values{ get; }

Visual Basic (Declaration)

Private Readonly Property System.Collections.IDictionary.Values As ]

Visual C++

private:
virtual property ICollection^ System.Collections.IDictionary.Values
 ICollection^ get () sealed = IDictionary::Values::get;
}

Field Value

The collection of values.
See Also

ReadOnlyDictionaryBase(Of TKey, TValue)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyDictionaryBase(Of TKey, TValue)>...::Values Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a collection of the values in this dictionary. The ordering of values in this collection is the same as that in the Keys collection.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual ICollection<TValue> Values{ get; }
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable ReadOnly Property Values As ICollection(Of TValue)
```

**Visual C++**

```csharp
public:
    virtual property ICollection<TValue>^ Values {
        ICollection<TValue>^ get ();
    }
```

**Field Value**

A read-only collection of the values in this dictionary.
See Also

ReadOnlyDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase(Of T) Class

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

ReadOnlyListBase is an abstract class that can be used as a base class for a read-only collection that needs to implement the generic IList<T> and non-generic IList collections. The derived class needs to override the Count property and the get part of the indexer. The implementation of all the other methods in IList<T> and IList are handled by ListBase.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
[SerializableAttribute]
public abstract class ReadOnlyListBase<T> : ReadOnlyCollectionBase<T>, ICollection<T>, IEnumerable<T>, IList, ICollection, IEnumerable
```

Visual Basic (Declaration)

```vbnet
<SerializableAttribute> _
Public MustInherit Class ReadOnlyListBase(Of T) _
    Inherits ReadOnlyCollectionBase(Of T) _
    Implements IList(Of T), ICollection(Of T), _
     IEnumerable(Of T), IList, ICollection, IEnumerable
```

Visual C++

```cpp
[SerializableAttribute]
generic<typename T>
public ref class ReadOnlyListBase abstract : public ReadOnlyCollection<T>, IList<T>, ICollection<T>, IEnumerable<T>, IList, ICollection, IEnumerable
```
Type Parameters

T
Inheritance Hierarchy

System...:::Object
Wintellect.PowerCollections...:::ReadOnlyCollectionBase<Of <T> >
Wintellect.PowerCollections...:::ReadOnlyListBase<Of <T> >
See Also

ReadOnlyListBase(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of <T>> Members

See Also  Methods  Properties  Constructors  Explicit Interface Implementations

☐ Include Inherited Members  ☐ Include Protected Members
☒ .NET Compact Framework Members Only
☒ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class ReadOnlyListBase<T>

Visual Basic (Declaration)

<SerializableAttribute>
Public MustInherit Class ReadOnlyListBase(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class ReadOnlyListBase abstract
Type Parameters

T

The type exposes the following members.
## Protected Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ReadOnlyListBase&lt;Of 'T&gt;</code>&gt; <code>ReadOnlyListBase&lt;Of 'T&gt;</code> New</td>
<td></td>
</tr>
</tbody>
</table>
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Contains</strong></td>
<td>Determines if the list contains any item that compares equal to item. The implementation simply checks whether IndexOf(item) returns a non-negative value. (Overrides <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;::Contains(T).</code>)</td>
</tr>
<tr>
<td><strong>ConvertAll&lt;Of &lt;TOutput&gt;&gt;</strong></td>
<td>Converts this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>. )</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>CountWhere</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>. )</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>. )</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>. )</td>
</tr>
<tr>
<td><strong>Find</strong></td>
<td>Finds the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned.</td>
</tr>
<tr>
<td><strong>FindAll</strong></td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>. )</td>
</tr>
<tr>
<td><strong>FindIndex</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>FindLast</strong></td>
<td>Finds the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned.</td>
</tr>
<tr>
<td><strong>FindLastIndex</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ForEach</strong></td>
<td>Performs the specified action on each item in this collection. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>GetEnumerator</strong></td>
<td>Enumerates all of the items in the list, in order. The item at index 0 is enumerated first, then the item at index 1, and so on. (Overrides <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>::<code>GetEnumerator()</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. Gets hash code of particular type. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>IndexOf</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>LastIndexOf</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>Range</strong></td>
<td>Returns a view onto a sub-range of this list. Items are not copied; the returned <code>IList&lt;T&gt;</code> is simply a different view onto the same underlying items.</td>
</tr>
<tr>
<td><strong>ToArray</strong></td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Shows the string representation of the collection. The string representation contains a list of the items in the collection. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>TrueForAll</strong></td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>TryFind</strong></td>
<td>Finds the first item in the list that satisfies the condition defined by predicate.</td>
</tr>
<tr>
<td><strong>TryFindLast</strong></td>
<td>Finds the last item in the list that satisfies the condition defined by predicate.</td>
</tr>
</tbody>
</table>
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>The property must be overridden by the derived class to return the number of items in the list. (Overrides <code>ReadOnlyCollectionBase&lt;T&gt;::Count</code>.)</td>
</tr>
<tr>
<td>Item</td>
<td>The get part of the indexer must be overridden by the derived class to get values of the list at a particular index.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `ICollection<(Of T)>::Add` | (Inherited from `ReadOnlyCollectionBase<(Of T)>`.)
| `ICollection<(Of T)>::Clear` | (Inherited from `ReadOnlyCollectionBase<(Of T)>`.)
| `ICollection<(Of T)>::IsReadOnly` | (Inherited from `ReadOnlyCollectionBase<(Of T)>`.)
| `ICollection<(Of T)>::Remove` | (Inherited from `ReadOnlyCollectionBase<(Of T)>`.)
| `IList<(Of T)>::Insert` | Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from `ReadOnlyCollectionBase<(Of T)>`.)
| `IList<(Of T)>::RemoveAt` | Indicates whether the collection is synchronized. (Inherited from `ReadOnlyCollectionBase<(Of T)>`.)
| `ICollection::CopyTo` | Indicates the synchronization object for this collection. (Inherited from `ReadOnlyCollectionBase<(Of T)>`.)
| `ICollection::IsSynchronized` | Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator<T> that was overridden by the derived classes to enumerate the members of the collection. (Inherited from `ReadOnlyCollectionBase<(Of T)>`.)
| `IEnumerable::GetEnumerator` |
`ReadOnlyCollectionBase<Of <T>>` adds an item to the end of the list. This implementation throws a `NotSupportedException` indicating that the list is read-only.

`IList::Add` removes all the items from the list, resulting in an empty list. This implementation throws a `NotSupportedException` indicating that the list is read-only.

`IList::Clear` determines if the list contains any item that compares equal to value.

`IList::Contains` finds the first occurrence of an item equal to value in the list, and returns the index of that item.

`IList::IndexOf` inserts a new item at the given index. This implementation throws a `NotSupportedException` indicating that the list is read-only.

`IList::Insert` returns whether the list is a fixed size. This implementation always returns true.

`IList::IsFixedSize` returns whether the list is read only. This implementation always returns true.

`IList::IsReadOnly` gets or sets the value at a particular index in the list.

`IList::Item` searches the list for the first item that compares equal to value. If one is found, it is removed. Otherwise, the list is unchanged. This implementation throws a `NotSupportedException` indicating that the list is read-only.

`IList::Remove` removes the item at the given index. This implementation throws a `NotSupportedException` indicating that the list is read-only.

`IList::RemoveAt`
See Also

ReadOnlyListBase(Of T)
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase(Of Of T) Constructor

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

protected ReadOnlyListBase()

Visual Basic (Declaration)

Protected Sub New

Visual C++

protected:
ReadOnlyListBase ()
**See Also**

ReadOnlyListBase<Of T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase(Of T) Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class ReadOnlyListBase<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public MustInherit Class ReadOnlyListBase(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class ReadOnlyListBase abstract
Type Parameters

T

The type exposes the following methods.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Contains</strong></td>
<td>Determines if the list contains any item that compares equal to item. The implementation simply checks whether IndexOf(item) returns a non-negative value. (Overrides <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt; :: Contains(T)</code>.)</td>
</tr>
<tr>
<td><strong>ConvertAll&lt;Of &lt;TOutput&gt;&gt;</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>CountWhere</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>Find</strong></td>
<td>Finds the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned.</td>
</tr>
<tr>
<td><strong>FindAll</strong></td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>FindIndex</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>FindLast</strong></td>
<td>Finds the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned.</td>
</tr>
<tr>
<td><strong>FindLastIndex</strong></td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
- **ForEach** performs the specified action on each item in this collection. (Inherited from `ReadOnlyCollectionBase<Of <T>>`.)

Enumerates all of the items in the list, in order. The item at index 0 is enumerated first, then the item at index 1, and so on.

- **GetEnumerator** enumerates all of the items in the list, in order. The item at index 0 is enumerated first, then the item at index 1, and so on. (Overrides `ReadOnlyCollectionBase<Of <T>>`...::GetEnumerator().)

Serves as a hash function for a particular type.

- **GetHashCode** `GetHashCode()` is suitable for use in hashing algorithms and data structures like a hash table. (Inherited from `Object`.)

- **GetType** gets the `Type` of the current instance. (Inherited from `Object`.)

- **IndexOf** overloaded.

- **LastIndexOf** overloaded.

- **Range** returns a view onto a sub-range of this list. Items are not copied; the returned `IList<T>` is simply a different view onto the same underlying items.

- **ToArray** creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from `ReadOnlyCollectionBase<Of <T>>`.)

- **ToString** shows the string representation of the collection. The string representation contains a list of the items in the collection. (Inherited from `ReadOnlyCollectionBase<Of <T>>`.)

- **TrueForAll** determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from `ReadOnlyCollectionBase<Of <T>>`.)

- **TryFind** finds the first item in the list that satisfies the condition defined by predicate.

- **TryFindLast** finds the last item in the list that satisfies the condition defined by predicate.
# Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;&gt; &lt;T&gt;</code>::Add</td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;&gt; &lt;T&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection&lt;&gt; &lt;T&gt;</code>::Clear</td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;&gt; &lt;T&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection&lt;&gt; &lt;T&gt;</code>::Remove</td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;&gt; &lt;T&gt;</code>.)</td>
</tr>
<tr>
<td><code>IList&lt;&gt; &lt;T&gt;</code>::Insert</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>ReadOnlyCollectionBase&lt;&gt; &lt;T&gt;</code>.)</td>
</tr>
<tr>
<td><code>IList&lt;&gt; &lt;T&gt;</code>::RemoveAt</td>
<td>Adds an item to the end of the list. This implementation throws a NotSupportedException indicating that the list is read-only.</td>
</tr>
<tr>
<td><code>IList&lt;&gt; &lt;T&gt;</code>::Add</td>
<td>Removes all the items from the list, resulting in an empty list. This implementation throws a NotSupportedException indicating that the list is read-only.</td>
</tr>
<tr>
<td><code>IEnumerable&lt;&gt; GetEnumerator</code></td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from <code>ReadOnlyCollectionBase&lt;&gt; &lt;T&gt;</code>.)</td>
</tr>
</tbody>
</table>
**IList::Contains**
Determines if the list contains any item that compares equal to value.
Find the first occurrence of an item equal to value in the list, and returns the index of that item.

**IList::IndexOf**
Insert a new item at the given index. This implementation throws a NotSupportedException indicating that the list is read-only.

**IList::Insert**
Searches the list for the first item that compares equal to value. If one is found, it is removed. Otherwise, the list is unchanged. This implementation throws a NotSupportedException indicating that the list is read-only.

**IList::Remove**
Removes the item at the given index. This implementation throws a NotSupportedException indicating that the list is read-only.

**IList::RemoveAt**
See Also

ReadOnlyListBase(Of T)
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if the list contains any item that compares equal to item. The implementation simply checks whether IndexOf(item) returns a non-negative value.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public override bool Contains(T item)
```

Visual Basic (Declaration)

```vbnet
Public Overrides Function Contains ( _
    item As T _
) As Boolean
```

Visual C++

```cpp
public:
    virtual bool Contains ( T item
    ) override
```

Parameters

item

T

The item to search for.

Return Value

True if the list contains an item that compares equal to item.
Remarks

Equality in the list is determined by the default sense of equality for T. If T implements IComparable<T>, the Equals method of that interface is used to determine equality. Otherwise, Object.Equals is used to determine equality.
See Also

ReadOnlyListBase<of T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of <T>>...::~:CopyTo Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CopyTo(array&lt;T&gt;[])</code></td>
<td>Copies all the items in the list, in order, to array, starting at index 0.</td>
</tr>
<tr>
<td></td>
<td>Copies all the items in the collection into an array. Implementled by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>CopyTo(array&lt;T&gt;[], Int32)</code></td>
<td>Copies a range of elements from the list to array, starting at arrayIndex.</td>
</tr>
<tr>
<td><code>CopyTo(Int32, array&lt;T&gt;[], Int32)</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase(Of T) Class
ReadOnlyListBase(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase(Of T)>...:::CopyTo Method (array<T>[])()

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies all the items in the list, in order, to array, starting at index 0.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual void CopyTo(
    T[] array
)

Visual Basic (Declaration)

Public Overridable Sub CopyTo ( _
    array As T() _
)

Visual C++

public:
virtual void CopyTo ( 
    array<T>^ array
)

Parameters

array
    array<T>[]()
    The array to copy to. This array must have a size that is greater than or equal to Count.
See Also

ReadOnlyListBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<Exercise<

...::CopyTo Method (Int32, array<

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Copies a range of elements from the list to array, starting at arrayIndex.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual void CopyTo(
    int index,
    T[] array,
    int arrayIndex,
    int count
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Sub CopyTo ( _
    index As Integer, _
    array As T(), _
    arrayIndex As Integer, _
    count As Integer _
)
```

**Visual C++**

```cpp
public:
    virtual void CopyTo ( 
        int index,
        array<T>[] array,
        int arrayIndex,
        int count
    )
```

**Parameters**

**index**

`Int32`

The starting index in the source list of the range to copy.

**array**

`array<T>[]()`

The array to copy to. This array must have a size that is greater than or equal to Count + arrayIndex.
arrayIndex
  Int32
  The starting index in array to copy to.

count
  Int32
  The number of items to copy.
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadonlyListBase<Of T>...::Find Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, then the default value for T (null or all-zero) is returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public virtual T Find(
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function Find ( predicate As Predicate(Of T) ) As T
```

**Visual C++**

```cpp
public:
virtual T Find ( Predicate<T>^ predicate
```  

**Parameters**

`predicate`

`Predicate<(Of <T>)>`

A delegate that defined the condition to check for.

**Return Value**

The first item that satisfies the condition `predicate`. If no item satisfies that condition, the default value for `T` is returned.
Remarks

If the default value for T (null or all-zero) matches the condition defined by predicate, and the list might contain the default value, then it is impossible to distinguish the different between finding the default value and not finding any item. To distinguish these cases, use TryFind(Predicate<Of<T>>, T%).
See Also

ReadOnlyListBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace
ReadOnlyListBase<(Of <T>)>..:::TryFind(Predicate<(Of <T>)>, T%)

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of <T> >...::FindIndex Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FindIndex(Predicate&lt;Of&lt;T&gt;&gt;)</code></td>
<td>Finds the index of the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.</td>
</tr>
<tr>
<td><code>FindIndex(Int32, Predicate&lt;Of&lt;T&gt;&gt;)</code></td>
<td>Finds the index of the first item, in the range of items extending from index to the end, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.</td>
</tr>
<tr>
<td><code>FindIndex(Int32, Int32, Predicate&lt;Of&lt;T&gt;&gt;)</code></td>
<td>Finds the index of the first item, in the range of count items starting from index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase<(Of <T>)> Class
ReadOnlyListBase<(Of <T>)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadonlyListBase<Of T>...::FindIndex Method (Predicate<Of T>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the first item in the list that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public virtual int FindIndex(
    Predicate<T> predicate
)
```

### Visual Basic (Declaration)

```vbnet
Public Overridable Function FindIndex ( _
    predicate As Predicate(Of T) _
) As Integer
```

### Visual C++

```cpp
public:
virtual int FindIndex ( _
    Predicate<T>^ predicate
)
```

## Parameters

**predicate**

- `Predicate(Of <T>)`

  A delegate that defined the condition to check for.

## Return Value

The index of the first item that satisfies the condition `predicate`. If no item satisfies that condition, -1 is returned.
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadonlyListBase<Of <T>>::FindIndex Method (Int32, Predicate<Of <T>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the first item, in the range of items extending from index to the end, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public virtual int FindIndex(
    int index,
    Predicate<T> predicate
)
```

Visual Basic (Declaration)

```vbnet
Public Overridable Function FindIndex ( _
    index As Integer, _
    predicate As Predicate(Of T) _
) As Integer
```

Visual C++

```cpp
public:
virtual int FindIndex ( 
    int index, 
    Predicate<T>^ predicate
)
```

Parameters

index

    Int32
    The starting index of the range to check.

predicate

    Predicate<(Of <T>)>
    A delegate that defined the condition to check for.

Return Value

The index of the first item in the given range that satisfies the condition predicate. If no item satisfies that condition, -1 is returned.
See Also

ReadOnlyListBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<
(Of <T>)>

..::FindIndex Method (Int32, Int32,
Predicate<
(Of <T>)>
)

See Also

[This topic is pre-release documentation and is subject to change in future
releases. Blank topics are included as placeholders.]

Finds the index of the first item, in the range of count items starting from index,
that satisfies the condition defined by predicate. If no item matches the
condition, -1 is returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public virtual int FindIndex(
    int index,
    int count,
    Predicate<T> predicate
)
```

Visual Basic (Declaration)

```vbnet
Public Overridable Function FindIndex (_
    index As Integer, _
    count As Integer, _
    predicate As Predicate(Of T) _
) As Integer
```

Visual C++

```cpp
public:
virtual int FindIndex (
    int index,
    int count,
    Predicate<T>& predicate
)
```

Parameters

index

**Int32**
The starting index of the range to check.

count

**Int32**
The number of items in range to check.

predicate

**Predicate<Of <T>>**
A delegate that defined the condition to check for.

**Return Value**

The index of the first item in the given range that satisfies the condition predicate. If no item satisfies that condition, -1 is returned.
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Finds the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, than the default value for T (null or all-zero) is returned.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual T FindLast(
    Predicate<T> predicate
)

Visual Basic (Declaration)

Public Overridable Function FindLast ( _
    predicate As Predicate(Of T) _
) As T

Visual C++

public:
    virtual T FindLast ( 
        Predicate&lt;T&gt;^ predicate
    )

Parameters

predicate
    Predicate(Of &lt;T&gt;)
    A delegate that defined the condition to check for.

Return Value

The last item that satisfies the condition predicate. If no item satisfies that
condition, the default value for T is returned.
Remarks

If the default value for T (null or all-zero) matches the condition defined by predicate, and the list might contain the default value, then it is impossible to distinguish the different between finding the default value and not finding any item. To distinguish these cases, use `TryFindLast(Predicate<Of <T>>, T%)`.
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace
ReadOnlyListBase(Of T)::.TryFindLast(Predicate(Of T), T)

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of <T>>::FindLastIndex Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FindLastIndex(Predicate&lt;Of &lt;T&gt;&gt;)</code></td>
<td>Finds the index of the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.</td>
</tr>
<tr>
<td><code>FindLastIndex(Int32, Predicate&lt;Of &lt;T&gt;&gt;)</code></td>
<td>Finds the index of the last item, in the range of items extending from the beginning of the list to index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.</td>
</tr>
<tr>
<td><code>FindLastIndex(Int32, Int32, Predicate&lt;Of &lt;T&gt;&gt;)</code></td>
<td>Finds the index of the last item, in the range of count items ending at index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase(Of T) Class
ReadOnlyListBase(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Finds the index of the last item in the list that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public virtual int FindLastIndex(
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function FindLastIndex ( _
    predicate As Predicate(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:
    virtual int FindLastIndex (  
        Predicate<T>^ predicate
    )
```

**Parameters**

`predicate`

`Predicate<(Of <T>)>`

A delegate that defined the condition to check for.

**Return Value**

The index of the last item that satisfies the condition `predicate`. If no item satisfies that condition, -1 is returned.
See Also

ReadOnlyListBase<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Finds the index of the last item, in the range of items extending from the beginning of the list to index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public virtual int FindLastIndex(
    int index,
    Predicate<T> predicate
)
```

### Visual Basic (Declaration)

```vbnet
Public Overridable Function FindLastIndex ( _
    index As Integer, _
    predicate As Predicate(Of T) _
) As Integer
```

### Visual C++

```cpp
public:
virtual int FindLastIndex ( _
    int index,
    Predicate<T>^ predicate
)
```

## Parameters

**index**

`Int32`

The ending index of the range to check.

**predicate**

`Predicate<(Of <T>)>`

A delegate that defined the condition to check for.

## Return Value

The index of the last item in the given range that satisfies the condition `predicate`. If no item satisfies that condition, -1 is returned.
See Also

ReadOnlyListBase<Of T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadonlyListBase<(Of <T>)>..::FindLastIndex Method (Int32, Int32, Predicate<(Of <T>)>)

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the last item, in the range of count items ending at index, that satisfies the condition defined by predicate. If no item matches the condition, -1 is returned.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
# Syntax

**C#**

```csharp
public virtual int FindLastIndex(
    int index,
    int count,
    Predicate<T> predicate
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Function FindLastIndex (_
    index As Integer, _
    count As Integer, _
    predicate As Predicate(Of T) _
) As Integer
```

**Visual C++**

```cpp
public:
virtual int FindLastIndex (_
    int index,_
    int count,_
    Predicate<T>^ predicate
)
```

**Parameters**

- **index**
  - `Int32`
  - The ending index of the range to check.

- **count**
  - `Int32`
  - The number of items in range to check.

- **predicate**
  - `Predicate<(Of<T>)>`
A delegate that defined the condition to check for.

**Return Value**

The index of the last item in the given range that satisfies the condition predicate. If no item satisfies that condition, -1 is returned.
See Also

ReadOnlyListBase<T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of <T>>::.GetEnumerator Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerates all of the items in the list, in order. The item at index 0 is enumerated first, then the item at index 1, and so on.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

public override IEnumarator<T> GetEnumerator()

**Visual Basic (Declaration)**

Public Overrides Function GetEnumerator As IEnumarator(Of T)

**Visual C++**

public:
virtual IEnumarator<T>^ GetEnumerator () override

**Return Value**

An IEnumarator<T> that enumerates all the items in the list.
See Also

**ReadOnlyListBase**(Of **T**) Class

**Wintellect.PowerCollections** Namespace

Send [comments about this topic to Microsoft.](mailto:support@microsoft.com)
ReadOnlyListBase<Of <T>>...::IndexOf Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IndexOf(T)</code></td>
<td>Finds the index of the first item in the list that is equal to item.</td>
</tr>
<tr>
<td><code>IndexOf(T, Int32)</code></td>
<td>Finds the index of the first item, in the range of items extending from index to the end, that is equal to item.</td>
</tr>
<tr>
<td><code>IndexOf(T, Int32, Int32)</code></td>
<td>Finds the index of the first item, in the range of count items starting from index, that is equal to item.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase<(Of <T>)> Class
ReadOnlyListBase<(Of <T>)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Finds the index of the first item in the list that is equal to item.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public virtual int IndexOf(
    T item
)
```

Visual Basic (Declaration)

```vbnet
Public Overridable Function IndexOf ( _
    item As T _
) As Integer
```

Visual C++

```cpp
public:
    virtual int IndexOf ( T item
)
```

Parameters

**item**

*T*  
The item to search for.

Return Value

The index of the first item in the list that that is equal to item. If no item is equal to item, -1 is returned.
Remarks

The default implementation of equality for type T is used in the search. This is the equality defined by IComparable<T> or object.Equals.
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Finds the index of the first item, in the range of items extending from index to the end, that is equal to item.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public virtual int IndexOf(
    T item,
    int index
)
```

Visual Basic (Declaration)

```vbnet
Public Overridable Function IndexOf ( _
    item As T, _
    index As Integer _
) As Integer
```

Visual C++

```cpp
public:
    virtual int IndexOf ( int item, int index )
```

Parameters

item

T
The item to search for.

index

Int32
The starting index of the range to check.

Return Value

The index of the first item in the given range that that is equal to item. If no item is equal to item, -1 is returned.
Remarks

The default implementation of equality for type T is used in the search. This is the equality defined by IComparable<T> or object.Equals.
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadonlyListBase<Of T>...::IndexOf Method (T, Int32, Int32)

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the first item, in the range of count items starting from index, that is equal to item.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual int IndexOf(
    T item,
    int index,
    int count
)

Visual Basic (Declaration)

Public Overridable Function IndexOf ( _
    item As T, _
    index As Integer, _
    count As Integer _
) As Integer

Visual C++

public:
    virtual int IndexOf ( 
    T item, 
    int index, 
    int count
)

Parameters

item
    T
    The item to search for.

index
    Int32
    The starting index of the range to check.

count
    Int32
The number of items in range to check.

**Return Value**

The index of the first item in the given range that that is equal to item. If no item is equal to item, -1 is returned.
Remarks

The default implementation of equality for type T is used in the search. This is the equality defined by IComparable<T> or object.Equals.
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<(Of <T>))::LastIndexOf Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LastIndexOf(T)</td>
<td>Finds the index of the last item in the list that is equal to item.</td>
</tr>
<tr>
<td>LastIndexOf(T, Int32)</td>
<td>Finds the index of the last item, in the range of items extending from the beginning of the list to index, that is equal to item.</td>
</tr>
<tr>
<td>LastIndexOf(T, Int32, Int32)</td>
<td>Finds the index of the last item, in the range of count items ending at index, that is equal to item.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase<(Of <T>)> Class
ReadOnlyListBase<(Of <T>)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of <T>>...::LastIndexOf Method (T)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the last item in the list that is equal to item.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual int LastIndexOf(
  T item
)

Visual Basic (Declaration)

Public Overridable Function LastIndexOf ( _
  item As T _
) As Integer

Visual C++

public:
  virtual int LastIndexOf ( 
    T item
  )

Parameters

item
  T
  The item to search for.

Return Value

The index of the last item in the list that that is equal to item. If no item is equal to item, -1 is returned.
Remarks

The default implementation of equality for type T is used in the search. This is the equality defined by IComparable<T> or object.Equals.
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadonlyListBase<(Of <T>)>::LastIndexOf Method (T, Int32)

See Also

Visual Basic (Declaration) ▸ Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the last item, in the range of items extending from the beginning of the list to index, that is equal to item.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual int LastIndexOf(
    T item,
    int index
)

Visual Basic (Declaration)

Public Overridable Function LastIndexOf ( _
    item As T, _
    index As Integer _
) As Integer

Visual C++

public:
    virtual int LastIndexOf ( 
        T item,
        int index
    )

Parameters

item
    T
    The item to search for.

index
    Int32
    The ending index of the range to check.

Return Value

The index of the last item in the given range that that is equal to item. If no item is equal to item, -1 is returned.
Remarks

The default implementation of equality for type T is used in the search. This is the equality defined by IComparable<T> or object.Equals.
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadonlyListBase<Of T>...:::LastIndexOf Method (T, Int32, Int32)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the index of the last item, in the range of count items ending at index, that is equal to item.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public virtual int LastIndexOf(
    T item,
    int index,
    int count
)
```

Visual Basic (Declaration)

```vbnet
Public Overridable Function LastIndexOf ( _
    item As T, _
    index As Integer, _
    count As Integer _
) As Integer
```

Visual C++

```cpp
public:
    virtual int LastIndexOf ( int item, int index, int count
```

Parameters

item

T
The item to search for.

index

Int32
The ending index of the range to check.

count

Int32
The number of items in range to check.

**Return Value**

The index of the last item in the given range that is equal to item. If no item is equal to item, -1 is returned.
Remarks

The default implementation of equality for type T is used in the search. This is the equality defined by IComparable<T> or object.Equals.
See Also

**ReadOnlyListBase(Of T)>** Class  
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
ReadOnlyListBase<Of <T>>::Range Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a view onto a sub-range of this list. Items are not copied; the returned IList<T> is simply a different view onto the same underlying items.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual IList<T> Range(
    int start,
    int count
)

Visual Basic (Declaration)

Public Overridable Function Range ( _
    start As Integer, _
    count As Integer _
) As IList(Of T)

Visual C++

public:
virtual IList<T>^ Range ( 
    int start,
    int count
)

Parameters

start
    Int32
    The starting index of the view.

count
    Int32
    The number of items in the view.

Return Value

A list that is a view onto the given sub-part of this list.
Remarks

This method can be used to apply an algorithm to a portion of a list. For example:

```csharp
Algorithms.Reverse(deque.Range(3, 6))
```

will return the reverse of the 6 items beginning at index 3.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::ArgumentOutOfRangeException</td>
<td>start or count is negative.</td>
</tr>
<tr>
<td>System..::ArgumentOutOfRangeException</td>
<td>start + count is greater than the size of the list.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of <T>>...:System.Collections.Generic.IList<T>.Insert Method

See Also

- Visual Basic (Declaration)  - Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

**Namespace:** Wintellect.PowerCollections

**Assembly:**  PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
private void IList<T>.Insert(
    int index,
    T item
)
```

**Visual Basic (Declaration)**

```vbnet
Private Sub System.Collections.Generic.IList(T).Insert ( _
    index As Integer, _
    item As T _
) Implements IList(Of T).Insert
```

**Visual C++**

```cpp
private:
virtual void System.Collections.Generic.IList<T>.Insert ( 
    int index,
    T item
) sealed = IList<T>::Insert
```

### Parameters

- **index**
  - `Int32`
- **item**
  - `T`
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<\(T\)>.::.System.Collections.Generic.IList<\(T\)>.RemoveAt Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
private void IList<T>.RemoveAt(
    int index
)
```

**Visual Basic (Declaration)**

```vbnet
Private Sub System.Collections.Generic.IList<T>.RemoveAt ( _
    index As Integer _
) Implements IList(Of T).RemoveAt
```

**Visual C++**

```cpp
private:
virtual void System.Collections.Generic.IList<T>.RemoveAt (_
    int index
) sealed = IList<T>::RemoveAt
```

**Parameters**

index

   Int32
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of <T>>...::System.Collections.IList.Add Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds an item to the end of the list. This implementation throws a NotSupportedException indicating that the list is read-only.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
private int IList.Add(
    Object value
)
```

#### Visual Basic (Declaration)

```vbnet
Private Function System.Collections.IList.Add ( _
    value As Object _
) As Integer Implements IList.Add
```

#### Visual C++

```cpp
private:
virtual int System.Collections.IList.Add ( 
    Object^ value
) sealed = IList::Add
```

### Parameters

value

**Object**

The item to add to the list.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::NotSupportedException</td>
<td>Always thrown.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Removes all the items from the list, resulting in an empty list. This implementation throws a NotSupportedException indicating that the list is read-only.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private void IList.Clear()

Visual Basic (Declaration)

Private Sub System.Collections.IList.Clear Implements IList.Clear

Visual C++

private:
virtual void System.Collections.IList.Clear () sealed = IList::Clear
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::NotSupportedException</td>
<td>Always thrown.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if the list contains any item that compares equal to value.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private bool IList.Contains(
    Object value
)

Visual Basic (Declaration)

Private Function System.Collections.IList.Contains ( _
    value As Object _
) As Boolean Implements IList.Contains

Visual C++

private:
    virtual bool System.Collections.IList.Contains ( 
        Object^ value
    ) sealed = IList::Contains

Parameters

value
    Object
    The item to search for.
Equality in the list is determined by the default sense of equality for T. If T implements IComparable<T>, the Equals method of that interface is used to determine equality. Otherwise, Object.Equals is used to determine equality.
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Find the first occurrence of an item equal to value in the list, and returns the index of that item.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private int IList.IndexOf(
    Object value
)

Visual Basic (Declaration)

Private Function System.Collections.IList.IndexOf ( _
    value As Object _
) As Integer Implements IList.IndexOf

Visual C++

private:
virtual int System.Collections.IList.IndexOf ( _
    Object^ value _
) sealed = IList::IndexOf

Parameters

value
    Object
    The item to search for.

Return Value

The index of value, or -1 if no item in the list compares equal to value.
Remarks

Equality in the list is determined by the default sense of equality for T. If T implements IComparable<T>, the Equals method of that interface is used to determine equality. Otherwise, Object.Equals is used to determine equality.
See Also

ReadOnlyListBase<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of <T>>...::System.Collections.IList.Insert Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Insert a new item at the given index. This implementation throws a NotSupportedException indicating that the list is read-only.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
private void IList.Insert(
    int index,
    Object value
)
```

**Visual Basic (Declaration)**

```vbnet
Private Sub System.Collections.IList.Insert (_
    index As Integer, _
    value As Object _
) Implements IList.Insert
```

**Visual C++**

```cpp
private:
virtual void System.Collections.IList.Insert ( _
    int index,
    Object^ value
) sealed = IList::Insert
```

**Parameters**

**index**

Int32

The index in the list to insert the item at. After the insertion, the inserted item is located at this index. The first item in the list has index 0.

**value**

Object

The item to insert at the given index.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::NotSupportedException</code></td>
<td>Always thrown.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadonlyListBase<Of T>::.::System.Collections.IList.Remove Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Searches the list for the first item that compares equal to value. If one is found, it is removed. Otherwise, the list is unchanged. This implementation throws a NotSupportedException indicating that the list is read-only.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

**C#**

```csharp
private void IList.Remove(
    Object value
)
```

**Visual Basic (Declaration)**

```vbnet
Private Sub System.Collections.IList.Remove (_
    value As Object _
) Implements IList.Remove
```

**Visual C++**

```cpp
private:
    virtual void System.Collections.IList.Remove ( _
        Object^ value
    ) sealed = IList::Remove
```

### Parameters

**value**

*Object*

The item to remove from the list.
Equality in the list is determined by the default sense of equality for T. If T implements IComparable<T>, the Equals method of that interface is used to determine equality. Otherwise, Object.Equals is used to determine equality.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::NotSupportedException</code></td>
<td>Always thrown.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of <T>>.::.System.Collections.IList.RemoveAt Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes the item at the given index. This implementation throws a NotSupportedException indicating that the list is read-only.

**Namespace**: Wintellect.PowerCollections
**Assembly**: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
private void IList.RemoveAt(
    int index
)
```

**Visual Basic (Declaration)**

```vbnet
Private Sub System.Collections.IList.RemoveAt ( _
    index As Integer _
) Implements IList.RemoveAt
```

**Visual C++**

```cpp
private:
virtual void System.Collections.IList.RemoveAt (
    int index
) sealed = IList::RemoveAt
```

**Parameters**

index

```plaintext
Int32
The index in the list to remove the item at. The first item in the list has index 0.
```
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::NotSupportedException</td>
<td>Always thrown.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase<Of T>} Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of T>...::TryFind Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the first item in the list that satisfies the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public virtual bool TryFind(
    Predicate<T> predicate,
    out T foundItem
)
```

#### Visual Basic (Declaration)

```vbnet
Public Overridable Function TryFind ( _
    predicate As Predicate(Of T), _
    [OutAttribute] ByRef foundItem As T _
) As Boolean
```

#### Visual C++

```csharp
public:
virtual bool TryFind ( 
    Predicate<T>^ predicate,
    [OutAttribute] T% foundItem
)
```

### Parameters

- **predicate**
  - `Predicate<(Of <T>)>`
  - A delegate that defines the condition to check for.

- **foundItem**
  - `T%`
  - If true is returned, this parameter receives the first item in the list that satisfies the condition defined by predicate.

### Return Value

True if an item that satisfies the condition predicate was found. False if no item
in the list satisfies that condition.
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of <T>>::TryFindLast Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Finds the last item in the list that satisfies the condition defined by predicate.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public virtual bool TryFindLast(
    Predicate<T> predicate,
    out T foundItem
)
```

### Visual Basic (Declaration)

```vbnet
Public Overridable Function TryFindLast ( _
    predicate As Predicate(Of T), _
    <OutAttribute> ByRef foundItem As T _
) As Boolean
```

### Visual C++

```cpp
public:
virtual bool TryFindLast (
    Predicate<T>^ predicate,
    [OutAttribute] T% foundItem
)
```

## Parameters

**predicate**

`Predicate<T> predicate`

A delegate that defines the condition to check for.

**foundItem**

`T%`

If true is returned, this parameter receives the last item in the list that satisfies the condition defined by `predicate`.

## Return Value

True if an item that satisfies the condition `predicate` was found. False if no item
in the list satisfies that condition.
See Also

ReadOnlyListBase<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<
<
<T>
>
Properties

See Also

☐ Include Inherited Members ☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public abstract class ReadOnlyListBase<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public MustInherit Class ReadOnlyListBase(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class ReadOnlyListBase abstract
Type Parameters

T

The type exposes the following properties.
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>The property must be overridden by the derived class to return the number of items in the list. (Overrides <code>ReadOnlyCollectionBase&lt;Of T&gt;()::Count</code>.)</td>
</tr>
<tr>
<td>Item</td>
<td>The get part of the indexer must be overridden by the derived class to get values of the list at a particular index.</td>
</tr>
</tbody>
</table>
# Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;Of &lt;T&gt;&gt;,::IsReadOnly</code></td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>) Indicates whether the collection is synchronized.</td>
</tr>
<tr>
<td><code>ICollection,::IsSynchronized</code></td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>) Indicates the synchronization object for this collection.</td>
</tr>
<tr>
<td><code>ICollection,::SyncRoot</code></td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;Of &lt;T&gt;&gt;</code>)</td>
</tr>
<tr>
<td><code>IList,::IsFixedSize</code></td>
<td>Returns whether the list is a fixed size. This implementation always returns true.</td>
</tr>
<tr>
<td><code>IList,::IsReadOnly</code></td>
<td>Returns whether the list is read only. This implementation always returns true.</td>
</tr>
<tr>
<td><code>IList,::Item</code></td>
<td>Gets or sets the value at a particular index in the list.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase<(Of <T>)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<(Of <T>)>...::Count Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

The property must be overridden by the derived class to return the number of items in the list.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public abstract int Count { get; }
```

**Visual Basic (Declaration)**

```vbnet
Public MustOverride ReadOnly Property Count As Integer
```

**Visual C++**

```cpp
public:
virtual property int Count {
    int get () abstract override;
}
```

### Field Value

The number of items in the list.
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of <T>>...::Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

The get part of the indexer must be overridden by the derived class to get values of the list at a particular index.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
## Syntax

**C#**

```csharp
public virtual T this[
    int index
]{{ get; set;}}
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable Default Property Item ( _
    index As Integer _
) As T
```

**Visual C++**

```csharp
public:
    virtual property T default[int index] {
        T get (int index);
        void set (int index, T value);
    }
```

**Parameters**

index

*Int32*

The index in the list to get or set an item at. The first item in the list has index 0, and the last has index Count-1.

**Return Value**

The item at the given index.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>index is less than zero or greater than or equal to Count.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Returns whether the list is a fixed size. This implementation always returns true.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private bool IList.IsFixedSize{ get; }

Visual Basic (Declaration)

Private ReadOnly Property System.Collections.IList.IsFixedSize As Boolean

Visual C++

private:
virtual property bool System.Collections.IList.IsFixedSize { bool get () sealed = IList::IsFixedSize::get; }

Field Value

Always true, indicating that the list is fixed size.
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase(Of <T>)*`: System.Collections.IList.IsReadOnly Property

[See Also]

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns whether the list is read only. This implementation always returns true.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
private bool IList.IsReadOnly { get; }
```

**Visual Basic (Declaration)**

```
Private ReadOnly Property System.Collections.IList.IsReadOnly As Boolean
```

**Visual C++**

```cpp
private:
virtual property bool System.Collections.IList.IsReadOnly {
    bool get () sealed = IList::IsReadOnly::get;
}
```

### Field Value

Always true, indicating that the list is read-only.
See Also

**ReadOnlyListBase(Of T)** Class

**Wintellect.PowerCollections** Namespace

Send comments about this topic to Microsoft.
ReadOnlyListBase<Of <T>>::System.Collections.IList.Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets or sets the value at a particular index in the list.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private Object IList.Item[
    int index
] { get; set; }

Visual Basic (Declaration)

Private Property System.Collections.IList.Item ( _
    index As Integer _
) As Object Implements IList.Item

Visual C++

private:
    virtual property Object^ System.Collections.IList.Item[int index] { 
        Object^ get (int index) sealed = IList::Item::get;
        void set (int index, Object^ value) sealed = IList::Item::set;
    }

Parameters

index
    Int32
    The index in the list to get or set an item at. The first item in the list has
    index 0, and the last has index Count-1.

Field Value

The item at the given index.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...::ArgumentOutOfRangeException</td>
<td>index is less than zero or greater than or equal to Count.</td>
</tr>
<tr>
<td>System...::ArgumentException</td>
<td>value cannot be converted to T.</td>
</tr>
<tr>
<td>System...::NotSupportedException</td>
<td>Always thrown from the setter, indicating that the list is read-only.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyListBase(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
MultiDictionaryBase is a base class that can be used to more easily implement a class that associates multiple values to a single key. The class implements the generic IDictionary<TKey, ICollection<TValue>> interface. The resulting collection is read-only -- items cannot be added or removed.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

```csharp
[SerializableAttribute]
public abstract class ReadOnlyMultiDictionaryBase<TKey, TValue> : ReadOnlyCollection<TKey, TValue>, IEnumerable
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute> _
Public MustInherit Class ReadOnlyMultiDictionaryBase(Of TKey, TValue) 
Inherits ReadOnlyCollectionBase(Of KeyValuePair(Of TKey, ICollection(Of TValue))), _ 
Implements IDictionary(Of TKey, ICollection(Of TValue)), ICollection(Of KeyValuePair(Of TKey, ICollection(Of TValue))), IEnumerable(Of KeyValuePair(Of TKey, ICollection(Of TValue))), IEnumerable
```

**Visual C++**

```cpp
[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class ReadOnlyMultiDictionaryBase abstract : public ICollection<TKeyValuePair<TKey, ICollection<TValue>>>, ICollection<TKey, ICollection<TValue>>, IEnumerable
```

---

**SerializableAttribute**

This attribute tags a class as serializable.

**public abstract class**

A class that defines one or more abstract methods.

**ReadOnlyMultiDictionaryBase**

A generic class that represents a multi dictionary, which is a dictionary that allows multiple values for each key.

**ICollection<TKey, TValue>**

A non-generic collection interface for collections that can contain duplicate values.

**IEnumerable**

An interface that represents a collection of elements that can be enumerated.

**IDictionary<TKey, ICollection<TValue>>**

An interface that represents a dictionary that allows multiple values for each key.

**KeyValuePair<TKey, TValue>**

A typed key-value pair.
Type Parameters

TKey
   The key type of the dictionary.
TValue
   The value type of the dictionary.
To use ReadOnlyMultiDictionaryBase as a base class, the derived class must override Count, Contains(TKey,TValue), EnumerateKeys, and TryEnumerateValuesForKey.
Inheritance Hierarchy

System::Object
  Wintellect.PowerCollections::ReadOnlyCollectionBase(Of KeyValuePair(Of TKey, ICollection(Of TValue))>>)
  Wintellect.PowerCollections::ReadOnlyMultiDictionaryBase(Of TKey, TValue)
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyMultiDictionaryBase<
(OfYear<TKey, TValue>) Members

See Also  Methods  Properties  Constructors  Explicit Interface Implementations

☐  Include Inherited Members  ☐  Include Protected Members
☐  .NET Compact Framework Members Only
☐  XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#  
[SerializableAttribute]
public abstract class ReadOnlyMultiDictionaryBase<TKey, TValue>

Visual Basic (Declaration)  
<SerializableAttribute> _
Public MustInherit Class ReadOnlyMultiDictionaryBase(Of TKey, TValue)

Visual C++  
[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class ReadOnlyMultiDictionaryBase abstract
Type Parameters

TKey
TValue

The type exposes the following members.
### Protected Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ReadOnlyMultiDictionaryBase</strong>(Of <strong>TKey,</strong> TValue&gt;)</td>
<td><em>New</em></td>
</tr>
</tbody>
</table>
### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Contains</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>ContainsKey</strong></td>
<td>Determines whether a given key is found in the dictionary.</td>
</tr>
<tr>
<td><strong>ConvertAll(Of TKey, ICollection&lt;TValue&gt;)</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from <strong>ReadOnlyCollectionBase(Of Of &lt;T&gt;).</strong>)</td>
</tr>
<tr>
<td><strong>CopyTo</strong></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <strong>ReadOnlyCollectionBase(Of Of &lt;T&gt;).</strong>)</td>
</tr>
<tr>
<td><strong>CountWhere</strong></td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from <strong>ReadOnlyCollectionBase(Of Of &lt;T&gt;).</strong>)</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>. (Inherited from <strong>Object.</strong>)</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate. (Inherited from <strong>ReadOnlyCollectionBase(Of Of &lt;T&gt;).</strong>)</td>
</tr>
<tr>
<td><strong>FindAll</strong></td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from <strong>ReadOnlyCollectionBase(Of Of &lt;T&gt;).</strong>)</td>
</tr>
<tr>
<td><strong>ForEach</strong></td>
<td>Performs the specified action on each item in this collection. (Inherited from <strong>ReadOnlyCollectionBase(Of Of &lt;T&gt;).</strong>)</td>
</tr>
<tr>
<td><strong>GetEnumerator</strong></td>
<td>Enumerate all the keys in the dictionary, and for each key, the collection of values for that key. (Overrides **ReadOnlyCollectionBase(Of KeyValuePair(Of TKey, ICollection&lt;TValue&gt;)&gt;&gt;)&gt;&gt;&gt;:GetEnumerator())}.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode()</td>
<td>Serves as a hash function for a particular type. Get_HashCode() is suitable for use in hashing algorithms and data structures like a hash table. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType()</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToArray()</td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo. (Inherited from ReadOnlyCollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td>ToString()</td>
<td>Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.</td>
</tr>
<tr>
<td>TrueForAll()</td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from ReadOnlyCollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
</tbody>
</table>
### Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CountAllValues</strong></td>
<td>Gets a total count of values in the collection. This default implementation is slow; it enumerates all of the keys in the dictionary and calls CountValues on each. A derived class may be able to supply a more efficient implementation.</td>
</tr>
<tr>
<td><strong>CountValues</strong></td>
<td>Gets a count of the number of values associated with a key. The default implementation is slow; it enumerators all of the values (using TryEnumerateValuesForKey) to count them. A derived class may be able to supply a more efficient implementation.</td>
</tr>
<tr>
<td><strong>EnumerateKeys</strong></td>
<td>Enumerate all the keys in the dictionary. This method must be overridden by a derived class. If the derived class does not use the default comparison for values, this methods should be overridden to compare two values for equality.</td>
</tr>
<tr>
<td><strong>EqualValues</strong></td>
<td>This is used for the correct implementation of ICollection.Contains on the Values and KeyValuePairs collections. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Creates a shallow copy of the current Object. Enumerate all of the values associated with a given key. This method must be overridden by the derived class. If the key exists and has values associated with it, an enumerator for those values is returned through values. If the key does not exist, false is returned.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td><strong>TryEnumerateValuesForKey</strong></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td>Gets the number of keys in the dictionary. This property must be overridden in the derived class. (Overrider <a href="#">ReadOnlyCollectionBase</a> of <code>&lt;KeyValuePair&lt;TKey, ICollection&lt;TValue&gt;&gt;&gt;</code>...::Count.)</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Returns a collection of all of the values in the dictionary associated with key. If the key is not present in the dictionary, an ICollection with no values is returned. The returned ICollection is read-only.</td>
</tr>
<tr>
<td><strong>Keys</strong></td>
<td>Gets a read-only collection all the keys in this dictionary.</td>
</tr>
<tr>
<td><strong>KeyPairs</strong></td>
<td>Gets a read-only collection of all key-value pairs in the dictionary. If a key has multiple values associated with it, then a key-value pair is present for each value associated with the key.</td>
</tr>
<tr>
<td><strong>Values</strong></td>
<td>Gets a read-only collection of all the values in the dictionary.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;(Of &lt;T&gt;)&gt;...::Add</code></td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>ICollection&lt;(Of &lt;T&gt;)&gt;...::Clear</code></td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>ICollection&lt;(Of &lt;T&gt;)&gt;...::IsReadOnly</code></td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>ICollection&lt;(Of &lt;T&gt;)&gt;...::Remove</code></td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>IDictionary&lt;(Of &lt;TKey, ICollection&lt;(Of &lt;TValue&gt;)&gt;)&gt;...::Add</code></td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>ReadOnlyCollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
<tr>
<td><code>IDictionary&lt;(Of &lt;TKey, ICollection&lt;(Of &lt;TValue&gt;)&gt;)&gt;...::Item</code></td>
<td>Indicates whether the collection is</td>
</tr>
<tr>
<td><code>IDictionary&lt;(Of &lt;TKey, ICollection&lt;(Of &lt;TValue&gt;)&gt;)&gt;...::Remove</code></td>
<td></td>
</tr>
<tr>
<td><code>IDictionary&lt;(Of &lt;TKey, ICollection&lt;(Of &lt;TValue&gt;)&gt;)&gt;...::TryGetValue</code></td>
<td></td>
</tr>
<tr>
<td><code>IDictionary&lt;(Of &lt;TKey, ICollection&lt;(Of &lt;TValue&gt;)&gt;)&gt;...::Values</code></td>
<td></td>
</tr>
<tr>
<td><code>ICollection...::CopyTo</code></td>
<td>(Inherited from <code>ReadOnlyCollectionBase&lt;(Of &lt;T&gt;)&gt;</code>)</td>
</tr>
</tbody>
</table>
**ICollection::IsSynchronized**

synchronized.

(Inherited from `ReadOnlyCollectionBase<Of <T>>`.)

Indicates the synchronization object for this collection.

(Inherited from `ReadOnlyCollectionBase<Of <T>>`).

**ICollection::SyncRoot**

Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator<T> that was overridden by the derived classes to enumerate the members of the collection.

(Inherited from `ReadOnlyCollectionBase<Of <T>>`).

**IEnumerableObject::GetEnumerator**

(Inherited from `ReadOnlyCollectionBase<Of <T>>`).
See Also

ReadOnlyMultiDictionaryBase<(Of <TKey, TValue>))>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyMultiDictionaryBase(Of TKey, TValue)> Constructor

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
protected ReadOnlyMultiDictionaryBase()
```

**Visual Basic (Declaration)**

```vbnet
Protected Sub New
```

**Visual C++**

```cpp
protected:
ReadOnlyMultiDictionaryBase ()
```
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyMultiDictionaryBase<
(OF <TKey, TValue>)> Methods

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
- Syntax

**C#**

```csharp
[SerializableAttribute]
public abstract class ReadOnlyMultiDictionaryBase<TKey, TValue>
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute> _
Public MustInherit Class ReadOnlyMultiDictionaryBase(Of TKey, TValue)
```

**Visual C++**

```cpp
[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class ReadOnlyMultiDictionaryBase abstract
Type Parameters

TKey
TValue

The type exposes the following methods.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>ContainsKey</td>
<td>Determines whether a given key is found in the dictionary.</td>
</tr>
<tr>
<td>ConvertAll&lt;Of TOutput&gt;</td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order. (Inherited from <code>ReadOnlyCollectionBase&lt;Of T&gt;</code>.</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>ReadOnlyCollectionBase&lt;Of T&gt;</code>.</td>
</tr>
<tr>
<td>CountWhere</td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate. (Inherited from <code>ReadOnlyCollectionBase&lt;Of T&gt;</code>.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.</td>
</tr>
<tr>
<td>Exists</td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate. (Inherited from <code>ReadOnlyCollectionBase&lt;Of T&gt;</code>.</td>
</tr>
<tr>
<td>FindAll</td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate. (Inherited from <code>ReadOnlyCollectionBase&lt;Of T&gt;</code>.</td>
</tr>
<tr>
<td>ForEach</td>
<td>Performs the specified action on each item in this collection. (Inherited from <code>ReadOnlyCollectionBase&lt;Of T&gt;</code>.</td>
</tr>
<tr>
<td>GetEnumerator</td>
<td>Enumerate all the keys in the dictionary, and for each key, the collection of values for that key. (Overrides <code>ReadOnlyCollectionBase&lt;Of KeyValuePair&lt;TKey, ICollection&lt;TValue&gt;&gt;&gt;.GetEnumerator()</code>).</td>
</tr>
</tbody>
</table>
Serves as a hash function for a particular type. 

**GetHashCode**

`GetHashCode()` is suitable for use in hashing algorithms and data structures like a hash table.
(Inherited from `Object`.)

**GetType**

Gets the `Type` of the current instance.
(Inherited from `Object`.)

**ToArray**

Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.
(Inherited from `ReadOnlyCollectionBase<Of <T>>`.)

Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.

**ToString**

(Overrides `ReadOnlyCollectionBase<Of <KeyValuePair<Of TKey, ICollection<TValue>>>>`...`ToString()`.)

Determines if all of the items in the collection satisfy the condition defined by predicate.
(Inherited from `ReadOnlyCollectionBase<Of <T>>`.)
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CountAllValues</strong></td>
<td>Gets a total count of values in the collection. This default implementation is slow; it enumerates all of the keys in the dictionary and calls CountValues on each. A derived class may be able to supply a more efficient implementation.</td>
</tr>
<tr>
<td><strong>CountValues</strong></td>
<td>Gets a count of the number of values associated with a key. The default implementation is slow; it enumerators all of the values (using TryEnumerateValuesForKey) to count them. A derived class may be able to supply a more efficient implementation.</td>
</tr>
<tr>
<td><strong>EnumerateKeys</strong></td>
<td>Enumerate all the keys in the dictionary. This method must be overridden by a derived class. If the derived class does not use the default comparison for values, this methods should be overridden to compare two values for equality.</td>
</tr>
<tr>
<td><strong>EqualValues</strong></td>
<td>This is used for the correct implementation of ICollection.Contains on the Values and KeyValuePairs collections.</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>TryEnumerateValuesForKey</strong></td>
<td>Enumerate all of the values associated with a given key. This method must be overridden by the derived class. If the key exists and has values associated with it, an enumerator for those values is returned throught values. If the key does not exist, false is returned.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection(Of T)&gt;::Add</td>
<td>(Inherited from ReadOnlyCollectionBase(Of T&gt;).)</td>
</tr>
<tr>
<td>ICollection(Of T)&gt;::Clear</td>
<td>(Inherited from ReadOnlyCollectionBase(Of T&gt;).)</td>
</tr>
<tr>
<td>ICollection(Of T&gt;)::Remove</td>
<td>(Inherited from ReadOnlyCollectionBase(Of T&gt;).)</td>
</tr>
<tr>
<td>IDictionary(Of TKey, ICollection(Of TValue)&gt;)&gt;::Add</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetIEnumerator to get all the items and copy them to the provided array. (Inherited from ReadOnlyCollectionBase(Of T&gt;).)</td>
</tr>
<tr>
<td>IDictionary(Of TKey, ICollection(Of TValue)&gt;)&gt;::Remove</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator&lt;T&gt; that was overridden by the derived classes to enumerate the members of the collection. (Inherited from ReadOnlyCollectionBase(Of T&gt;).)</td>
</tr>
<tr>
<td>ICollection::CopyTo</td>
<td></td>
</tr>
<tr>
<td>IEnumerable::GetEnumerator</td>
<td></td>
</tr>
<tr>
<td>IEnumerable::GetEnumerator</td>
<td></td>
</tr>
</tbody>
</table>
See Also

ReadOnlyMultiDictionaryBase<(<TKey, TValue>)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyMultiDictionaryBase(Of TKey, TValue)::.Contains Method

See Also Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Contains(KeyValuePair&lt;TKey, ICollection&lt;TValue&gt;&gt;)</code></td>
<td>Determines if this dictionary contains the given key and all of the values associated with that key. (Overrides <code>ReadOnlyCollectionBase&lt;KeyValuePair&lt;TKey, ICollection&lt;TValue&gt;&gt;&gt;</code>::<code>Contains(T)</code>.)</td>
</tr>
<tr>
<td><code>Contains(TKey, TValue)</code></td>
<td>Determines if this dictionary contains a key-value pair equal to key and value. The dictionary is not changed. This method must be overridden in the derived class.</td>
</tr>
</tbody>
</table>
See Also

ReadOnlyMultiDictionaryBase<(Of <TKey, TValue>)> Class
ReadOnlyMultiDictionaryBase<(Of <TKey, TValue>)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this dictionary contains the given key and all of the values associated with that key..

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
# Syntax

**C#**

```csharp
public override bool Contains(
    KeyValuePair<TKey, ICollection<TValue>> pair
)
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides Function Contains ( _
    pair As KeyValuePair(Of TKey, ICollection(Of TValue)) _
) As Boolean
```

**Visual C++**

```cpp
public:
    virtual bool Contains ( 
        KeyValuePair<TKey, ICollection<TValue>^> pair
    ) override
```

**Parameters**

pair
    `KeyValuePair<Of <TKey, ICollection<Of <TValue>>>>>`
    A key and collection of values to search for.

**Return Value**

True if the dictionary has associated all of the values in `pair.Value` with `pair.Key`. 
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this dictionary contains a key-value pair equal to key and value. The dictionary is not changed. This method must be overridden in the derived class.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
public abstract bool Contains(
    TKey key,
    TValue value
)
```

#### Visual Basic (Declaration)

```vbnet
Public MustOverride Function Contains ( _
    key As TKey, _
    value As TValue _
) As Boolean
```

#### Visual C++

```cpp
public:
virtual bool Contains ( 
    TKey key, 
    TValue value 
) abstract
```

### Parameters

- **key**
  - TKey
  - The key to search for.

- **value**
  - TValue
  - The value to search for.

### Return Value

True if the dictionary has associated value with key.
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyMultiDictionaryBase(Of TKey, TValue)>::ContainsKey Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines whether a given key is found in the dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual bool ContainsKey(TKey key)

Visual Basic (Declaration)

Public Overridable Function ContainsKey ( _
    key As TKey _
) As Boolean

Visual C++

public:
    virtual bool ContainsKey ( 
        TKey key
    )

Parameters

key
    TKey TKey
    Key to look for in the dictionary.

Return Value

True if the key is present in the dictionary.
Remarks

The default implementation simply calls TryGetValue. It may be appropriate to override this method to provide a more efficient implementation.
See Also

**ReadOnlyMultiDictionaryBase(Of TKey, TValue)>** Class
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
Gets a total count of values in the collection. This default implementation is slow; it enumerates all of the keys in the dictionary and calls `CountValues` on each. A derived class may be able to supply a more efficient implementation.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#
protected virtual int CountAllValues()

Visual Basic (Declaration)
Protected Overridable Function CountAllValues As Integer

Visual C++
protected:
virtual int CountAllValues ()

Return Value

The total number of values associated with all keys in the dictionary.
See Also

ReadOnlyMultiDictionaryBase<(Of <TKey, TValue>)) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Gets a count of the number of values associated with a key. The default implementation is slow; it enumerators all of the values (using TryEnumerateValuesForKey) to count them. A derived class may be able to supply a more efficient implementation.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

protected virtual int CountValues(TKey key)

Visual Basic (Declaration)

Protected Overridable Function CountValues ( _
    key As TKey _
) As Integer

Visual C++

protected:
virtual int CountValues ( 
    TKey key
)

Parameters

key
    TKey
    The key to count values for.

Return Value

The number of values associated with key.
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyMultiDictionaryBase(Of TKey, TValue)>...:::EnumerateKeys

Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Enumerate all the keys in the dictionary. This method must be overridden by a derived class.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
protected abstract IEnumerator<TKey> EnumerateKeys()
```

**Visual Basic (Declaration)**

```vbnet
Protected MustOverride Function EnumerateKeys As IEnumerable(Of TKey)
```

**Visual C++**

```cpp
protected:
   virtual IEnumerable<TKey>^ EnumerateKeys () abstract
```

**Return Value**

An `IEnumerator<TKey>` that enumerates all of the keys in the collection that have at least one value associated with them.
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyMultiDictionaryBase(Of TKey, TValue)>...::EqualValues Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

If the derived class does not use the default comparison for values, this method should be overridden to compare two values for equality. This is used for the correct implementation of ICollection.Contains on the Values and KeyValuePairs collections.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
protected virtual bool EqualValues(
    TValue value1,
    TValue value2
)
```

**Visual Basic (Declaration)**

```vbnet
Protected Overridable Function EqualValues ( _
    value1 As TValue, _
    value2 As TValue _
) As Boolean
```

**Visual C++**

```cpp
protected:
    virtual bool EqualValues ( 
    TValue value1, 
    TValue value2 
)
```

**Parameters**

- **value1**
  - `TValue`
  - First value to compare.

- **value2**
  - `TValue`
  - Second value to compare.

**Return Value**

True if the values are equal.
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Enumerate all the keys in the dictionary, and for each key, the collection of values for that key.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public override IEnumerator<
    KeyValuePair<TKey,
    ICollection<TValue>>
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides Function GetEnumerator As IEnumerator(Of
    KeyValuePair(Of TKey,
    ICollection(Of TValue)))
```

**Visual C++**

```cpp
public:
virtual Enumerator<
    KeyValuePair<TKey,
    ICollection<TValue>>^
```

**Return Value**

An enumerator to enumerate all the key, ICollection<value> pairs in the dictionary.
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyMultiDictionaryBase<br>(Of <TKey,<br>TValue>)<...:System.Collections.Generic.IDictionary<TKey,System.Collections.<br>Method
See Also

[This topic is pre-release documentation and is subject to change in future<br>releases. Blank topics are included as placeholders.]

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private void IDictionary<TKey, ICollection<TValue>>.Add(
    TKey key,
    ICollection<TValue> values
)

Visual Basic (Declaration)

values As ICollection(Of TValue)_)
) Implements IDictionary(Of TKey, ICollection(Of TValue)).Add

Visual C++

private:
virtual void System.Collections.Generic.IDictionary<TKey,System.Collections.Generic.ICollection<TValue>>::Add(TKey key,_
    ICollection<TValue>^ values
) sealed = IDictionary<TKey, ICollection<TValue>^>::Add

Parameters

key
    TKey

values
    ICollection<(Of <TValue>)>
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyMultiDictionaryBase(Of TKey, TValue)\...

Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private bool IDictionary<TKey, ICollection<TValue>>.Remove(TKey key)

Visual Basic (Declaration)

Private Function System.Collections.Generic.IDictionary<TKey,System.Collections.Generic.ICollection<TValue>>.Remove(key As TKey) As Boolean Implements IDictionary(Of TKey, ICollection(Of TValue))

Visual C++


Parameters

key

TKey
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

#### C#

```csharp
private bool IDictionary<TKey, ICollection<TValue>>.TryGetValue(TKey key, out ICollection<TValue> values)
```

#### Visual Basic (Declaration)

```vbnet
    <OutAttribute> ByRef values As ICollection(Of TValue)) _
) As Boolean Implements IDictionary(Of TKey, ICollection(Of TValue))
```

#### Visual C++

```cpp
private:
    virtual bool System.Collections.Generic.IDictionary<TKey,System.Collections.Generic.ICollection<TValue>>::TryGetValue(TKey key, _
    [OutAttribute] ICollection<TValue>^% values
) sealed = IDictionary<TKey, ICollection<TValue>>^::TryGetValue
```

### Parameters

- **key**
  - TKey

- **values**
  - ICollection<(Of TValue)>%
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Shows the string representation of the dictionary. The string representation contains a list of the mappings in the dictionary.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

public override string ToString() 

### Visual Basic (Declaration)

Public Overrides Function ToString As String

### Visual C++

public:
virtual String^ ToString() override

## Return Value

The string representation of the dictionary.
See Also

ReadOnlyMultiDictionaryBase<(Of < TKey, TValue >)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Enumerate all of the values associated with a given key. This method must be overridden by the derived class. If the key exists and has values associated with it, an enumerator for those values is returned through values. If the key does not exist, false is returned.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
protected abstract bool TryEnumerateValuesForKey(
    TKey key,
    out IEnumerable<TValue> values
)
```

**Visual Basic (Declaration)**

```vbnet
Protected MustOverride Function TryEnumerateValuesForKey (_
    key As TKey, _
    <OutAttribute> ByRef values As IEnumerable(Of TValue) _
) As Boolean
```

**Visual C++**

```csharp
protected:
virtual bool TryEnumerateValuesForKey ( _
    TKey key,
    [OutAttribute] IEnumerable<TValue>^ values
) abstract
```

### Parameters

**key**

- **TKey**
  - The key to get values for.

**values**

- `IEnumerable<(Of <TValue>)>%`
  - If true is returned, this parameter receives an enumerators that enumerates the values associated with that key.

### Return Value

- True if the key exists and has values associated with it. False otherwise.
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyMultiDictionaryBase(Of TKey, TValue) Properties

See Also

☐ Include Inherited Members ☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Syntax

### C#

```csharp
[SerializableAttribute]
public abstract class ReadOnlyMultiDictionaryBase<TKey, TValue>
```

### Visual Basic (Declaration)

```vbnet
<SerializableAttribute> _
Public MustInherit Class ReadOnlyMultiDictionaryBase(Of TKey, TValue)
```

### Visual C++

```c++
[SerializableAttribute]
generic<typename TKey, typename TValue>
public ref class ReadOnlyMultiDictionaryBase abstract
```
Type Parameters

TKey
TValue

The type exposes the following properties.
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td>Gets the number of keys in the dictionary. This property must be overridden</td>
</tr>
<tr>
<td></td>
<td>in the derived class. (Overrides <code>ReadOnlyCollectionBase&lt;TKey, ICollection&lt;TValue&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Returns a collection of all of the values in the dictionary associated with the key.</td>
</tr>
<tr>
<td></td>
<td>If the key is not present in the dictionary, an <code>ICollection</code> with no values is returned. The returned <code>ICollection</code> is read-only.</td>
</tr>
<tr>
<td><strong>Keys</strong></td>
<td>Gets a read-only collection all the keys in this dictionary.</td>
</tr>
<tr>
<td><strong>KeyValuePair</strong></td>
<td>Gets a read-only collection of all key-value pairs in the dictionary. If a key has multiple values associated with it, then a key-value pair is present for each value associated with the key.</td>
</tr>
<tr>
<td><strong>Values</strong></td>
<td>Gets a read-only collection of all the values in the dictionary.</td>
</tr>
</tbody>
</table>
# Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;T&gt;</code>, <code>::IsReadOnly</code></td>
<td>Indicates whether the collection is synchronized. (Inherited from <code>ReadOnlyCollectionBase&lt;T&gt;</code>. )</td>
</tr>
<tr>
<td><code>IDictionary&lt; TKey, ICollection&lt;TValue&gt; &gt;</code>, <code>::Item</code></td>
<td>Indicates the synchronization object for this collection. (Inherited from <code>ReadOnlyCollectionBase&lt;T&gt;</code>. )</td>
</tr>
<tr>
<td><code>IDictionary&lt; TKey, ICollection&lt;TValue&gt; &gt;</code>, <code>::Values</code></td>
<td></td>
</tr>
<tr>
<td><code>ICollection</code>, <code>::IsSynchronized</code></td>
<td></td>
</tr>
<tr>
<td><code>ICollection</code>, <code>::SyncRoot</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue)
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Gets the number of keys in the dictionary. This property must be overridden in the derived class.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public abstract int Count { get; }

Visual Basic (Declaration)

Public MustOverride ReadOnly Property Count As Integer

Visual C++

public:
virtual property int Count {
    int get () abstract override;
}
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyMultiDictionaryBase<Of <TKey, TValue>>::.Item Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns a collection of all of the values in the dictionary associated with key. If the key is not present in the dictionary, an ICollection with no values is returned. The returned ICollection is read-only.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public virtual ICollection<TValue> this[TKey key] {
    get;
}
```

### Visual Basic (Declaration)

```vbnet
Public Overridable ReadOnly Default Property Item ( _
    key As TKey _
) As ICollection(Of TValue)
```

### Visual C++

```cpp
public:
virtual property ICollection<TValue>^ default[TKey key] {
    ICollection<TValue>^ get (TKey key);
}
```

## Parameters

**key**

- **TKey**
  - The key to get the values associated with.

## Field Value

An ICollection<TValue> with all the values associated with key.
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
ReadOnlyMultiDictionaryBase(Of TKey, TValue)>...:::Keys Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Gets a read-only collection all the keys in this dictionary.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual ICollection<TKey> Keys{ get; }

Visual Basic (Declaration)

Public Overridable ReadOnly Property Keys As ICollection(Of TKey)

Visual C++

public:
virtual property ICollection<TKey>^ Keys {
    ICollection<TKey>^ get ();
}

Field Value

An readonly ICollection<TKey> of all the keys in this dictionary.
See Also

ReadOnlyMultiDictionaryBase<
(Of TKey, TValue
)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Gets a read-only collection of all key-value pairs in the dictionary. If a key has multiple values associated with it, then a key-value pair is present for each value associated with the key.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public virtual ICollection<KeyValuePair<TKey, TValue>> KeyValuePairs;
```

**Visual Basic (Declaration)**

```vbnet
Public Overridable ReadOnly Property KeyValuePairs As ICollection(Of TKey, TValue)
```

**Visual C++**

```cpp
public: virtual property ICollection<KeyValuePair<TKey, TValue>>^ KeyValuePairs;
ICollection<KeyValuePair<TKey, TValue>>^ get ();
```
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
private ICollection<TValue> IDictionary<TKey, ICollection<TValue>>.]
TKey key
} { get; set; }
```

**Visual Basic (Declaration)**

```vbnet
key As TKey _
) As ICollection(Of TValue) Implements IDictionary(Of TKey, ICollection)
```

```csharp
private:
virtual property ICollection<TValue>^ System.Collections.Generic.IDictionary<TKey, ICollection<TValue>>^ get (TKey key) sealed = IDictionary^<TKey, ICollection<TValue>^ value) sealed = IDictionary
}
```

**Parameters**

- **key**
  - TKey
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private ICollection<ICollection<TValue>> IDictionary<TKey, ICollection<TValue>>


private:
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Gets a read-only collection of all the values in the dictionary.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public virtual ICollection<TValue> Values{ get; }

Visual Basic (Declaration)

Public Overridable ReadOnly Property Values As ICollection(Of TValue)

Visual C++

public:
virtual property ICollection<TValue>^ Values {
    ICollection<TValue>^ get ();
}

Return Value

A read-only ICollection<TValue> of all the values in the dictionary.
See Also

ReadOnlyMultiDictionaryBase(Of TKey, TValue) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<Of T>> Class

See Also  Members

- [Visual Basic (Declaration)]
- [Visual Basic (Usage)]
- [C#]
- [Visual C++]
- [J#]
- [JScript]
- [XAML]

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Set<T> is a collection that contains items of type T. The item are maintained in a haphazard, unpredictable order, and duplicate items are not allowed.

**Namespace**: Wintellect.PowerCollections

**Assembly**: PowerCollections (in PowerCollections.dll)
Syntax

C#

[SerializableAttribute]
public class Set<T> : CollectionBase<T>, ICollection<T>, IEnumerable<T>, IEnumerable, ICloneable

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class Set(Of T) _
    Inherits CollectionBase(Of T) _
    Implements ICollection(Of T), IEnumerable(Of T), _
        IEnumerable, ICloneable

Visual C++

[SerializableAttribute]
generic<typeparamname T>
public ref class Set : public CollectionBase<T>,
    ICollection<T>, IEnumerable<T>, IEnumerable, ICloneable
Type Parameters

T
Remarks

The items are compared in one of two ways. If T implements IComparable<T> then the Equals method of that interface will be used to compare items, otherwise the Equals method from Object will be used. Alternatively, an instance of IComparer<T> can be passed to the constructor to use to compare items.

Set is implemented as a hash table. Inserting, deleting, and looking up an element all are done in approximately constant time, regardless of the number of items in the Set.

OrderedSet<Of<T>> is similar, but uses comparison instead of hashing, and does maintains the items in sorted order.
Inheritance Hierarchy

System..:::Object
Wintellect.PowerCollections..:::CollectionBase<Of <T>>
Wintellect.PowerCollections..:::Set<Of <T>>
See Also

See Also

Set(Of <T>) Members
Wintellect.PowerCollections Namespace
Wintellect.PowerCollections...:::OrderedSet(Of <T>)

Send comments about this topic to Microsoft.
Set<Of<T>> Members

See Also  Methods  Properties  Constructors  Explicit Interface Implementations

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

```csharp
[SerializableAttribute]
public class Set<T>
```

**Visual Basic (Declaration)**

```vbnet
<SerializableAttribute> _
Public Class Set(Of T)
```

**Visual C++**

```cpp
[SerializableAttribute]
generic< typename T >
public ref class Set
```
- **Type Parameters**

T

The type exposes the following members.
## Public Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Set&lt;Of&lt;T&gt;&gt; Set&lt;Of&lt;T&gt;&gt; New</code></td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Must be overridden to allow adding items to this collection. Adds all the items in collection to the set. If the set already contains an item equal to one of the items in collection, that item will be replaced.</td>
</tr>
<tr>
<td><strong>AddMany</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.)</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all items from the set.</td>
</tr>
<tr>
<td><strong>Clone</strong></td>
<td>Makes a shallow clone of this set; i.e., if items of the set are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td><strong>CloneContents</strong></td>
<td>Makes a deep clone of this set. A new set is created with a clone of each element of this set, by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable&lt;T&gt;.Equals or Object.Equals.</td>
</tr>
<tr>
<td><strong>ConvertAll&lt;Of TOutput&gt;</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array.</td>
</tr>
<tr>
<td>CountWhere</td>
<td>Counts the number of items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td>Difference</td>
<td>Computes the difference of this set with another set. The difference of these two sets is all items that appear in this set, but not in otherSet. A new set is created with the difference of the sets and is returned. This set and the other set are unchanged.</td>
</tr>
<tr>
<td>DifferenceWith</td>
<td>Computes the difference of this set with another set. The difference of these two sets is all items that appear in this set, but not in otherSet. This set receives the difference of the two sets; the other set is unchanged.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>Exists</td>
<td>Determines if the collection contains any item that satisfies the condition defined by predicate.</td>
</tr>
<tr>
<td>FindAll</td>
<td>Enumerates the items in the collection that satisfy the condition defined by predicate.</td>
</tr>
<tr>
<td>ForEach</td>
<td>Performs the specified action on each item in this collection.</td>
</tr>
<tr>
<td>GetEnumerator</td>
<td>Returns an enumerator that enumerates all the items in the set. The items are enumerated in sorted order. (Overrides CollectionBase&lt;(Of &lt;T&gt;)&gt;::GetEnumerator().) Serves as a hash function for a particular type. GetHashCode() is suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
</tbody>
</table>

(Inherited from CollectionBase<(Of <T>)>.)
(Inherited from Object.)

**GetHashCode**

Gets the **Type** of the current instance.
(Inherited from Object.)

**GetType**

Computes the intersection of this set with another set. The intersection of two sets is all items that appear in both of the sets. A new set is created with the intersection of the sets and is returned. This set and the other set are unchanged.

**Intersection**

Computes the intersection of this set with another set. The intersection of two sets is all items that appear in both of the sets. This set receives the intersection of the two sets, the other set is unchanged.

**IntersectionWith**

Determines if this set is disjoint from another set. Two sets are disjoint if no item from one set is equal to any item in the other set.

**IsDisjointFrom**

Determines if this set is equal to another set. This set is equal to otherSet if they contain the same items.

**IsEqualTo**

Determines if this set is a proper subset of another set. Neither set is modified. This set is a subset of otherSet if every element in this set is also in otherSet. Additionally, this set must have strictly fewer items than otherSet.

**IsProperSubsetOf**

Determines if this set is a proper superset of another set. Neither set is modified. This set is a proper superset of otherSet if every element in otherSet is also in this set. Additionally, this set must have strictly more items than otherSet.

**IsProperSupersetOf**

Determines if this set is a subset of another set. Neither set is modified. This set is a subset of otherSet if every element in this set is also in otherSet.

**IsSubsetOf**

Determines if this set is a superset of another set. Neither set is modified. This set is a superset of otherSet if every element in otherSet is also in this set.
Remarks

IsSupersetOf is computed in time $O(M)$, where $M$ is the size of the otherSet.

- **Remove**
  - Must be overridden to allow removing items from this collection.
  - Removes all the items in the collection that satisfy the condition defined by predicate. (Inherited from CollectionBase<Of<T>>.)

- **RemoveAll**
  - Removes all the items in the collection that satisfy the condition defined by predicate.

- **RemoveMany**
  - Removes all the items in the collection from the set.
  - Computes the symmetric difference of this set with another set. The symmetric difference of two sets is all items that appear in either of the sets, but not both. A new set is created with the symmetric difference of the sets and is returned. This set and the other set are unchanged.

- **SymmetricDifference**
  - Computes the symmetric difference of this set with another set. The symmetric difference of two sets is all items that appear in either of the sets, but not both. This set receives the symmetric difference of the two sets; the other set is unchanged.

- **SymmetricDifferenceWith**
  - Computes the symmetric difference of this set with another set. The symmetric difference of two sets is all items that appear in either of the sets, but not both. This set receives the symmetric difference of the two sets; the other set is unchanged.
ToArray

Creates an array of the correct size, and copies all the items in the collection into the array, by calling CopyTo.
(Inherited from CollectionBase(Of T).)

ToString

Shows the string representation of the collection. The string representation contains a list of the items in the collection.
(Contained collections (except string) are expanded recursively.
(Inherited from CollectionBase(Of T).)

TrueForAll

Determines if all of the items in the collection satisfy the condition defined by predicate.
(Inherited from CollectionBase(Of T).)

TryGetItem

Determines if this set contains an item equal to item, according to the comparison mechanism that was used when the set was created. The set is not changed.

If the set does contain an item equal to item, then the item from the set is returned.

Union

Computes the union of this set with another set. The union of two sets is all items that appear in either or both of the sets. A new set is created with the union of the sets and is returned. This set and the other set are unchanged.
Computes the union of this set with another set. The union of two sets is all items that appear in either or both of the sets. This set receives the union of the two sets, the other set is unchanged.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>)</td>
</tr>
</tbody>
</table>
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comparer</strong></td>
<td>Returns the IEqualityComparer&lt;T&gt; used to compare items in this set.</td>
</tr>
<tr>
<td><strong>Count</strong></td>
<td>Returns the number of items in the set. (Overrides CollectionBase&lt;Of T&gt;::Count.)</td>
</tr>
</tbody>
</table>
### Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `ICollection<Of <T>>::Add` | (Inherited from `CollectionBase<Of <T>>`.)  
Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from `CollectionBase<Of <T>>`.) |
| `ICollection<Of <T>>::IsReadOnly` | Indicates whether the collection is synchronized. (Inherited from `CollectionBase<Of <T>>`.)  
Indicates the synchronization object for this collection. (Inherited from `CollectionBase<Of <T>>`.) |
| `ICollection::CopyTo` | Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator<T> that was overridden by the derived classes to enumerate the members of the collection. (Inherited from `CollectionBase<Of <T>>`.)  
Makes a shallow clone of this set; i.e., if items of the set are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment. |
| `IEnumerable::GetEnumerator` | |
| `ICloneable::Clone` | |
See Also

Set<(Of <T>>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<(Of <T>)> Constructor

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Set<
  Of <T>
  > Set<
  Of <T>
  > New()< |
| Set<
  Of <T>
  > Set<
  Of <T>
  > New(IEnumerable<
  Of <T>>) |
| Set<
  Of <T>
  > Set<
  Of <T>
  > New(IEqualityComparer<
  Of <T>>) |
| Set<
  Of <T>
  > Set<
  Of <T>
  > New(IEnumerable<
  Of <T>>, IEqualityComparer<
  Of <T>>) |

- Creates a new Set. The Equals method and GetHashCode method on T will be used to compare items for equality.
- Creates a new Set. The Equals method and GetHashCode method on T will be used to compare items for equality.
- Creates a new Set. The Equals and GetHashCode method of the passed comparer object will be used to compare items in this set.
- Creates a new Set. The Equals and GetHashCode method of the passed comparer object will be used to compare items in this set. The set is initialized with all the items in the given collection.
See Also

Set(Of T) Class
Set(Of T) Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<Of T> Constructor

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new Set. The Equals method and GetHashCode method on T will be used to compare items for equality.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public Set()

Visual Basic (Declaration)

Public Sub New

Visual C++

public:
Set ()
Remarks

Items that are null are permitted, and will be sorted before all other items.
See Also

Set<Of<T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set(Of T) Constructor (IEnumerable(Of T))

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new Set. The Equals method and GetHashCode method on T will be used to compare items for equality.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
C#

```csharp
public Set(
    IEnumerable<T> collection
)
```

Visual Basic (Declaration)

```vbnet
Public Sub New (_
    collection As IEnumerable(Of T) _
)
```

Visual C++

```csharp
public:
Set ( 
    IEnumerable<T>^ collection
)
```

Parameters

collection

    IEnumerable(Of <T>)
    A collection with items to be placed into the Set.
Remarks

Items that are null are permitted.
See Also

Set(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<Of <T>> Constructor (IEqualityComparer<Of <T>>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new Set. The Equals and GetHashCode method of the passed comparer object will be used to compare items in this set.

**Namespace:** Wintellect.PowerCollections

**Assembly:**  PowerCollections (in PowerCollections.dll)
Syntax

C#

public Set(
    IEqualityComparer<T> equalityComparer
)

Visual Basic (Declaration)

Public Sub New (_
    equalityComparer As IEqualityComparer(Of T) _
)

Visual C++

public:
Set ( IEqualityComparer<T>^ equalityComparer
)

Parameters

equalityComparer
    IEqualityComparer<(Of <T>)>
    An instance of IEqualityComparer<T> that will be used to compare items.
See Also

`Set(Of<T>)` Class
`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
Set<(Of <T>)> Constructor (IEnumerable<(Of <T>)>), IEqualityComparer<(Of <T>)>)

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new Set. The Equals and GetHashCode method of the passed comparer object will be used to compare items in this set. The set is initialized with all the items in the given collection.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

### C#

```csharp
public Set(
    IEnumerable<T> collection,
    IEqualityComparer<T> equalityComparer
)
```

### Visual Basic (Declaration)

```vbnet
Public Sub New (_
    collection As IEnumerable(Of T), _
    equalityComparer As IEqualityComparer(Of T) _
)
```

### Visual C++

```cpp
public:
Set ( 
    IEnumerable<T>^ collection,
    IEqualityComparer<T>^ equalityComparer
)
```

**Parameters**

**collection**

*IEnumerable<Of <T>>*

A collection with items to be placed into the Set.

**equalityComparer**

*IEqualityComparer<Of <T>>*

An instance of IEqualityComparer<T> that will be used to compare items.
See Also

Set<T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<(Of <T>)> Methods

See Also

☐ Include Inherited Members
☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class Set<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class Set(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class Set
Type Parameters

T

The type exposes the following methods.
# Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Must be overridden to allow adding items to this collection.</td>
</tr>
<tr>
<td><strong>AddMany</strong></td>
<td>Adds all the items in collection to the set. If the set already contains an item equal to one of the items in collection, that item will be replaced. Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from <code>ICollectionBase&lt;Of&lt;T&gt;&gt;</code>).</td>
</tr>
<tr>
<td><strong>AsReadOnly</strong></td>
<td>Provides a read-only view of this collection. The returned ICollection&lt;T&gt; provides a view of the collection that prevents modifications to the collection. Use the method to provide access to the collection without allowing changes. Since the returned object is just a view, changes to the collection will be reflected in the view. (Inherited from CollectionBase&lt;Of&lt;T&gt;&gt;.).</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all items from the set.</td>
</tr>
<tr>
<td><strong>Clone</strong></td>
<td>Makes a shallow clone of this set; i.e., if items of the set are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment. Makes a deep clone of this set. A new set is created with a clone of each element of this set, by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment. Determines if the collection contains a particular item. This default implementation iterates all of the items in the collection via GetEnumerator, testing each item against item using IComparable&lt;T&gt;.Equals or Object.Equals. Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.</td>
</tr>
<tr>
<td><strong>CloneContents</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ConvertAll&lt;Of&lt; TOutput&gt;&gt;</strong></td>
<td>Convert this collection of items by applying a delegate to each item in the collection. The resulting enumeration contains the result of applying converter to each item in this collection, in order.</td>
</tr>
</tbody>
</table>
(Inherited from CollectionBase(Of T).)

Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerate to get all the items and copy them to the provided array.

- **CopyTo**

Counts the number of items in the collection that satisfy the condition defined by predicate.

- **CountWhere**

Computes the difference of this set with another set. The difference of these two sets is all items that appear in this set, but not in otherSet. A new set is created with the difference of the sets and is returned. This set and the other set are unchanged.

- **Difference**

Computes the difference of this set with another set. The difference of these two sets is all items that appear in this set, but not in otherSet. This set receives the difference of the two sets; the other set is unchanged.

- **DifferenceWith**

Determines whether the specified Object is equal to the current Object.

- **Equals**

Determines if the collection contains any item that satisfies the condition defined by predicate.

- **Exists**

Enumerates the items in the collection that satisfy the condition defined by predicate.

- **FindAll**

Performs the specified action on each item in this collection.

- **ForEach**

Returns an enumerator that enumerates all the items in the set. The items are enumerated in sorted order.

- **GetEnumerator**

Serves as a hash function for a particular type. GetHashCode() is suitable for use in hashing algorithms and data structures like a hash table.
**GetType**
Gets the **Type** of the current instance.
(Inherited from **Object**.)

**Intersection**
Computes the intersection of this set with another set. The intersection of two sets is all items that appear in both of the sets. A new set is created with the intersection of the sets and is returned. This set and the other set are unchanged.

**IntersectionWith**
Computes the intersection of this set with another set. The intersection of two sets is all items that appear in both of the sets. This set receives the intersection of the two sets, the other set is unchanged.

**IsDisjointFrom**
Determines if this set is disjoint from another set. Two sets are disjoint if no item from one set is equal to any item in the other set.

**IsEqualTo**
Determines if this set is equal to another set. This set is equal to otherSet if they contain the same items.

**IsProperSubsetOf**
Determines if this set is a proper subset of another set. Neither set is modified. This set is a subset of otherSet if every element in this set is also in otherSet. Additionally, this set must have strictly fewer items than otherSet.

**IsProperSupersetOf**
Determines if this set is a proper superset of another set. Neither set is modified. This set is a proper superset of otherSet if every element in otherSet is also in this set. Additionally, this set must have strictly more items than otherSet.

**IsSubsetOf**
Determines if this set is a subset of another set. Neither set is modified. This set is a subset of otherSet if every element in this set is also in otherSet.

**IsSupersetOf**
Determines if this set is a superset of another set. Neither set is modified. This set is a superset of otherSet if every element in otherSet is also in this set.
Remarks

IsSupersetOf is computed in time $O(M)$, where $M$ is the size of the otherSet.

- **Remove**
  - Must be overridden to allow removing items from this collection.
  - Removes all the items in the collection that satisfy the condition defined by predicate.
  - (Inherited from `CollectionBase<Of<T>>`.)

- **RemoveAll**
  - Removes all the items in the collection from the set.
  - (Inherited from `CollectionBase<Of<T>>`.)

- **RemoveMany**
  - Computes the symmetric difference of this set with another set. The symmetric difference of two sets is all items that appear in either of the sets, but not both. A new set is created with the symmetric difference of the sets and is returned. This set and the other set are unchanged.

- **SymmetricDifference**
  - Computes the symmetric difference of this set with another set. The symmetric difference of two sets is all items that appear in either of the sets, but not both. This set receives the symmetric difference of the two sets; the other set is unchanged.

- **SymmetricDifferenceWith**
  - Computes the symmetric difference of this set with another set. The symmetric difference of two sets is all items that appear in either of the sets, but not both. This set receives the symmetric difference of the two sets; the other set is unchanged.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ToArray</code></td>
<td>Creates an array of the correct size, and copies all the items in the collection into the array, by calling <code>CopyTo</code>. (Inherited from <code>CollectionBase&lt;OF &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Shows the string representation of the collection. The string representation contains a list of the items in the collection. Contained collections (except string) are expanded recursively. (Inherited from <code>CollectionBase&lt;OF &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>TrueForAll</code></td>
<td>Determines if all of the items in the collection satisfy the condition defined by predicate. (Inherited from <code>CollectionBase&lt;OF &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>TryGetItem</code></td>
<td>Determines if this set contains an item equal to <code>item</code>, according to the comparison mechanism that was used when the set was created. The set is not changed.</td>
</tr>
<tr>
<td><code>Union</code></td>
<td>Computes the union of this set with another set. The union of two sets is all items that appear in either or both of the sets. A new set is created with the union of the sets and is returned. This set and the other set are unchanged.</td>
</tr>
</tbody>
</table>
Computes the union of this set with another set. The union of two sets is all items that appear in either or both of the sets. This set receives the union of the two sets, the other set is unchanged.
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a> is reclaimed by garbage collection. (Inherited from <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>. (Inherited from <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICollection(&lt;\text{Of }&lt;T&gt;&gt;)::Add</td>
<td>Copies all the items in the collection into an array. Implemented by using the enumerator returned from GetEnumerator to get all the items and copy them to the provided array. (Inherited from <code>CollectionBase\(&lt;\text{Of }&lt;T&gt;&gt;\)</code>.)</td>
</tr>
<tr>
<td>ICollection::CopyTo</td>
<td>Provides an IEnumerator that can be used to iterate all the members of the collection. This implementation uses the IEnumerator(&lt;T&gt;) that was overridden by the derived classes to enumerate the members of the collection. (Inherited from <code>CollectionBase\(&lt;\text{Of }&lt;T&gt;&gt;\)</code>.)</td>
</tr>
<tr>
<td>IEnumerable::GetEnumerator</td>
<td>Makes a shallow clone of this set; i.e., if items of the set are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.</td>
</tr>
<tr>
<td>ICloneable::Clone</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Set<(Of <T>�)>
\texttt{Wintellect.PowerCollections} Namespace

Send \texttt{comments} about this topic to Microsoft.
Set<Of <T>>...::AddMany Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Adds all the items in collection to the set. If the set already contains an item equal to one of the items in collection, that item will be replaced.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public void AddMany(
    IEnumerable<T> collection
)
```

Visual Basic (Declaration)

```vbnet
Public Sub AddMany ( _
    collection As IEnumerable(Of T) _
)
```

Visual C++

```cpp
public:
void AddMany ( 
    IEnumerable<T>^ collection
)
```

Parameters

```xml
<
IEEnumerable(Of '<T>')
A collection of items to add to the set.
```
Equality between items is determined by the comparison instance or delegate used to create the set.

Adding the collection takes time $O(M)$, where $M$ is the number of items in collection.
See Also

Set<Of T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set(Of T)->...:Clear Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all items from the set.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

public override sealed void Clear()

**Visual Basic (Declaration)**

Public Overrides NotOverridable Sub Clear

**Visual C++**

public:
virtual void Clear () override sealed
Remarks

Clearing the set takes a constant amount of time, regardless of the number of items in it.
See Also

Set(Of T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set(Of T)>...:::Clone Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a shallow clone of this set; i.e., if items of the set are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public Set<T> Clone()

Visual Basic (Declaration)

Public Function Clone As Set(Of T)

Visual C++

public: 
Set<T>^ Clone ()

Return Value

The cloned set.
Remarks

Cloning the set takes time $O(N)$, where $N$ is the number of items in the set.
See Also

Set(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set(Of <T>)::.CloneContents Method

See Also

Visual Basic (Declaration)  Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Makes a deep clone of this set. A new set is created with a clone of each element of this set, by calling ICloneable.Clone on each element. If T is a value type, then each element is copied as if by simple assignment.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public Set<T> CloneContents()
```

**Visual Basic (Declaration)**

```vbnet
Public Function CloneContents As Set(Of T)
```

**Visual C++**

```cpp
public: Set<T>^ CloneContents ()
```

**Return Value**

The cloned set.
Remarks

If T is a reference type, it must implement ICloneable. Otherwise, an InvalidOperationException is thrown.

Cloning the set takes time O(N), where N is the number of items in the set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::InvalidOperationException</td>
<td>T is a reference type that does not implement ICloneable.</td>
</tr>
</tbody>
</table>
See Also

Set<$T$> Class  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<Of <T)>>.::Difference Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the difference of this set with another set. The difference of these two sets is all items that appear in this set, but not in otherSet. A new set is created with the difference of the sets and is returned. This set and the other set are unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public Set<T> Difference(
    Set<T> otherSet
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function Difference ( _
    otherSet As Set(Of T) _
) As Set(Of T)
```

**Visual C++**

```cpp
public:
Set<T>^ Difference ( _
    Set<T>^ otherSet
)
```

**Parameters**

otherSet

Set<Of <T>>

Set to difference with.

**Return Value**

The difference of the two sets.
Remarks

The difference of two sets is computed in time $O(N)$, where $N$ is the size of the smaller set.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::InvalidOperationException</td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Set<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<\langle T\rangle>...::DifferenceWith Method

See Also

Visual Basic (Declaration) Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the difference of this set with another set. The difference of these two sets is all items that appear in this set, but not in otherSet. This set receives the difference of the two sets; the other set is unchanged.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public void DifferenceWith(
    Set<T> otherSet
)
```

Visual Basic (Declaration)

```vbnet
Public Sub DifferenceWith (~
    otherSet As Set(Of T) ~
)
```

Visual C++

```cpp
public:
void DifferenceWith (~
    Set<T>^ otherSet
)
```

Parameters

otherSet

```csharp
Set<Of <T>>
```
Set to difference with.
Remarks

The difference of two sets is computed in time $O(N)$, where $N$ is the size of the smaller set.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
- See Also

**Set(Of T)** Class  
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.
Set<(Of <T>)>...::GetEnumerator Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns an enumerator that enumerates all the items in the set. The items are enumerated in sorted order.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public override sealed IEnumerator<T> GetEnumerator()
```

**Visual Basic (Declaration)**

```vbnet
Public Overrides NotOverridable Function GetEnumerator As IEnumerator
```

**Visual C++**

```cpp
public:
virtual IEnumerator<T>^ GetEnumerator() override sealed
```

**Return Value**

An enumerator for enumerating all the items in the Set.
Remarks

Typically, this method is not called directly. Instead the "foreach" statement is used to enumerate the items, which uses this method implicitly.

If an item is added to or deleted from the set while it is being enumerated, then the enumeration will end with an InvalidOperationException.

Enumerating all the items in the set takes time $O(N)$, where $N$ is the number of items in the set.
See Also

**Set(Of T)** Class
**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
Set(Of <T>)::.Intersection Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the intersection of this set with another set. The intersection of two sets is all items that appear in both of the sets. A new set is created with the intersection of the sets and is returned. This set and the other set are unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

    public Set<T> Intersection(
        Set<T> otherSet
    )

Visual Basic (Declaration)

    Public Function Intersection ( _
        otherSet As Set(Of T) _
    ) As Set(Of T)

Visual C++

    public:
    Set<T>^ Intersection ( _
        Set<T>^ otherSet
    )

Parameters

otherSet
    Set<Of <T>>
    Set to intersection with.

Return Value

The intersection of the two sets.
Remarks

When equal items appear in both sets, the intersection will include an arbitrary choice of one of the two equal items.

The intersection of two sets is computed in time $O(N)$, where $N$ is the size of the smaller set.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..:::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
**See Also**

`Set(Of <T>)` Class  
`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
Set<(Of <T>)>.:::IntersectionWith Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the intersection of this set with another set. The intersection of two sets is all items that appear in both of the sets. This set receives the intersection of the two sets, the other set is unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public void IntersectionWith(
    Set<T> otherSet
)

Visual Basic (Declaration)

Public Sub IntersectionWith (_,
    otherSet As Set(Of T) _
)

Visual C++

public:
    void IntersectionWith (_,
        Set<T>^ otherSet
    )

Parameters

otherSet
    Set<(Of <T>)>
    Set to intersection with.
Remarks

When equal items appear in both sets, the intersection will include an arbitrary choice of one of the two equal items.

The intersection of two sets is computed in time $O(N)$, where $N$ is the size of the smaller set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Set<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<(Of <T<?>>)>::IsDisjointFrom Method

See Also

Visual Basic (Declaration) → Visual Basic (Usage)
C#
Visual C++
J#
JScript
XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this set is disjoint from another set. Two sets are disjoint if no item from one set is equal to any item in the other set.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public bool IsDisjointFrom(
    Set<T> otherSet
)
```

Visual Basic (Declaration)

```vbnet
Public Function IsDisjointFrom ( _
    otherSet As Set(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
bool IsDisjointFrom ( _
    Set<T>^ otherSet
)
```

Parameters

otherSet

Set<(Of <T>)>  
Set to check disjointness with.

Return Value

True if the two sets are disjoint, false otherwise.
Remarks

The answer is computed in time $O(N)$, where $N$ is the size of the smaller set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::InvalidOperationException</td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

See Also

Set<(Of <T)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<(Of <T>)>::IsEqualTo Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this set is equal to another set. This set is equal to otherSet if they contain the same items.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public bool IsEqualTo(
    Set<T> otherSet
)

Visual Basic (Declaration)

Public Function IsEqualTo (_
    otherSet As Set(Of T) _
) As Boolean

Visual C++

public:
    bool IsEqualTo ( 
        Set<T>^ otherSet
    )

Parameters

otherSet
    Set<(Of <T>)>
    Set to compare to

Return Value

True if this set is equal to otherSet, false otherwise.
Remarks

IsEqualTo is computed in time $O(N)$, where $N$ is the number of items in this set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..:::InvalidOperationException</td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
<tr>
<td>See Also</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Set</strong>&lt;<strong>(Of &lt;T&gt;&gt;)</strong> Class</td>
<td></td>
</tr>
<tr>
<td><strong>Wintellect.PowerCollections</strong> Namespace</td>
<td></td>
</tr>
</tbody>
</table>

Send [comments](https://www.microsoft.com) about this topic to Microsoft.
Set<(Of <T>)>...::IsProperSubsetOf Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this set is a proper subset of another set. Neither set is modified. This set is a subset of otherSet if every element in this set is also in otherSet. Additionally, this set must have strictly fewer items than otherSet.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public bool IsProperSubsetOf(
    Set<T> otherSet
)

Visual Basic (Declaration)

Public Function IsProperSubsetOf ( _
    otherSet As Set(Of T) _
) As Boolean

Visual C++

public:
    bool IsProperSubsetOf ( 
        Set<T>^ otherSet
    )

Parameters

otherSet
    Set<(Of <T>)>  
    Set to compare to.

Return Value

True if this is a proper subset of otherSet.
Remarks

IsProperSubsetOf is computed in time $O(N)$, where $N$ is the size of the this set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

`Set<Of <T>>` Class
`Wintellect.PowerCollections` Namespace

Send [comments](#) about this topic to Microsoft.
Set<(Of <T>)>...::IsProperSupersetOf Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this set is a proper superset of another set. Neither set is modified. This set is a proper superset of otherSet if every element in otherSet is also in this set. Additionally, this set must have strictly more items than otherSet.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public bool IsProperSupersetOf(
    Set<T> otherSet
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function IsProperSupersetOf ( _
    otherSet As Set(Of T) _
) As Boolean
```

**Visual C++**

```cpp
public:
    bool IsProperSupersetOf ( 
        Set<T>^ otherSet
    )
```

**Parameters**

`otherSet`  
`Set<Of <T>>`  
Set to compare to.

**Return Value**

True if this is a proper superset of otherSet.
Remarks

IsProperSubsetOf is computed in time $O(M)$, where $M$ is the size of otherSet.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::InvalidOperationException</td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Set<Of T> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<(Of <T>)>::IsSubsetOf Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this set is a subset of another set. Neither set is modified. This set is a subset of otherSet if every element in this set is also in otherSet.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public bool IsSubsetOf(
    Set<T> otherSet
)
```

Visual Basic (Declaration)

```vbnet
Public Function IsSubsetOf ( _
    otherSet As Set(Of T) _
) As Boolean
```

Visual C++

```cpp
public:
    bool IsSubsetOf (  
    Set<T>^ otherSet
    )
```

Parameters

otherSet

Set<(Of <T>)>
Set to compare to.

Return Value

True if this is a subset of otherSet.
Remarks

IsSubsetOf is computed in time $O(N)$, where $N$ is the size of the this set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

**Set(Of T)** Class

**Wintellect.PowerCollections** Namespace

Send [comments](#) about this topic to Microsoft.
Set<(Of <T>)>::IsSupersetOf Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Determines if this set is a superset of another set. Neither set is modified. This set is a superset of otherSet if every element in otherSet is also in this set.
Remarks

IsSupersetOf is computed in time $O(M)$, where $M$ is the size of the otherSet.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
# Syntax

**C#**

```csharp
public bool IsSupersetOf(
    Set<T> otherSet
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function IsSupersetOf (_, otherSet As Set(Of T) _) As Boolean
```

**Visual C++**

```cpp
public:
bool IsSupersetOf (  
    Set<T>^ otherSet
)
```

## Parameters

otherSet

Set<(Of <T>))

Set to compare to.

## Return Value

True if this is a superset of otherSet.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Set(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<(Of <T>)>...::RemoveMany Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Removes all the items in collection from the set.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
## Syntax

### C#

```csharp
public int RemoveMany(
    IEnumerable<T> collection
)
```

### Visual Basic (Declaration)

```vbnet
Public Function RemoveMany(
    _
collection As IEnumerable(Of T) _
) As Integer
```

### Visual C++

```cpp
public:
    int RemoveMany (
        IEnumerable<T>^ collection
    )
```

## Parameters

`collection`  
`IEnumerable<Of <T>>`  
A collection of items to remove from the set.

## Return Value

The number of items removed from the set.
Remarks

Equality between items is determined by the comparison instance or delegate used to create the set.

Removing the collection takes time $O(M)$, where $M$ is the number of items in collection.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentNullException</code></td>
<td>collection is null.</td>
</tr>
</tbody>
</table>
See Also

Set<Of<T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<(Of <T>)>...::SymmetricDifference Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the symmetric difference of this set with another set. The symmetric difference of two sets is all items that appear in either of the sets, but not both. A new set is created with the symmetric difference of the sets and is returned. This set and the other set are unchanged.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public Set<T> SymmetricDifference(
    Set<T> otherSet
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function SymmetricDifference ( _
    otherSet As Set(Of T) _
) As Set(Of T)
```

**Visual C++**

```cpp
public:
    Set<T>^ SymmetricDifference ( 
    Set<T>^ otherSet
)
```

### Parameters

otherSet

Set<Of <T>>

Set to symmetric difference with.

### Return Value

The symmetric difference of the two sets.
Remarks

The symmetric difference of two sets is computed in time $O(N)$, where $N$ is the size of the smaller set.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::InvalidOperationException</td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Set<Of<T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Computes the symmetric difference of this set with another set. The symmetric difference of two sets is all items that appear in either of the sets, but not both. This set receives the symmetric difference of the two sets; the other set is unchanged.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public void SymmetricDifferenceWith(
    Set<T> otherSet
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub SymmetricDifferenceWith ( _
    otherSet As Set(Of T) _
)
```

**Visual C++**

```cpp
public:
void SymmetricDifferenceWith ( 
    Set<T>^ otherSet
)
```

Parameters

otherSet
  Set<Of <T>>
  Set to symmetric difference with.
Remarks

The symmetric difference of two sets is computed in time $O(N)$, where $N$ is the size of the smaller set.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Set(Of T>) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<Of <T>> ... System.Collections.Generic.ICollection<T>.Add Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Namespace: Wintellect.PowerCollections

Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#
private void ICollection<T>.Add(T item)

Visual Basic (Declaration)
Private Sub System.Collections.Generic.ICollection(Of T).Add( _
    item As T _
) Implements ICollection(Of T).Add

Visual C++
private:
    virtual void System.Collections.Generic.ICollection<T>.Add(T item)
) sealed = ICollection<T>::Add

Parameters

item
    T
See Also

Set<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<Of <T>>::System.ICloneable.Clone Method

Make a shallow clone of this set; i.e., if items of the set are reference types, then they are not cloned. If T is a value type, then each element is copied as if by simple assignment.

**Namespace:** Wintellect.PowerCollections
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

private Object ICloneable.Clone()  

Visual Basic (Declaration)

Private Function System.ICloneable.Clone As Object Implements IClone

Visual C++

private: virtual Object ^ System.ICloneable.Clone () sealed = ICloneable::Clone

Return Value

The cloned set.
Cloning the set takes time $O(N)$, where $N$ is the number of items in the set.
See Also

Set(Of T) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this set contains an item equal to item, according to the comparison mechanism that was used when the set was created. The set is not changed.

If the set does contain an item equal to item, then the item from the set is returned.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
C#

```csharp
public bool TryGetItem(
    T item,
    out T foundItem
)
```

Visual Basic (Declaration)

```vbnet
Public Function TryGetItem ( _
    item As T, _
    <OutAttribute> ByRef foundItem As T _
) As Boolean
```

Visual C++

```cpp
public:
    bool TryGetItem ( 
        T item,
        [OutAttribute] T% foundItem
    )
```

Parameters

item

T

The item to search for.

foundItem

T%

Returns the item from the set that was equal to item.

Return Value

True if the set contains item. False if the set does not contain item.
Remarks

Searching the set for an item takes approximately constant time, regardless of the number of items in the set.
Examples

In the following example, the set contains strings which are compared in a case-insensitive manner.

```csharp
Set<string> set = new Set<string>(StringComparer.CurrentCultureIgnoreCase);
set.Add("HELLO");
string s;
bool b = set.TryGetItem("Hello", out s); // b receives
```
See Also

Set<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set(Of <T>)::.Union Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the union of this set with another set. The union of two sets is all items that appear in either or both of the sets. A new set is created with the union of the sets and is returned. This set and the other set are unchanged.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public Set<T> Union(
    Set<T> otherSet
)
```

**Visual Basic (Declaration)**

```vbnet
Public Function Union(_
    otherSet As Set(Of T) _
) As Set(Of T)
```

**Visual C++**

```cpp
public:
    Set<T>^ Union ( 
        Set<T>^ otherSet
    )
```

**Parameters**

otherSet

Set(Of T)>

Set to union with.

**Return Value**

The union of the two sets.
**Remarks**

If equal items appear in both sets, the union will include an arbitrary choice of one of the two equal items.

The union of two sets is computed in time $O(M + N)$, where $M$ is the size of the one set, and $N$ is the size of the other set.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..:::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Set<Of <T>> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<Of <T>>::UnionWith Method

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Computes the union of this set with another set. The union of two sets is all items that appear in either or both of the sets. This set receives the union of the two sets, the other set is unchanged.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public void UnionWith(
    Set<T> otherSet
)
```

**Visual Basic (Declaration)**

```vbnet
Public Sub UnionWith (
    otherSet As Set(Of T)
)
```

**Visual C++**

```cpp
public:
void UnionWith (
    Set<T>& otherSet
)
```

**Parameters**

otherSet

- `Set<Of <T>>`
  - Set to union with.
Remarks

If equal items appear in both sets, the union will include an arbitrary choice of one of the two equal items.

The union of two sets is computed in time $O(M + N)$, where $M$ is the size of the larger set, and $N$ is the size of the smaller set.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::InvalidOperationException</code></td>
<td>This set and otherSet don't use the same method for comparing items.</td>
</tr>
</tbody>
</table>
See Also

Set<(Of <T>)) Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set<(Of <T>))> Properties

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public class Set<T>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Class Set(Of T)

Visual C++

[SerializableAttribute]
generic<typename T>
public ref class Set
Type Parameters

\( T \)

The type exposes the following properties.
## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comparer</strong></td>
<td>Returns the IEqualityComparer&lt;T&gt; used to compare items in this set.</td>
</tr>
<tr>
<td><strong>Count</strong></td>
<td>Returns the number of items in the set. (Overrides <code>CollectionBase&lt;Of &lt;T&gt;&gt;::Count</code>.)</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ICollection&lt;Of &lt;T&gt; &gt;::IsReadOnly</code></td>
<td>Indicates whether the collection is synchronized. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection::IsSynchronized</code></td>
<td>Indicates whether the collection is synchronized. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
<tr>
<td><code>ICollection::SyncRoot</code></td>
<td>Indicates the synchronization object for this collection. (Inherited from <code>CollectionBase&lt;Of &lt;T&gt;&gt;</code>.)</td>
</tr>
</tbody>
</table>
See Also

Set(Of<T>)
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set(Of <T>)...::Comparer Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the IEqualityComparer<T> used to compare items in this set.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

public IEqualityComparer<T> Comparer{ get; }

Visual Basic (Declaration)

Public ReadOnly Property Comparer As IEqualityComparer(Of T)

Visual C++

public:
property IEqualityComparer<T>* Comparer { 
    IEqualityComparer<T>* get ();
}

Field Value

If the set was created using a comparer, that comparer is returned. Otherwise the default comparer for T (EqualityComparer<T>.Default) is returned.
See Also

Set<(Of <T>)> Class
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Set(Of T)>...::Count Property

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Returns the number of items in the set.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override sealed int Count { get; }

Visual Basic (Declaration)

Public Overrides NotOverridable ReadOnly Property Count As Integer

Visual C++

public:
virtual property int Count {
    int get () override sealed;
}

Field Value

The number of items in the set.
Remarks

The size of the set is returned in constant time.
See Also

**Set(Of T)** Class

*Wintellect.PowerCollections* Namespace

Send [comments](#) about this topic to Microsoft.
Stores a triple of objects within a single struct. This struct is useful to use as the T of a collection, or as the TKey or TValue of a dictionary.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#  

[SerializableAttribute]
public struct Triple<TFirst, TSecond, TThird> : IComparable,
            IComparable<Triple<TFirst, TSecond, TThird>>

Visual Basic (Declaration)

<SerializableAttribute>
Public Structure Triple(Of TFirst, TSecond, TThird) _
    Implements IComparable, IComparable(Of Triple(Of TFirst, TSecond, TThird))

Visual C++

[SerializableAttribute]
generic<typename TFirst, typename TSecond, typename TThird>
public value class Triple : IComparable,
            IComparable<Triple<TFirst, TSecond, TThird>>
Type Parameters

TFirst
TSecond
TThird
See Also

Triple<(Of <TFirst, TSecond, TThird>)> Members
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Triple<Of <TFirst, TSecond, TThird>> Members

See Also  Methods  Constructors  Fields  Explicit Interface Implementations

- ✔ Include Inherited Members  - Include Protected Members
- ✔ .NET Compact Framework Members Only
- ✔ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

```csharp
[SerializableAttribute]
public struct Triple<TFirst, TSecond, TThird>
```

Visual Basic (Declaration)

```vbnet
<SerializableAttribute> _
Public Structure Triple(Of TFirst, TSecond, TThird)
```

Visual C++

```cpp
[SerializableAttribute]
generic<typename TFirst, typename TSecond, typename TThird>
public value class Triple
```
Type Parameters

TFirst
TSecond
TThird

The type exposes the following members.
### Public Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Triple&lt;Of &lt;TFirst, TSecond, TThird&gt;&gt;Triple&lt;Of &lt;TFirst, TSecond, TThird&gt;&gt; New</code></td>
<td>Creates a new triple with given elements.</td>
</tr>
</tbody>
</table>
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompareTo</td>
<td>Compares this triple to another triple of the same type. The triples are compared by using the IComparable&lt;T&gt; or IComparable interface on TFirst, TSecond, and TThird. The triples are compared by their first elements first, if their first elements are equal, then they are compared by their second elements. If their second elements are also equal, then they are compared by their third elements. If TFirst, TSecond, or TThird does not implement IComparable&lt;T&gt; or IComparable, then a NotSupportedException is thrown, because the triples cannot be compared.</td>
</tr>
<tr>
<td>Equality</td>
<td>Determines if two triples are equal. Two triples are equal if the all three elements compare equal using IComparable&lt;T&gt;.Equals or object.Equals.</td>
</tr>
<tr>
<td>Equals</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the triple, suitable for use in a hash-table or other hashed collection. Two triples that compare equal (using Equals) will have the same hash code. The hash code for the triple is derived by combining the hash codes for each of the two elements of the triple. (Overrides ValueType...::GetHashCode().)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Inequality</td>
<td>Determines if two triples are not equal. Two triples are equal if the all three elements compare equal using IComparable&lt;T&gt;.Equals or object.Equals. Returns a string representation of the triple. The string representation of the triple is of the form: First: {0}, Second: {1}, Third: {2} where {0} is the result of First.ToString(),</td>
</tr>
<tr>
<td>ToString</td>
<td></td>
</tr>
</tbody>
</table>
{1} is the result of Second.ToString(), and {2} is the result of Third.ToString() (or "null" if they are null.)
(Overrides ValueType...::ToString().)
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>)</td>
</tr>
</tbody>
</table>
## Public Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>The first element of the triple.</td>
</tr>
<tr>
<td>Second</td>
<td>The second element of the triple.</td>
</tr>
<tr>
<td>Third</td>
<td>The thrid element of the triple.</td>
</tr>
</tbody>
</table>
## Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `IComparable<T>::CompareTo` | Compares this triple to another triple of the same type. The triples are compared by using the `IComparable<T>` or `IComparable` interface on `TFirst`, `TSecond`, and `TTThird`. The triples are compared by their first elements first, if their first elements are equal, then they are compared by their second elements. If their second elements are also equal, then they are compared by their third elements.

If `TFirst`, `TSecond`, or `TTThird` does not implement `IComparable<T>` or `IComparable`, then an `NotSupportedException` is thrown, because the triples cannot be compared. |
See Also

Triple<(Of TFirst, TSecond, TThird)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Triple<(Of <TFirst, TSecond, TThird>)> Constructor

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Creates a new triple with given elements.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public Triple(
    TFirst first,
    TSecond second,
    TThird third
)

Visual Basic (Declaration)

Public Sub New ( _
    first As TFirst, _
    second As TSecond, _
    third As TThird _
)

Visual C++

public:
    Triple ( 
        TFirst first,
        TSecond second,
        TThird third
    )

Parameters

first
    TFirst
    The first element of the triple.

second
    TSecond
    The second element of the triple.

third
    TThird
The third element of the triple.
See Also

**Triple(Of TFirst, TSecond, TThird)> Structure**
**Wintellect.PowerCollections Namespace**

Send [comments](#) about this topic to Microsoft.
Triple<(Of <TFirst, TSecond, TThird>)> Fields

See Also

- Include Inherited Members
- Include Protected Members
- .NET Compact Framework Members Only
- XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
Syntax

C#

[SerializableAttribute]
public struct Triple<TFirst, TSecond, TThird>

Visual Basic (Declaration)

<SerializableAttribute> _
Public Structure Triple(Of TFirst, TSecond, TThird)

Visual C++

[SerializableAttribute]
generic<typename TFirst, typename TSecond, typename TThird>
public value class Triple
Type Parameters

TFirst
TSecond
TThird

The type exposes the following fields.
## Public Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>The first element of the triple.</td>
</tr>
<tr>
<td>Second</td>
<td>The second element of the triple.</td>
</tr>
<tr>
<td>Third</td>
<td>The thrid element of the triple.</td>
</tr>
</tbody>
</table>
See Also

Triple<(Of <TFirst, TSecond, TThird>)>
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Triple<(Of <TFirst, TSecond, TThird>)>...::First Field

**See Also**

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

The first element of the triple.

**Namespace:** Wintellect.PowerCollections

**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public TFirst First
```

**Visual Basic (Declaration)**

```
Public First As TFirst
```

**Visual C++**

```cpp
public:
TFirst First
```
See Also

Triple(Of TFirst, TSecond, TThird)> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
The second element of the triple.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
**Syntax**

**C#**

```csharp
public TSecond Second
```

**Visual Basic (Declaration)**

```vbnet
Public Second As TSecond
```

**Visual C++**

```cpp
public:
TSecond Second
```
See Also

`Triple<(Of <TFirst, TSecond, TThird>)>` Structure
`Wintellect.PowerCollections` Namespace

Send [comments](#) about this topic to Microsoft.
Triple<(Of <TFirst, TSecond, TThird>)>::Third Field

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

The third element of the triple.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

**C#**

```csharp
public TThird Third
```

**Visual Basic (Declaration)**

```vbnet
Public Third As TThird
```

**Visual C++**

```cpp
public:
TThird Third
```
See Also

`Triple<(Of <TFirst, TSecond, TThird>)>` Structure
`Wintellect.PowerCollections` Namespace

Send [comments](#) about this topic to Microsoft.
Triple<(Of <TFirst, TSecond, TThird>)> Methods

See Also

☑ Include Inherited Members ☐ Include Protected Members
☐ .NET Compact Framework Members Only
☐ XNA Framework Members Only

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Syntax

### C#

```csharp
[SerializableAttribute]
public struct Triple<TFirst, TSecond, TThird>
```

### Visual Basic (Declaration)

```vbnet
<SerializableAttribute> _
Public Structure Triple(Of TFirst, TSecond, TThird)
```

### Visual C++

```cpp
[SerializableAttribute]
generic<typename TFirst, typename TSecond, typename TThird>
public value class Triple
```
Type Parameters

TFirst
TSecond
TThird

The type exposes the following methods.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CompareTo</strong></td>
<td>Compares this triple to another triple of the same type. The triples are compared by using the IComparable&lt;T&gt; or IComparable interface on TFirst, TSecond, and TThird. The triples are compared by their first elements first, if their first elements are equal, then they are compared by their second elements. If their second elements are also equal, then they are compared by their third elements. If TFirst, TSecond, or TThird does not implement IComparable&lt;T&gt; or IComparable, then an NotSupportedException is thrown, because the triples cannot be compared.</td>
</tr>
<tr>
<td><strong>Equality</strong></td>
<td>Determines if two triples are equal. Two triples are equal if the all three elements compare equal using IComparable&lt;T&gt;.Equals or object.Equals.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Returns a hash code for the triple, suitable for use in a hashtable or other hashed collection. Two triples that compare equal (using Equals) will have the same hash code. The hash code for the triple is derived by combining the hash codes for each of the two elements of the triple. (Overrides ValueType..::.GetHashCode()().</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Inequality</strong></td>
<td>Determines if two triples are not equal. Two triples are equal if the all three elements compare equal using IComparable&lt;T&gt;.Equals or object.Equals. Returns a string representation of the triple. The string representation of the triple is of the form: First: {0}, Second: {1}, Third: {2} where {0} is the result of First.ToString(),</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td></td>
</tr>
</tbody>
</table>
{1} is the result of Second.ToString(), and {2} is the result of Third.ToString() (or "null" if they are null.) (Overrides ValueType...::ToString().)
## Protected Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>. (Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>
### Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| IComparable::CompareTo | Compares this triple to another triple of the same type. The triples are compared by using the IComparable<T> or IComparable interface on TFirst, TSecond, and TThird. The triples are compared by their first elements first, if their first elements are equal, then they are compared by their second elements. If their second elements are also equal, then they are compared by their third elements.  
If TFirst, TSecond, or TThird does not implement IComparable<T> or IComparable, then an NotSupportedException is thrown, because the triples cannot be compared. |
See Also

Triple<(Of <TFirst, TSecond, TThird>)}
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Triple<(Of <TFirst, TSecond, TThird>)>...::CompareTo Method

See Also

- Visual Basic (Declaration)
- Visual Basic (Usage)
- C#
- Visual C++
- J#
- JScript
- XAML

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Compares this triple to another triple of the same type. The triples are compared by using the IComparable<T> or IComparable interface on TFirst, TSecond, and TThird. The triples are compared by their first elements first, if their first elements are equal, then they are compared by their second elements. If their second elements are also equal, then they are compared by their third elements.

If TFirst, TSecond, or TThird does not implement IComparable<T> or IComparable, then a NotSupportedException is thrown, because the triples cannot be compared.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
- Syntax

C#

```csharp
public int CompareTo(
    Triple<TFirst, TSecond, TThird> other
)
```

Visual Basic (Declaration)

```vbnet
Public Function CompareTo ( _
    other As Triple(Of TFirst, TSecond, TThird) _
) As Integer
```

Visual C++

```cpp
public:
    virtual int CompareTo ( 
        Triple<TFirst, TSecond, TThird> other 
    ) sealed
```

Parameters

other

```csharp
    Triple<(Of <TFirst, TSecond, TThird>)>
```

The triple to compare to.

Return Value

An integer indicating how this triple compares to other. Less than zero indicates this triple is less than other. Zero indicate this triple is equals to other. Greater than zero indicates this triple is greater than other.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::NotSupportedException</code></td>
<td>Either <code>FirstSecond</code>, <code>TSecond</code>, or <code>TThird</code> is not comparable via the <code>IComparable&lt;T&gt;</code> or <code>IComparable</code> interfaces.</td>
</tr>
</tbody>
</table>
See Also

Triple<(Of <TFirst, TSecond, TThird>)> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if two triples are equal. Two triples are equal if the all three elements compare equal using IComparable<T>.Equals or object.Equals.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public static bool operator ==(
    Triple<TFirst, TSecond, TThird> pair1,
    Triple<TFirst, TSecond, TThird> pair2
)
```

Visual Basic (Declaration)

```vbnet
Public Shared Operator = ( _
    pair1 As Triple(Of TFirst, TSecond, TThird), _
    pair2 As Triple(Of TFirst, TSecond, TThird) _
) As Boolean
```

Visual C++

```cpp
public:
static bool operator == ( 
    Triple<TFirst, TSecond, TThird> pair1,
    Triple<TFirst, TSecond, TThird> pair2
)
```

Parameters

pair1

```
    Triple<Of <TFirst, TSecond, TThird>>
```
First triple to compare.

pair2

```
    Triple<Of <TFirst, TSecond, TThird>>
```
Second triple to compare.

Return Value

True if the triples are equal. False if the triples are not equal.
See Also

Triple<(Of <TFirst, TSecond, TThird>)> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Triple<(Of <TFirst, TSecond, TThird>)>::Equals Method

See Also  Members

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals(Object)</strong></td>
<td>Determines if this triple is equal to another object. The triple is equal to another object if that object is a Triple, all element types are the same, and the all three elements compare equal using object.Equals. (Overrides ValueType...::Equals(Object).)</td>
</tr>
<tr>
<td><strong>Equals(Triple&lt;TFirst, TSecond, TThird&gt;)</strong></td>
<td>Determines if this triple is equal to another triple. Two triples are equal if the all three elements compare equal using IComparable&lt;T&gt;.Equals or object.Equals.</td>
</tr>
</tbody>
</table>
See Also

`Triple(Of <TFirst, TSecond, TThird>)` Structure
`Triple(Of <TFirst, TSecond, TThird>)` Members
`Wintellect.PowerCollections` Namespace

Send comments about this topic to Microsoft.
Determines if this triple is equal to another object. The triple is equal to another object if that object is a Triple, all element types are the same, and the all three elements compare equal using object.Equals.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override bool Equals(
    Object obj
)

Visual Basic (Declaration)

Public Overrides Function Equals ( _
    obj As Object _
) As Boolean

Visual C++

public:
    virtual bool Equals ( 
        Object^ obj
    ) override

Parameters

obj

Object
    Object to compare for equality.

Return Value

True if the objects are equal. False if the objects are not equal.
See Also

Triple(Of TFirst, TSecond, TThird) Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if this triple is equal to another triple. Two triples are equal if the all three elements compare equal using IComparable<T>.Equals or object.Equals.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

```csharp
public bool Equals(
    Triple<TFirst, TSecond, TThird> other
)
```

Visual Basic (Declaration)

```vbnet
Public Function Equals (_
    other As Triple(Of TFirst, TSecond, TThird) _
) As Boolean
```

Visual C++

```cpp
public:
    bool Equals (
        Triple<TFirst, TSecond, TThird> other
    )
```

Parameters

other

`Triple<(Of <TFirst, TSecond, TThird>)>`

Triple to compare with for equality.

Return Value

True if the triples are equal. False if the triples are not equal.
See Also

Triple(Of <TFirst, TSecond, TThird>) Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Returns a hash code for the triple, suitable for use in a hash-table or other hashed collection. Two triples that compare equal (using Equals) will have the same hash code. The hash code for the triple is derived by combining the hash codes for each of the two elements of the triple.

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#
public override int GetHashCode()

Visual Basic (Declaration)
Public Overrides Function GetHashCode As Integer

Visual C++
public:
virtual int GetHashCode () override

Return Value

The hash code.
See Also

Triple<(Of <TFirst, TSecond, TThird>)> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Determines if two triples are not equal. Two triples are equal if the all three elements compare equal using IComparable<T>.Equals or object.Equals.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
### Syntax

**C#**

```csharp
public static bool operator !=(
    Triple<TFirst, TSecond, TThird> pair1,
    Triple<TFirst, TSecond, TThird> pair2
)
```

**Visual Basic (Declaration)**

```vbnet
Public Shared Operator <> ( _
    pair1 As Triple(Of TFirst, TSecond, TThird), _
    pair2 As Triple(Of TFirst, TSecond, TThird) _
) As Boolean
```

**Visual C++**

```cpp
public:
    static bool operator != ( 
        Triple<TFirst, TSecond, TThird> pair1,
        Triple<TFirst, TSecond, TThird> pair2
    )
```

### Parameters

- **pair1**
  ```csharp
  Triple<(Of <TFirst, TSecond, TThird>)>
  First triple to compare.
  ```

- **pair2**
  ```csharp
  Triple<(Of <TFirst, TSecond, TThird>)>
  Second triple to compare.
  ```

### Return Value

True if the triples are not equal. False if the triples are equal.
**See Also**

Triple(Of TFirst, TSecond, TThird) Structure  
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Triple<(Of <TFirst, TSecond, TThird>)>...::System.IComparable.CompareTo Method

See Also

[This topic is pre-release documentation and is subject to change in future releases. Blank topics are included as placeholders.]

Compares this triple to another triple of the same type. The triples are compared by using the IComparable<T> or IComparable interface on TFirst, TSecond, and TThird. The triples are compared by their first elements first, if their first elements are equal, then they are compared by their second elements. If their second elements are also equal, then they are compared by their third elements.

If TFirst, TSecond, or TThird does not implement IComparable<T> or IComparable, then an NotSupportedException is thrown, because the triples cannot be compared.

Namespace: Wintellect.PowerCollections
Assembly: PowerCollections (in PowerCollections.dll)
Syntax

C#

private int IComparable.CompareTo(
    Object obj
)

Visual Basic (Declaration)

Private Function System.IComparable.CompareTo ( _
    obj As Object _
) As Integer Implements IComparable.CompareTo

Visual C++

private:
virtual int System.IComparable.CompareTo ( _
    Object^ obj
) sealed = IComparable::CompareTo

Parameters

obj
    Object
    The triple to compare to.

Return Value

An integer indicating how this triple compares to obj. Less than zero indicates this triple is less than obj. Zero indicate this triple is equals to obj. Greater than zero indicates this triple is greater than obj.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentException</code></td>
<td>obj is not of the correct type.</td>
</tr>
<tr>
<td><code>System::NotSupportedException</code></td>
<td>Either FirstSecond, TSecond, or TThird is not comparable via the IComparable&lt;T&gt; or IComparable interfaces.</td>
</tr>
</tbody>
</table>
See Also

Triple<(Of <TFirst, TSecond, TThird>)> Structure
Wintellect.PowerCollections Namespace

Send comments about this topic to Microsoft.
Returns a string representation of the triple. The string representation of the triple is of the form: First: \{0\}, Second: \{1\}, Third: \{2\} where \{0\} is the result of First.ToString(), \{1\} is the result of Second.ToString(), and \{2\} is the result of Third.ToString() (or "null" if they are null.)

**Namespace:** Wintellect.PowerCollections  
**Assembly:** PowerCollections (in PowerCollections.dll)
Syntax

C#

public override string ToString()

Visual Basic (Declaration)

Public Overrides Function ToString As String

Visual C++

public:
virtual String^ ToString () override

Return Value

The string representation of the triple.
See Also

**Triple(Of TFirst, TSecond, TThird)>** Structure
**Wintellect.PowerCollections** Namespace

Send [comments](mailto:feedback@microsoft.com) about this topic to Microsoft.