List<T> Class

Reference

Definition

Namespace: System.Collections.Generic

Assembly: System.Collections.dll

Source: List.cs

Represents a strongly typed list of objects that can be accessed by index. Provides methods to search, sort, and manipulate lists.

C#

```
public class List<T> : System.Collections.Generic.ICollection<T>,
System.Collections.Generic.IEnumerable<T>, System.Collections.Generic.IList<T>,
System.Collections.Generic.IReadOnlyCollection<T>,
System.Collections.Generic.IReadOnlyList<T>, System.Collections.IList
```

Type Parameters

T

The type of elements in the list.

Inheritance Object → List<T>

Derived System.Data.Services.ExpandSegmentCollection
System.Workflow.Activities.OperationParameterInfoCollection
System.Workflow.Activities.WorkflowRoleCollection
System.Workflow.ComponentModel.ActivityCollection
System.Workflow.ComponentModel.Design.ActivityDesignerGlyphCollection
More...

Implements ICollection<T> , IEnumerable<T> , IList<T> , IReadOnlyCollection<T> , IReadOnlyList<T> , ICollection , IEnumerable , IList

Remarks

For more information about this API, see Supplemental API remarks for List<T>.

Constructors

Expand table

List<T>()	Initializes a new instance of the List<T> class that is empty and has the default initial capacity.
List<T>(IEnumerable<T>)	Initializes a new instance of the List<T> class that contains elements copied from the specified collection and has sufficient capacity to accommodate the number of elements copied.

List<T>(Int32)	Initializes a new instance of the List<T> class that is empty and has the specified initial capacity.

Properties

Expand table

Capacity	Gets or sets the total number of elements the internal data structure can hold without resizing.
Count	Gets the number of elements contained in the List<T>.
Item[Int32]	Gets or sets the element at the specified index.

Methods

Expand table

Add(T)	Adds an object to the end of the List<T>.
AddRange(IEnumerable<T>)	Adds the elements of the specified collection to the end of the List<T>.
AsReadOnly()	Returns a read-only ReadOnlyCollection<T> wrapper for the current collection.
BinarySearch(Int32, Int32, T, IComparer<T>)	Searches a range of elements in the sorted List<T> for an element using the specified comparer and returns the zero-based index of the element.
BinarySearch(T)	Searches the entire sorted List<T> for an element using the default comparer and returns the zero-based index of the element.
BinarySearch(T, IComparer<T>)	Searches the entire sorted List<T> for an element using the specified comparer and returns the zero-based index of the element.
Clear()	Removes all elements from the List<T>.
Contains(T)	Determines whether an element is in the List<T>.
ConvertAll<TOutput>(Converter<T,TOutput>)	Converts the elements in the current List<T> to another type, and returns a list containing the converted elements.
CopyTo(Int32, T[], Int32, Int32)	Copies a range of elements from the List<T> to a compatible one-dimensional array, starting at the specified index of the target array.
CopyTo(T[])	Copies the entire List<T> to a compatible one-dimensional array, starting at the beginning of the target array.
CopyTo(T[], Int32)	Copies the entire List<T> to a compatible one-dimensional array, starting at the specified index of the target array.
EnsureCapacity(Int32)	Ensures that the capacity of this list is at least the specified `capacity`. If the current capacity is less than `capacity`, it is increased to at least the specified `capacity`.
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
Exists(Predicate<T>)	Determines whether the List<T> contains elements that match the conditions defined by the specified predicate.
Find(Predicate<T>)	Searches for an element that matches the conditions defined by the specified predicate, and returns the first occurrence within the entire List<T>.
FindAll(Predicate<T>)	Retrieves all the elements that match the conditions defined by the specified predicate.

FindIndex(Int32, Int32, Predicate<T>)	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the first occurrence within the range of elements in the List<T> that starts at the specified index and contains the specified number of elements.
FindIndex(Int32, Predicate<T>)	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the first occurrence within the range of elements in the List<T> that extends from the specified index to the last element.
FindIndex(Predicate<T>)	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the first occurrence within the entire List<T>.
FindLast(Predicate<T>)	Searches for an element that matches the conditions defined by the specified predicate, and returns the last occurrence within the entire List<T>.
FindLastIndex(Int32, Int32, Predicate<T>)	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the last occurrence within the range of elements in the List<T> that contains the specified number of elements and ends at the specified index.
FindLastIndex(Int32, Predicate<T>)	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the last occurrence within the range of elements in the List<T> that extends from the first element to the specified index.
FindLastIndex(Predicate<T>)	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the last occurrence within the entire List<T>.
ForEach(Action<T>)	Performs the specified action on each element of the List<T>.
GetEnumerator()	Returns an enumerator that iterates through the List<T>.
GetHashCode()	Serves as the default hash function. (Inherited from Object)
GetRange(Int32, Int32)	Creates a shallow copy of a range of elements in the source List<T>.
GetType()	Gets the Type of the current instance. (Inherited from Object)
IndexOf(T)	Searches for the specified object and returns the zero-based index of the first occurrence within the entire List<T>.
IndexOf(T, Int32)	Searches for the specified object and returns the zero-based index of the first occurrence within the range of elements in the List<T> that extends from the specified index to the last element.
IndexOf(T, Int32, Int32)	Searches for the specified object and returns the zero-based index of the first occurrence within the range of elements in the List<T> that starts at the specified index and contains the specified number of elements.
Insert(Int32, T)	Inserts an element into the List<T> at the specified index.
InsertRange(Int32, IEnumerable<T>)	Inserts the elements of a collection into the List<T> at the specified index.
LastIndexOf(T)	Searches for the specified object and returns the zero-based index of the last occurrence within the entire List<T>.
LastIndexOf(T, Int32)	Searches for the specified object and returns the zero-based index of the last occurrence within the range of elements in the List<T> that extends from the first element to the specified index.
LastIndexOf(T, Int32, Int32)	Searches for the specified object and returns the zero-based index of the last occurrence within the range of elements in the List<T> that contains the specified number of elements and ends at the specified index.
MemberwiseClone()	Creates a shallow copy of the current Object.

	(Inherited from Object)
Remove(T)	Removes the first occurrence of a specific object from the List<T>.
RemoveAll(Predicate<T>)	Removes all the elements that match the conditions defined by the specified predicate.
RemoveAt(Int32)	Removes the element at the specified index of the List<T>.
RemoveRange(Int32, Int32)	Removes a range of elements from the List<T>.
Reverse()	Reverses the order of the elements in the entire List<T>.
Reverse(Int32, Int32)	Reverses the order of the elements in the specified range.
Slice(Int32, Int32)	Creates a shallow copy of a range of elements in the source List<T>.
Sort()	Sorts the elements in the entire List<T> using the default comparer.
Sort(Comparison<T>)	Sorts the elements in the entire List<T> using the specified Comparison<T>.
Sort(IComparer<T>)	Sorts the elements in the entire List<T> using the specified comparer.
Sort(Int32, Int32, IComparer<T>)	Sorts the elements in a range of elements in List<T> using the specified comparer.
ToArray()	Copies the elements of the List<T> to a new array.
ToString()	Returns a string that represents the current object. (Inherited from Object)
TrimExcess()	Sets the capacity to the actual number of elements in the List<T>, if that number is less than a threshold value.
TrueForAll(Predicate<T>)	Determines whether every element in the List<T> matches the conditions defined by the specified predicate.

Explicit Interface Implementations

Expand table

ICollection.CopyTo(Array, Int32)	Copies the elements of the ICollection to an Array, starting at a particular Array index.
ICollection.IsSynchronized	Gets a value indicating whether access to the ICollection is synchronized (thread safe).
ICollection.SyncRoot	Gets an object that can be used to synchronize access to the ICollection.
ICollection<T>.IsReadOnly	Gets a value indicating whether the ICollection<T> is read-only.
IEnumerable.GetEnumerator()	Returns an enumerator that iterates through a collection.
IEnumerable<T>.GetEnumerator()	Returns an enumerator that iterates through a collection.
IList.Add(Object)	Adds an item to the IList.
IList.Contains(Object)	Determines whether the IList contains a specific value.
IList.IndexOf(Object)	Determines the index of a specific item in the IList.
IList.Insert(Int32, Object)	Inserts an item to the IList at the specified index.
IList.IsFixedSize	Gets a value indicating whether the IList has a fixed size.
IList.IsReadOnly	Gets a value indicating whether the IList is read-only.

IList.Item[Int32]	Gets or sets the element at the specified index.
IList.Remove(Object)	Removes the first occurrence of a specific object from the IList.

Extension Methods

Expand table

ToFrozenDictionary<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IEqualityComparer<TKey>)	Creates a FrozenDictionary<TKey,TValue> from an IEnumerable<T> according to specified key selector function.
ToFrozenDictionary<TSource,TKey,TElement>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>, IEqualityComparer<TKey>)	Creates a FrozenDictionary<TKey,TValue> from an IEnumerable<T> according to specified key selector and element selector functions.
ToFrozenSet<T>(IEnumerable<T>, IEqualityComparer<T>)	Creates a FrozenSet<T> with the specified values.
AddRange<T>(List<T>, ReadOnlySpan<T>)	Adds the elements of the specified span to the end of the List<T>.
AsReadOnly<T>(IList<T>)	Returns a read-only ReadOnlyCollection<T> wrapper for the specified list.
CopyTo<T>(List<T>, Span<T>)	Copies the entire List<T> to a span.
InsertRange<T>(List<T>, Int32, ReadOnlySpan<T>)	Inserts the elements of a span into the List<T> at the specified index.
ToImmutableArray<TSource>(IEnumerable<TSource>)	Creates an immutable array from the specified collection.
ToImmutableDictionary<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)	Constructs an immutable dictionary from an existing collection of elements, applying a transformation function to the source keys.
ToImmutableDictionary<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IEqualityComparer<TKey>)	Constructs an immutable dictionary based on some transformation of a sequence.
ToImmutableDictionary<TSource,TKey,TValue>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>)	Enumerates and transforms a sequence, and produces an immutable dictionary of its contents.
ToImmutableDictionary<TSource,TKey,TValue>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IEqualityComparer<TKey>)	Enumerates and transforms a sequence, and produces an immutable dictionary of its contents by using the specified key comparer.
ToImmutableDictionary<TSource,TKey,TValue>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IEqualityComparer<TKey>, IEqualityComparer<TValue>)	Enumerates and transforms a sequence, and produces an immutable dictionary of its contents by using the specified key and value comparers.
ToImmutableHashSet<TSource>(IEnumerable<TSource>)	Enumerates a sequence and produces an immutable hash set of its contents.
ToImmutableHashSet<TSource>(IEnumerable<TSource>, IEquality Comparer<TSource>)	Enumerates a sequence, produces an immutable hash set of its contents, and uses the specified equality comparer for the set type.
ToImmutableList<TSource>(IEnumerable<TSource>)	Enumerates a sequence and produces an immutable list of its contents.

ToImmutableSortedDictionary<TSource,TKey,TValue>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>)	Enumerates and transforms a sequence, and produces an immutable sorted dictionary of its contents.
ToImmutableSortedDictionary<TSource,TKey,TValue>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IComparer<TKey>)	Enumerates and transforms a sequence, and produces an immutable sorted dictionary of its contents by using the specified key comparer.
ToImmutableSortedDictionary<TSource,TKey,TValue>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IComparer<TKey>, IEqualityComparer<TValue>)	Enumerates and transforms a sequence, and produces an immutable sorted dictionary of its contents by using the specified key and value comparers.
ToImmutableSortedSet<TSource>(IEnumerable<TSource>)	Enumerates a sequence and produces an immutable sorted set of its contents.
ToImmutableSortedSet<TSource>(IEnumerable<TSource>, IComparer<TSource>)	Enumerates a sequence, produces an immutable sorted set of its contents, and uses the specified comparer.
CopyToDataTable<T>(IEnumerable<T>)	Returns a DataTable that contains copies of the DataRow objects, given an input IEnumerable<T> object where the generic parameter `T` is DataRow.
CopyToDataTable<T>(IEnumerable<T>, DataTable, LoadOption)	Copies DataRow objects to the specified DataTable, given an input IEnumerable<T> object where the generic parameter `T` is DataRow.
CopyToDataTable<T>(IEnumerable<T>, DataTable, LoadOption, FillErrorEventHandler)	Copies DataRow objects to the specified DataTable, given an input IEnumerable<T> object where the generic parameter `T` is DataRow.
Aggregate<TSource>(IEnumerable<TSource>, Func<TSource,TSource,TSource>)	Applies an accumulator function over a sequence.
Aggregate<TSource,TAccumulate>(IEnumerable<TSource>, TAccumulate, Func<TAccumulate,TSource,TAccumulate>)	Applies an accumulator function over a sequence. The specified seed value is used as the initial accumulator value.
Aggregate<TSource,TAccumulate,TResult>(IEnumerable<TSource>, TAccumulate, Func<TAccumulate,TSource,TAccumulate>, Func<TAccumulate,TResult>)	Applies an accumulator function over a sequence. The specified seed value is used as the initial accumulator value, and the specified function is used to select the result value.
All<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Determines whether all elements of a sequence satisfy a condition.
Any<TSource>(IEnumerable<TSource>)	Determines whether a sequence contains any elements.
Any<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Determines whether any element of a sequence satisfies a condition.
Append<TSource>(IEnumerable<TSource>, TSource)	Appends a value to the end of the sequence.
AsEnumerable<TSource>(IEnumerable<TSource>)	Returns the input typed as IEnumerable<T>.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Decimal>)	Computes the average of a sequence of Decimal values that are obtained by invoking a transform function on each element of the input sequence.

Average<TSource>(IEnumerable<TSource>, Func<TSource,Double>)	Computes the average of a sequence of Double values that are obtained by invoking a transform function on each element of the input sequence.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Int32>)	Computes the average of a sequence of Int32 values that are obtained by invoking a transform function on each element of the input sequence.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Int64>)	Computes the average of a sequence of Int64 values that are obtained by invoking a transform function on each element of the input sequence.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Decimal>>)	Computes the average of a sequence of nullable Decimal values that are obtained by invoking a transform function on each element of the input sequence.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Double>>)	Computes the average of a sequence of nullable Double values that are obtained by invoking a transform function on each element of the input sequence.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int32>>)	Computes the average of a sequence of nullable Int32 values that are obtained by invoking a transform function on each element of the input sequence.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int64>>)	Computes the average of a sequence of nullable Int64 values that are obtained by invoking a transform function on each element of the input sequence.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Single>>)	Computes the average of a sequence of nullable Single values that are obtained by invoking a transform function on each element of the input sequence.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Single>)	Computes the average of a sequence of Single values that are obtained by invoking a transform function on each element of the input sequence.
Cast<TResult>(IEnumerable)	Casts the elements of an IEnumerable to the specified type.
Chunk<TSource>(IEnumerable<TSource>, Int32)	Splits the elements of a sequence into chunks of size at most `size`.
Concat<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)	Concatenates two sequences.
Contains<TSource>(IEnumerable<TSource>, TSource)	Determines whether a sequence contains a specified element by using the default equality comparer.
Contains<TSource>(IEnumerable<TSource>, TSource, IEqualityComparer<TSource>)	Determines whether a sequence contains a specified element by using a specified IEqualityComparer<T>.
Count<TSource>(IEnumerable<TSource>)	Returns the number of elements in a sequence.
Count<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Returns a number that represents how many elements in the specified sequence satisfy a condition.

DefaultIfEmpty<TSource>(IEnumerable<TSource>)	Returns the elements of the specified sequence or the type parameter's default value in a singleton collection if the sequence is empty.
DefaultIfEmpty<TSource>(IEnumerable<TSource>, TSource)	Returns the elements of the specified sequence or the specified value in a singleton collection if the sequence is empty.
Distinct<TSource>(IEnumerable<TSource>)	Returns distinct elements from a sequence by using the default equality comparer to compare values.
Distinct<TSource>(IEnumerable<TSource>, IEqualityComparer<TSource>)	Returns distinct elements from a sequence by using a specified IEqualityComparer<T> to compare values.
DistinctBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)	Returns distinct elements from a sequence according to a specified key selector function.
DistinctBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IEqualityComparer<TKey>)	Returns distinct elements from a sequence according to a specified key selector function and using a specified comparer to compare keys.
ElementAt<TSource>(IEnumerable<TSource>, Index)	Returns the element at a specified index in a sequence.
ElementAt<TSource>(IEnumerable<TSource>, Int32)	Returns the element at a specified index in a sequence.
ElementAtOrDefault<TSource>(IEnumerable<TSource>, Index)	Returns the element at a specified index in a sequence or a default value if the index is out of range.
ElementAtOrDefault<TSource>(IEnumerable<TSource>, Int32)	Returns the element at a specified index in a sequence or a default value if the index is out of range.
Except<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)	Produces the set difference of two sequences by using the default equality comparer to compare values.
Except<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)	Produces the set difference of two sequences by using the specified IEqualityComparer<T> to compare values.
ExceptBy<TSource,TKey>(IEnumerable<TSource>, IEnumerable<TKey>, Func<TSource,TKey>)	Produces the set difference of two sequences according to a specified key selector function.
ExceptBy<TSource,TKey>(IEnumerable<TSource>, IEnumerable<TKey>, Func<TSource,TKey>, IEqualityComparer<TKey>)	Produces the set difference of two sequences according to a specified key selector function.
First<TSource>(IEnumerable<TSource>)	Returns the first element of a sequence.
First<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Returns the first element in a sequence that satisfies a specified condition.
FirstOrDefault<TSource>(IEnumerable<TSource>)	Returns the first element of a sequence, or a default value if the sequence contains no elements.
FirstOrDefault<TSource>(IEnumerable<TSource>, TSource)	Returns the first element of a sequence, or a specified default value if the sequence contains no elements.

FirstOrDefault<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Returns the first element of the sequence that satisfies a condition or a default value if no such element is found.
FirstOrDefault<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>, TSource)	Returns the first element of the sequence that satisfies a condition, or a specified default value if no such element is found.
GroupBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)	Groups the elements of a sequence according to a specified key selector function.
GroupBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IEqualityComparer<TKey>)	Groups the elements of a sequence according to a specified key selector function and compares the keys by using a specified comparer.
GroupBy<TSource,TKey,TElement>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>)	Groups the elements of a sequence according to a specified key selector function and projects the elements for each group by using a specified function.
GroupBy<TSource,TKey,TElement>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>, IEqualityComparer<TKey>)	Groups the elements of a sequence according to a key selector function. The keys are compared by using a comparer and each group's elements are projected by using a specified function.
GroupBy<TSource,TKey,TResult>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TKey,IEnumerable<TSource>,TResult>)	Groups the elements of a sequence according to a specified key selector function and creates a result value from each group and its key.
GroupBy<TSource,TKey,TResult>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TKey,IEnumerable<TSource>,TResult>, IEqualityComparer<TKey>)	Groups the elements of a sequence according to a specified key selector function and creates a result value from each group and its key. The keys are compared by using a specified comparer.
GroupBy<TSource,TKey,TElement,TResult>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>, Func<TKey,IEnumerable<TElement>,TResult>)	Groups the elements of a sequence according to a specified key selector function and creates a result value from each group and its key. The elements of each group are projected by using a specified function.
GroupBy<TSource,TKey,TElement,TResult>(IEnumerable<TSource>, Func<TSource, TKey>, Func<TSource,TElement>, Func<TKey,IEnumerable<TElement>, TResult>, IEqualityComparer<TKey>)	Groups the elements of a sequence according to a specified key selector function and creates a result value from each group and its key. Key values are compared by using a specified comparer, and the elements of each group are projected by using a specified function.
GroupJoin<TOuter,TInner,TKey,TResult>(IEnumerable<TOuter>, IEnumerable<TInner>, Func<TOuter,TKey>, Func<TInner,TKey>, Func<TOuter,IEnumerable<TInner>, TResult>)	Correlates the elements of two sequences based on equality of keys and groups the results. The default equality comparer is used to compare keys.
GroupJoin<TOuter,TInner,TKey,TResult>(IEnumerable<TOuter>, IEnumerable<TInner>, Func<TOuter,TKey>, Func<TInner,TKey>, Func<TOuter,IEnumerable<TInner>, TResult>, IEqualityComparer<TKey>)	Correlates the elements of two sequences based on key equality and groups the results. A specified IEqualityComparer<T> is used to compare keys.
Intersect<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)	Produces the set intersection of two sequences by using the default equality comparer to compare values.
Intersect<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)	Produces the set intersection of two sequences by using the specified IEqualityComparer<T> to

	compare values.
IntersectBy<TSource,TKey>(IEnumerable<TSource>, IEnumerable<TKey>, Func<TSource,TKey>)	Produces the set intersection of two sequences according to a specified key selector function.
IntersectBy<TSource,TKey>(IEnumerable<TSource>, IEnumerable<TKey>, Func<TSource,TKey>, IEqualityComparer<TKey>)	Produces the set intersection of two sequences according to a specified key selector function.
Join<TOuter,TInner,TKey,TResult>(IEnumerable<TOuter>, IEnumerable<TInner>, Func<TOuter,TKey>, Func<TInner,TKey>, Func<TOuter,TInner,TResult>)	Correlates the elements of two sequences based on matching keys. The default equality comparer is used to compare keys.
Join<TOuter,TInner,TKey,TResult>(IEnumerable<TOuter>, IEnumerable<TInner>, Func<TOuter,TKey>, Func<TInner,TKey>, Func<TOuter,TInner,TResult>, IEqualityComparer<TKey>)	Correlates the elements of two sequences based on matching keys. A specified IEqualityComparer<T> is used to compare keys.
Last<TSource>(IEnumerable<TSource>)	Returns the last element of a sequence.
Last<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Returns the last element of a sequence that satisfies a specified condition.
LastOrDefault<TSource>(IEnumerable<TSource>)	Returns the last element of a sequence, or a default value if the sequence contains no elements.
LastOrDefault<TSource>(IEnumerable<TSource>, TSource)	Returns the last element of a sequence, or a specified default value if the sequence contains no elements.
LastOrDefault<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Returns the last element of a sequence that satisfies a condition or a default value if no such element is found.
LastOrDefault<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>, TSource)	Returns the last element of a sequence that satisfies a condition, or a specified default value if no such element is found.
LongCount<TSource>(IEnumerable<TSource>)	Returns an Int64 that represents the total number of elements in a sequence.
LongCount<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Returns an Int64 that represents how many elements in a sequence satisfy a condition.
Max<TSource>(IEnumerable<TSource>)	Returns the maximum value in a generic sequence.
Max<TSource>(IEnumerable<TSource>, IComparer<TSource>)	Returns the maximum value in a generic sequence.
Max<TSource>(IEnumerable<TSource>, Func<TSource,Decimal>)	Invokes a transform function on each element of a sequence and returns the maximum Decimal value.
Max<TSource>(IEnumerable<TSource>, Func<TSource,Double>)	Invokes a transform function on each element of a sequence and returns the maximum Double value.
Max<TSource>(IEnumerable<TSource>, Func<TSource,Int32>)	Invokes a transform function on each element of a sequence and returns the maximum Int32 value.
Max<TSource>(IEnumerable<TSource>, Func<TSource,Int64>)	Invokes a transform function on each element of a sequence and returns the maximum Int64 value.

Max<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Decimal>>)	Invokes a transform function on each element of a sequence and returns the maximum nullable Decimal value.
Max<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Double>>)	Invokes a transform function on each element of a sequence and returns the maximum nullable Double value.
Max<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int32>>)	Invokes a transform function on each element of a sequence and returns the maximum nullable Int32 value.
Max<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int64>>)	Invokes a transform function on each element of a sequence and returns the maximum nullable Int64 value.
Max<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Single>>)	Invokes a transform function on each element of a sequence and returns the maximum nullable Single value.
Max<TSource>(IEnumerable<TSource>, Func<TSource,Single>)	Invokes a transform function on each element of a sequence and returns the maximum Single value.
Max<TSource,TResult>(IEnumerable<TSource>, Func<TSource,TResult>)	Invokes a transform function on each element of a generic sequence and returns the maximum resulting value.
MaxBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)	Returns the maximum value in a generic sequence according to a specified key selector function.
MaxBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IComparer<TKey>)	Returns the maximum value in a generic sequence according to a specified key selector function and key comparer.
Min<TSource>(IEnumerable<TSource>)	Returns the minimum value in a generic sequence.
Min<TSource>(IEnumerable<TSource>, IComparer<TSource>)	Returns the minimum value in a generic sequence.
Min<TSource>(IEnumerable<TSource>, Func<TSource,Decimal>)	Invokes a transform function on each element of a sequence and returns the minimum Decimal value.
Min<TSource>(IEnumerable<TSource>, Func<TSource,Double>)	Invokes a transform function on each element of a sequence and returns the minimum Double value.
Min<TSource>(IEnumerable<TSource>, Func<TSource,Int32>)	Invokes a transform function on each element of a sequence and returns the minimum Int32 value.
Min<TSource>(IEnumerable<TSource>, Func<TSource,Int64>)	Invokes a transform function on each element of a sequence and returns the minimum Int64 value.
Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Decimal>>)	Invokes a transform function on each element of a sequence and returns the minimum nullable Decimal value.
Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Double>>)	Invokes a transform function on each element of a sequence and returns the minimum nullable

	Double value.
Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int32>>)	Invokes a transform function on each element of a sequence and returns the minimum nullable Int32 value.
Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int64>>)	Invokes a transform function on each element of a sequence and returns the minimum nullable Int64 value.
Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Single>>)	Invokes a transform function on each element of a sequence and returns the minimum nullable Single value.
Min<TSource>(IEnumerable<TSource>, Func<TSource,Single>)	Invokes a transform function on each element of a sequence and returns the minimum Single value.
Min<TSource,TResult>(IEnumerable<TSource>, Func<TSource,TResult>)	Invokes a transform function on each element of a generic sequence and returns the minimum resulting value.
MinBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)	Returns the minimum value in a generic sequence according to a specified key selector function.
MinBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IComparer<TKey>)	Returns the minimum value in a generic sequence according to a specified key selector function and key comparer.
OfType<TResult>(IEnumerable)	Filters the elements of an IEnumerable based on a specified type.
Order<T>(IEnumerable<T>)	Sorts the elements of a sequence in ascending order.
Order<T>(IEnumerable<T>, IComparer<T>)	Sorts the elements of a sequence in ascending order.
OrderBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)	Sorts the elements of a sequence in ascending order according to a key.
OrderBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IComparer<TKey>)	Sorts the elements of a sequence in ascending order by using a specified comparer.
OrderByDescending<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)	Sorts the elements of a sequence in descending order according to a key.
OrderByDescending<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IComparer<TKey>)	Sorts the elements of a sequence in descending order by using a specified comparer.
OrderDescending<T>(IEnumerable<T>)	Sorts the elements of a sequence in descending order.
OrderDescending<T>(IEnumerable<T>, IComparer<T>)	Sorts the elements of a sequence in descending order.
Prepend<TSource>(IEnumerable<TSource>, TSource)	Adds a value to the beginning of the sequence.
Reverse<TSource>(IEnumerable<TSource>)	Inverts the order of the elements in a sequence.
Select<TSource,TResult>(IEnumerable<TSource>, Func<TSource,TResult>)	Projects each element of a sequence into a new form.
Select<TSource,TResult>(IEnumerable<TSource>,	Projects each element of a sequence into a new

Func<TSource,Int32,TResult>)	form by incorporating the element's index.
SelectMany<TSource,TResult>(IEnumerable<TSource>, Func<TSource,IEnumerable<TResult>>)	Projects each element of a sequence to an IEnumerable<T> and flattens the resulting sequences into one sequence.
SelectMany<TSource,TResult>(IEnumerable<TSource>, Func<TSource,Int32,IEnumerable<TResult>>)	Projects each element of a sequence to an IEnumerable<T>, and flattens the resulting sequences into one sequence. The index of each source element is used in the projected form of that element.
SelectMany<TSource,TCollection,TResult>(IEnumerable<TSource>, Func<TSource,IEnumerable<TCollection>>, Func<TSource,TCollection,TResult>)	Projects each element of a sequence to an IEnumerable<T>, flattens the resulting sequences into one sequence, and invokes a result selector function on each element therein.
SelectMany<TSource,TCollection,TResult>(IEnumerable<TSource>, Func<TSource,Int32,IEnumerable<TCollection>>, Func<TSource,TCollection,TResult>)	Projects each element of a sequence to an IEnumerable<T>, flattens the resulting sequences into one sequence, and invokes a result selector function on each element therein. The index of each source element is used in the intermediate projected form of that element.
SequenceEqual<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)	Determines whether two sequences are equal by comparing the elements by using the default equality comparer for their type.
SequenceEqual<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)	Determines whether two sequences are equal by comparing their elements by using a specified IEqualityComparer<T>.
Single<TSource>(IEnumerable<TSource>)	Returns the only element of a sequence, and throws an exception if there is not exactly one element in the sequence.
Single<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Returns the only element of a sequence that satisfies a specified condition, and throws an exception if more than one such element exists.
SingleOrDefault<TSource>(IEnumerable<TSource>)	Returns the only element of a sequence, or a default value if the sequence is empty; this method throws an exception if there is more than one element in the sequence.
SingleOrDefault<TSource>(IEnumerable<TSource>, TSource)	Returns the only element of a sequence, or a specified default value if the sequence is empty; this method throws an exception if there is more than one element in the sequence.
SingleOrDefault<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Returns the only element of a sequence that satisfies a specified condition or a default value if no such element exists; this method throws an exception if more than one element satisfies the condition.
SingleOrDefault<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>, TSource)	Returns the only element of a sequence that satisfies a specified condition, or a specified default value if no such element exists; this method throws an exception if more than one element satisfies the condition.
Skip<TSource>(IEnumerable<TSource>, Int32)	Bypasses a specified number of elements in a sequence and then returns the remaining

	elements.
SkipLast<TSource>(IEnumerable<TSource>, Int32)	Returns a new enumerable collection that contains the elements from `source` with the last `count` elements of the source collection omitted.
SkipWhile<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Bypasses elements in a sequence as long as a specified condition is true and then returns the remaining elements.
SkipWhile<TSource>(IEnumerable<TSource>, Func<TSource,Int32,Boolean>)	Bypasses elements in a sequence as long as a specified condition is true and then returns the remaining elements. The element's index is used in the logic of the predicate function.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Decimal>)	Computes the sum of the sequence of Decimal values that are obtained by invoking a transform function on each element of the input sequence.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Double>)	Computes the sum of the sequence of Double values that are obtained by invoking a transform function on each element of the input sequence.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Int32>)	Computes the sum of the sequence of Int32 values that are obtained by invoking a transform function on each element of the input sequence.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Int64>)	Computes the sum of the sequence of Int64 values that are obtained by invoking a transform function on each element of the input sequence.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Decimal>>)	Computes the sum of the sequence of nullable Decimal values that are obtained by invoking a transform function on each element of the input sequence.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Double>>)	Computes the sum of the sequence of nullable Double values that are obtained by invoking a transform function on each element of the input sequence.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int32>>)	Computes the sum of the sequence of nullable Int32 values that are obtained by invoking a transform function on each element of the input sequence.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int64>>)	Computes the sum of the sequence of nullable Int64 values that are obtained by invoking a transform function on each element of the input sequence.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Single>>)	Computes the sum of the sequence of nullable Single values that are obtained by invoking a transform function on each element of the input sequence.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Single>)	Computes the sum of the sequence of Single values that are obtained by invoking a transform function on each element of the input sequence.
Take<TSource>(IEnumerable<TSource>, Int32)	Returns a specified number of contiguous elements from the start of a sequence.

Take<TSource>(IEnumerable<TSource>, Range)	Returns a specified range of contiguous elements from a sequence.
TakeLast<TSource>(IEnumerable<TSource>, Int32)	Returns a new enumerable collection that contains the last `count` elements from `source`.
TakeWhile<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Returns elements from a sequence as long as a specified condition is true.
TakeWhile<TSource>(IEnumerable<TSource>, Func<TSource,Int32,Boolean>)	Returns elements from a sequence as long as a specified condition is true. The element's index is used in the logic of the predicate function.
ToArray<TSource>(IEnumerable<TSource>)	Creates an array from a IEnumerable<T>.
ToDictionary<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)	Creates a Dictionary<TKey,TValue> from an IEnumerable<T> according to a specified key selector function.
ToDictionary<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IEqualityComparer<TKey>)	Creates a Dictionary<TKey,TValue> from an IEnumerable<T> according to a specified key selector function and key comparer.
ToDictionary<TSource,TKey,TElement>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>)	Creates a Dictionary<TKey,TValue> from an IEnumerable<T> according to specified key selector and element selector functions.
ToDictionary<TSource,TKey,TElement>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>, IEqualityComparer<TKey>)	Creates a Dictionary<TKey,TValue> from an IEnumerable<T> according to a specified key selector function, a comparer, and an element selector function.
ToHashSet<TSource>(IEnumerable<TSource>)	Creates a HashSet<T> from an IEnumerable<T>.
ToHashSet<TSource>(IEnumerable<TSource>, IEqualityComparer<TSource>)	Creates a HashSet<T> from an IEnumerable<T> using the `comparer` to compare keys.
ToList<TSource>(IEnumerable<TSource>)	Creates a List<T> from an IEnumerable<T>.
ToLookup<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)	Creates a Lookup<TKey,TElement> from an IEnumerable<T> according to a specified key selector function.
ToLookup<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IEqualityComparer<TKey>)	Creates a Lookup<TKey,TElement> from an IEnumerable<T> according to a specified key selector function and key comparer.
ToLookup<TSource,TKey,TElement>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>)	Creates a Lookup<TKey,TElement> from an IEnumerable<T> according to specified key selector and element selector functions.
ToLookup<TSource,TKey,TElement>(IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>, IEqualityComparer<TKey>)	Creates a Lookup<TKey,TElement> from an IEnumerable<T> according to a specified key selector function, a comparer and an element selector function.
TryGetNonEnumeratedCount<TSource>(IEnumerable<TSource>, Int32)	Attempts to determine the number of elements in a sequence without forcing an enumeration.
Union<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)	Produces the set union of two sequences by using the default equality comparer.
Union<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)	Produces the set union of two sequences by using a specified IEqualityComparer<T>.

UnionBy<TSource,TKey>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource,TKey>)	Produces the set union of two sequences according to a specified key selector function.
UnionBy<TSource,TKey>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource,TKey>, IEqualityComparer<TKey>)	Produces the set union of two sequences according to a specified key selector function.
Where<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Filters a sequence of values based on a predicate.
Where<TSource>(IEnumerable<TSource>, Func<TSource,Int32,Boolean>)	Filters a sequence of values based on a predicate. Each element's index is used in the logic of the predicate function.
Zip<TFirst,TSecond>(IEnumerable<TFirst>, IEnumerable<TSecond>)	Produces a sequence of tuples with elements from the two specified sequences.
Zip<TFirst,TSecond,TThird>(IEnumerable<TFirst>, IEnumerable<TSecond>, IEnumerable<TThird>)	Produces a sequence of tuples with elements from the three specified sequences.
Zip<TFirst,TSecond,TResult>(IEnumerable<TFirst>, IEnumerable<TSecond>, Func<TFirst,TSecond,TResult>)	Applies a specified function to the corresponding elements of two sequences, producing a sequence of the results.
AsParallel(IEnumerable)	Enables parallelization of a query.
AsParallel<TSource>(IEnumerable<TSource>)	Enables parallelization of a query.
AsQueryable(IEnumerable)	Converts an IEnumerable to an IQueryable.
AsQueryable<TElement>(IEnumerable<TElement>)	Converts a generic IEnumerable<T> to a generic IQueryable<T>.
Ancestors<T>(IEnumerable<T>)	Returns a collection of elements that contains the ancestors of every node in the source collection.
Ancestors<T>(IEnumerable<T>, XName)	Returns a filtered collection of elements that contains the ancestors of every node in the source collection. Only elements that have a matching XName are included in the collection.
DescendantNodes<T>(IEnumerable<T>)	Returns a collection of the descendant nodes of every document and element in the source collection.
Descendants<T>(IEnumerable<T>)	Returns a collection of elements that contains the descendant elements of every element and document in the source collection.
Descendants<T>(IEnumerable<T>, XName)	Returns a filtered collection of elements that contains the descendant elements of every element and document in the source collection. Only elements that have a matching XName are included in the collection.
Elements<T>(IEnumerable<T>)	Returns a collection of the child elements of every element and document in the source collection.
Elements<T>(IEnumerable<T>, XName)	Returns a filtered collection of the child elements of every element and document in the source collection. Only elements that have a matching XName are included in the collection.

InDocumentOrder<T>(IEnumerable<T>)	Returns a collection of nodes that contains all nodes in the source collection, sorted in document order.
Nodes<T>(IEnumerable<T>)	Returns a collection of the child nodes of every document and element in the source collection.
Remove<T>(IEnumerable<T>)	Removes every node in the source collection from its parent node.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.6, 2.0, 2.1
UWP	10.0

Thread Safety

Public static (`Shared` in Visual Basic) members of this type are thread safe. Any instance members are not guaranteed to be thread safe.

It is safe to perform multiple read operations on a List<T>, but issues can occur if the collection is modified while it's being read. To ensure thread safety, lock the collection during a read or write operation. To enable a collection to be accessed by multiple threads for reading and writing, you must implement your own synchronization. For collections with built-in synchronization, see the classes in the System.Collections.Concurrent namespace. For an inherently thread-safe alternative, see the ImmutableList<T> class.

See also

- IList
- ImmutableList<T>
- Performing Culture-Insensitive String Operations in Collections
- Iterators (C#)
- Iterators (Visual Basic)