System.Collections.Generic.LinkedList<T> Class

```
[ILAsm]
.class public serializable beforefieldinit LinkedList`1<T> extends
System.Object implements System.Collections.Generic.ICollection`1<!0>,
System.Collections.Generic.IEnumerable`1<!0>,
System.Collections.ICollection, System.Collections.IEnumerable

[C#]
public class LinkedList<T>: System.Collections.Generic.ICollection<T>,
System.Collections.Generic.IEnumerable<T>, System.Collections.ICollection
```

Assembly Info:

- *Name:* System
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 4.0.0.0
- *Attributes:*
    - CLSCompliantAttribute(true)

Implements:

- **System.Collections.Generic.ICollection<T>**
- **System.Collections.Generic.IEnumerable<T>**
- **System.Collections.ICollection**
- **System.Runtime.Serialization.IDeserializationCallback**
- **System.Runtime.Serialization.ISerializable**

Summary

Represents a doubly linked list.

Inherits From: System.Object

Library: BCL

Description

``System.Collections.Generic.LinkedList`1<T>`` is a general-purpose linked list. It supports enumerators and implements the `System.Collections.ICollection` interface, consistent with other collection classes in the standard.

``System.Collections.Generic.LinkedList`1<T>`` provides separate nodes of type ``System.Collections.Generic.LinkedListNode`1<T>``, so insertion and removal are O(1) operations.

You can remove nodes and reinsert them, either in the same list or in another list, which results in no additional objects allocated on the heap. Because the list also maintains an internal count, getting the System.Collections.Generic.LinkedList`1<T>.Count property is an O(1) operation.

Each node in a System.Collections.Generic.LinkedList`1<T> object is of the type System.Collections.Generic.LinkedListNode`1<T>. Because the System.Collections.Generic.LinkedList`1<T> is doubly linked, each node points forward to the System.Collections.Generic.LinkedListNode`1<T>.Next node and backward to the System.Collections.Generic.LinkedListNode`1<T>.Previous node.

Lists that contain reference types perform better when a node and its value are created at the same time. System.Collections.Generic.LinkedList`1<T> accepts null as a valid System.Collections.Generic.LinkedListNode`1<T>.Value property for reference types and allows duplicate values.

If the System.Collections.Generic.LinkedList`1<T> is empty, the System.Collections.Generic.LinkedList`1<T>.First and System.Collections.Generic.LinkedList`1<T>.Last properties contain null.

The System.Collections.Generic.LinkedList`1<T> class does not support chaining, splitting, cycles, or other features that can leave the list in an inconsistent state. The list remains consistent on a single thread. The only multithreaded scenario supported by System.Collections.Generic.LinkedList`1<T> is multithreaded read operations.

LinkedList<T>() Constructor

[ILAsm]
```
.method public hidebysig specialname rtspecialname instance void .ctor()
cil managed
```

[C#]
```
public LinkedList ()
```

Summary

Initializes a new instance of the System.Collections.Generic.LinkedList`1<T> class that is empty.

Description

System.Collections.Generic.LinkedList`1<T> accepts null as a valid System.Collections.Generic.LinkedListNode`1<T>.Value for reference types and allows duplicate values.

If the System.Collections.Generic.LinkedList`1<T> is empty, the System.Collections.Generic.LinkedList`1<T>.First and System.Collections.Generic.LinkedList`1<T>.Last properties contain null.

This constructor is an O(1) operation.

LinkedList<T>(System.Collections.Generic.IEnumerable<T>) Constructor

```
[ILAsm]
.method public hidebysig specialname rtspecialname instance void
.ctor(class System.Collections.Generic.IEnumerable`1<!0> collection) cil
managed

[C#]
public LinkedList (System.Collections.Generic.IEnumerable<T> collection)
```

Summary

Initializes a new instance of the `System.Collections.Generic.LinkedList`1<T>` class that contains elements copied from the specified `System.Collections.IEnumerable` and has sufficient capacity to accommodate the number of elements copied.

Parameters

Parameter	Description
collection	The `System.Collections.IEnumerable` whose elements are copied to the new `System.Collections.Generic.LinkedList`1<T>`.

Description

`System.Collections.Generic.LinkedList`1<T>` accepts `null` as a valid `System.Collections.Generic.LinkedListNode`1<T>.Value` for reference types and allows duplicate values.

If *collection* has no elements then the new `System.Collections.Generic.LinkedList`1<T>` is empty, and the `System.Collections.Generic.LinkedList`1<T>.First` and `System.Collections.Generic.LinkedList`1<T>.Last` properties contain `null`.

This constructor is an O(n) operation, where *n* is the number of elements in *collection*.

Exceptions

Exception	Condition
System.ArgumentNullException	*collection* is `null`.

LinkedList<T>(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext) Constructor

```
[ILAsm]
.method family hidebysig specialname rtspecialname instance void
.ctor(class System.Runtime.Serialization.SerializationInfo info, valuetype
System.Runtime.Serialization.StreamingContext context) cil managed

[C#]
protected LinkedList (System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context)
```

Summary

Initializes a new instance of the System.Collections.Generic.LinkedList`1<T> class that is serializable with the specified System.Runtime.Serialization.SerializationInfo and System.Runtime.Serialization.StreamingContext.

Parameters

Parameter	Description
info	A System.Runtime.Serialization.SerializationInfo object containing the information required to serialize the System.Collections.Generic.LinkedList`1<T>.
context	A System.Runtime.Serialization.StreamingContext object containing the source and destination of the serialized stream associated with the System.Collections.Generic.LinkedList`1<T>.

Description

System.Collections.Generic.LinkedList`1<T> accepts null as a valid System.Collections.Generic.LinkedListNode`1<T>.Value for reference types and allows duplicate values.

If the System.Collections.Generic.LinkedList`1<T> is empty, the System.Collections.Generic.LinkedList`1<T>.First and System.Collections.Generic.LinkedList`1<T>.Last properties contain null.

This constructor is an O(n) operation.

LinkedList<T>.AddAfter(System.Collections.Generic.LinkedListNode<T>, System.Collections.Generic.LinkedListNode<T>) Method

[ILAsm]
```
.method public hidebysig instance class
System.Collections.Generic.LinkedListNode`1<!0> AddAfter(class
System.Collections.Generic.LinkedListNode`1<!0> node, !0 'value') cil
managed
```

[C#]
```
public void AddAfter (System.Collections.Generic.LinkedListNode<T> node,
System.Collections.Generic.LinkedListNode<T> newNode)
```

Summary

Adds the specified new node after the specified existing node in the `System.Collections.Generic.LinkedList`1<T>`.

Parameters

Parameter	Description
node	The `System.Collections.Generic.LinkedListNode`1<T>` after which to insert *newNode*.
newNode	The new `System.Collections.Generic.LinkedListNode`1<T>` to add to the `System.Collections.Generic.LinkedList`1<T>`.

Description

`System.Collections.Generic.LinkedList`1<T>` accepts `null` as a valid `System.Collections.Generic.LinkedListNode`1<T>.Value` for reference types and allows duplicate values.

This method is an O(1) operation.

Exceptions

Exception	Condition
System.ArgumentNullException	*node* is `null`.

	-or- *newNode* is `null`.
System.InvalidOperationException	*node* is not in the current ``System.Collections.Generic.LinkedList`1<T>``. -or- *newNode* belongs to another ``System.Collections.Generic.LinkedList`1<T>``.

LinkedList<T>.AddAfter(System.Collections.Generic.LinkedListNode<T>, T) Method

```
[ILAsm]
.method public hidebysig instance void AddAfter(class
System.Collections.Generic.LinkedListNode`1<!0> node, class
System.Collections.Generic.LinkedListNode`1<!0> newNode) cil managed

[C#]
public System.Collections.Generic.LinkedListNode<T> AddAfter
(System.Collections.Generic.LinkedListNode<T> node, T value)
```

Summary

Adds a new node containing the specified value after the specified existing node in the `System.Collections.Generic.LinkedList`1<T>`.

Parameters

Parameter	Description
node	The `System.Collections.Generic.LinkedListNode`1<T>` after which to insert a new `System.Collections.Generic.LinkedListNode`1<T>` containing *value*.
value	The value to add to the `System.Collections.Generic.LinkedList`1<T>`.

Return Value

The new `System.Collections.Generic.LinkedListNode`1<T>` containing *value*.

Description

`System.Collections.Generic.LinkedList`1<T>` accepts `null` as a valid `System.Collections.Generic.LinkedListNode`1<T>.Value` for reference types and allows duplicate values.

This method is an O(1) operation.

Exceptions

Exception	Condition
System.ArgumentNullException	*node* is `null`.

System.InvalidOperationException	*node* is not in the current ``System.Collections.Generic.LinkedList`1<T>``.

LinkedList<T>.AddBefore(System.Collections.Generic.LinkedListNode<T>, System.Collections.Generic.LinkedListNode<T>) Method

```
[ILAsm]
.method public hidebysig instance void AddBefore(class
System.Collections.Generic.LinkedListNode`1<!0> node, class
System.Collections.Generic.LinkedListNode`1<!0> newNode) cil managed

[C#]
public void AddBefore (System.Collections.Generic.LinkedListNode<T> node,
System.Collections.Generic.LinkedListNode<T> newNode)
```

Summary

Adds the specified new node before the specified existing node in the ``System.Collections.Generic.LinkedList`1<T>``.

Parameters

Parameter	Description
node	The ``System.Collections.Generic.LinkedListNode`1<T>`` before which to insert *newNode*.
newNode	The new ``System.Collections.Generic.LinkedListNode`1<T>`` to add to the ``System.Collections.Generic.LinkedList`1<T>``.

Description

``System.Collections.Generic.LinkedList`1<T>`` accepts `null` as a valid ``System.Collections.Generic.LinkedListNode`1<T>.Value`` for reference types and allows duplicate values.

This method is an O(1) operation.

Exceptions

Exception	Condition
System.ArgumentNullException	*node* is `null`.

	-or- *newNode* is `null`.
System.InvalidOperationException	*node* is not in the current ``System.Collections.Generic.LinkedList`1<T>``. -or- *newNode* belongs to another ``System.Collections.Generic.LinkedList`1<T>``.

LinkedList<T>.AddBefore(System.Collections.Generic.LinkedListNode<T>, T) Method

```
[ILAsm]
.method public hidebysig instance class
System.Collections.Generic.LinkedListNode`1<!0> AddBefore(class
System.Collections.Generic.LinkedListNode`1<!0> node, !0 'value') cil
managed

[C#]
public System.Collections.Generic.LinkedListNode<T> AddBefore
(System.Collections.Generic.LinkedListNode<T> node, T value)
```

Summary

Adds a new node containing the specified value before the specified existing node in the `System.Collections.Generic.LinkedList`1<T>`.

Parameters

Parameter	Description
node	The `System.Collections.Generic.LinkedListNode`1<T>` before which to insert a new `System.Collections.Generic.LinkedListNode`1<T>` containing *value*.
value	The value to add to the `System.Collections.Generic.LinkedList`1<T>`.

Return Value

The new `System.Collections.Generic.LinkedListNode`1<T>` containing *value*.

Description

`System.Collections.Generic.LinkedList`1<T>` accepts `null` as a valid `System.Collections.Generic.LinkedListNode`1<T>.Value` for reference types and allows duplicate values.

This method is an O(1) operation.

Exceptions

Exception	Condition

System.ArgumentNullException	*node* is `null`.
System.InvalidOperationException	*node* is not in the current ``System.Collections.Generic.LinkedList`1<T>``.

LinkedList<T>.AddFirst(System.Collections.Generic.LinkedListNode<T>) Method

[ILAsm]
```
.method public hidebysig instance void AddFirst(class
System.Collections.Generic.LinkedListNode`1<!0> node) cil managed
```

[C#]
```
public void AddFirst (System.Collections.Generic.LinkedListNode<T> node)
```

Summary

Adds the specified new node at the start of the `System.Collections.Generic.LinkedList`1<T>`.

Parameters

Parameter	Description
node	The new `System.Collections.Generic.LinkedListNode`1<T>` to add at the start of the `System.Collections.Generic.LinkedList`1<T>`.

Description

`System.Collections.Generic.LinkedList`1<T>` accepts `null` as a valid `System.Collections.Generic.LinkedListNode`1<T>.Value` for reference types and allows duplicate values.

If the `System.Collections.Generic.LinkedList`1<T>` is empty, the new node becomes the `System.Collections.Generic.LinkedList`1<T>.First` and the `System.Collections.Generic.LinkedList`1<T>.Last`.

This method is an O(1) operation.

Exceptions

Exception	Condition
System.ArgumentNullException	*node* is `null`.
System.InvalidOperationException	*node* belongs to another `System.Collections.Generic.LinkedList`1<T>`.

LinkedList<T>.AddFirst(T) Method

```
[ILAsm]
.method public hidebysig instance class
System.Collections.Generic.LinkedListNode`1<!0> AddFirst(!0 'value') cil
managed

[C#]
public System.Collections.Generic.LinkedListNode<T> AddFirst (T value)
```

Summary

Adds a new node containing the specified value at the start of the ``System.Collections.Generic.LinkedList`1<T>``.

Parameters

Parameter	Description
value	The value to add at the start of the ``System.Collections.Generic.LinkedList`1<T>``.

Return Value

The new ``System.Collections.Generic.LinkedListNode`1<T>`` containing *value*.

Description

``System.Collections.Generic.LinkedList`1<T>`` accepts `null` as a valid ``System.Collections.Generic.LinkedListNode`1<T>.Value`` for reference types and allows duplicate values.

If the ``System.Collections.Generic.LinkedList`1<T>`` is empty, the new node becomes the ``System.Collections.Generic.LinkedList`1<T>.First`` and the ``System.Collections.Generic.LinkedList`1<T>.Last``.

This method is an O(1) operation.

LinkedList<T>.AddLast(System.Collections.Generic.LinkedListNode<T>) Method

```
[ILAsm]
.method public hidebysig instance class
System.Collections.Generic.LinkedListNode`1<!0> AddLast(!0 'value') cil
managed

[C#]
public void AddLast (System.Collections.Generic.LinkedListNode<T> node)
```

Summary

Adds the specified new node at the end of the System.Collections.Generic.LinkedList`1<T>.

Parameters

Parameter	Description
node	The new System.Collections.Generic.LinkedListNode`1<T> to add at the end of the System.Collections.Generic.LinkedList`1<T>.

Description

System.Collections.Generic.LinkedList`1<T> accepts null as a valid System.Collections.Generic.LinkedListNode`1<T>.Value for reference types and allows duplicate values.

If the System.Collections.Generic.LinkedList`1<T> is empty, the new node becomes the System.Collections.Generic.LinkedList`1<T>.First and the System.Collections.Generic.LinkedList`1<T>.Last.

This method is an O(1) operation.

Exceptions

Exception	Condition
System.ArgumentNullException	*node* is null.
System.InvalidOperationException	*node* belongs to another System.Collections.Generic.LinkedList`1<T>.

LinkedList<T>.AddLast(T) Method

```
[ILAsm]
.method public hidebysig instance void AddLast(class
System.Collections.Generic.LinkedListNode`1<!0> node) cil managed

[C#]
public System.Collections.Generic.LinkedListNode<T> AddLast (T value)
```

Summary

Adds a new node containing the specified value at the end of the System.Collections.Generic.LinkedList`1<T>.

Parameters

Parameter	Description
value	The value to add at the end of the System.Collections.Generic.LinkedList`1<T>.

Return Value

The new System.Collections.Generic.LinkedListNode`1<T> containing *value*.

Description

System.Collections.Generic.LinkedList`1<T> accepts null as a valid System.Collections.Generic.LinkedListNode`1<T>.Value for reference types and allows duplicate values.

If the System.Collections.Generic.LinkedList`1<T> is empty, the new node becomes the System.Collections.Generic.LinkedList`1<T>.First and the System.Collections.Generic.LinkedList`1<T>.Last.

This method is an O(1) operation.

LinkedList<T>.Clear() Method

[ILAsm]
```
.method public hidebysig newslot virtual final instance void Clear() cil managed
```

[C#]
```
public void Clear ()
```

Summary

Removes all nodes from the System.Collections.Generic.LinkedList`1<T>.

Description

System.Collections.Generic.LinkedList`1<T>.Count is set to zero, and references to other objects from elements of the collection are also released. System.Collections.Generic.LinkedList`1<T>.First and System.Collections.Generic.LinkedList`1<T>.Last are set to null.

This method is an O(*n*) operation, where *n* is System.Collections.Generic.LinkedList`1<T>.Count.

LinkedList<T>.Contains(T) Method

```
[ILAsm]
.method public hidebysig newslot virtual final instance bool Contains(!0
'value') cil managed

[C#]
public bool Contains (T value)
```

Summary

Determines whether a value is in the ``System.Collections.Generic.LinkedList`1<T>``.

Parameters

Parameter	Description
value	The value to locate in the ``System.Collections.Generic.LinkedList`1<T>``. The value can be `null` for reference types.

Return Value

`true` if *value* is found in the ``System.Collections.Generic.LinkedList`1<T>``; otherwise, `false`.

Description

This method performs a linear search; therefore, this method is an O(*n*) operation, where *n* is ``System.Collections.Generic.LinkedList`1<T>.Count``.

LinkedList<T>.CopyTo(T[], System.Int32) Method

```
[ILAsm]
.method public hidebysig newslot virtual final instance void CopyTo(!0[]
array, int32 index) cil managed

[C#]
public void CopyTo (T[] array, int index)
```

Summary

Copies the entire ``System.Collections.Generic.LinkedList`1<T>`` to a compatible one-dimensional `System.Array`, starting at the specified index of the target array.

Parameters

Parameter	Description
array	The one-dimensional `System.Array` that is the destination of the elements copied from ``System.Collections.Generic.LinkedList`1<T>``. The `System.Array` must have zero-based indexing.
index	The zero-based index in *array* at which copying begins.

Description

This method uses `System.Array.Copy` to copy the elements.

The elements are copied to the `System.Array` in the same order in which the enumerator iterates through the ``System.Collections.Generic.LinkedList`1<T>``.

This method is an O(*n*) operation, where *n* is ``System.Collections.Generic.LinkedList`1<T>.Count``.

Exceptions

Exception	Condition
System.ArgumentNullException	*array* is `null`.
System.ArgumentOutOfRangeException	*index* is less than zero.
System.ArgumentException	The number of elements in the source ``System.Collections.Generic.LinkedList`1<T>``

	is greater than the available space from *index* to the end of the destination *array*.

LinkedList<T>.Find(T) Method

```
[ILAsm]
.method public hidebysig instance class
System.Collections.Generic.LinkedListNode`1<!0> Find(!0 'value') cil
managed

[C#]
public System.Collections.Generic.LinkedListNode<T> Find (T value)
```

Summary

Finds the first node that contains the specified value.

Parameters

Parameter	Description
value	The value to locate in the ``System.Collections.Generic.LinkedList`1<T>``.

Return Value

The first ``System.Collections.Generic.LinkedListNode`1<T>`` that contains the specified value, if found; otherwise, `null`.

Description

The ``System.Collections.Generic.LinkedList`1<T>`` is searched forward starting at ``System.Collections.Generic.LinkedList`1<T>.First`` and ending at ``System.Collections.Generic.LinkedList`1<T>.Last``.

This method performs a linear search; therefore, this method is an O(*n*) operation, where *n* is ``System.Collections.Generic.LinkedList`1<T>.Count``.

LinkedList<T>.FindLast(T) Method

```
[ILAsm]
.method public hidebysig instance class
System.Collections.Generic.LinkedListNode`1<!0> FindLast(!0 'value') cil
managed

[C#]
public System.Collections.Generic.LinkedListNode<T> FindLast (T value)
```

Summary

Finds the last node that contains the specified value.

Parameters

Parameter	Description
value	The value to locate in the `System.Collections.Generic.LinkedList`1<T>`.

Return Value

The last `System.Collections.Generic.LinkedListNode`1<T>` that contains the specified value, if found; otherwise, `null`.

Description

The `System.Collections.Generic.LinkedList`1<T>` is searched backward starting at `System.Collections.Generic.LinkedList`1<T>.Last` and ending at `System.Collections.Generic.LinkedList`1<T>.First`.

This method performs a linear search; therefore, this method is an O(*n*) operation, where *n* is `System.Collections.Generic.LinkedList`1<T>.Count`.

LinkedList<T>.GetEnumerator() Method

[ILAsm]
```
.method public hidebysig instance valuetype
System.Collections.Generic.LinkedList`1/Enumerator<!0> GetEnumerator() cil
managed
```

[C#]
```
public System.Collections.Generic.LinkedList<T>.Enumerator GetEnumerator
()
```

Summary

Returns an enumerator that iterates through the System.Collections.Generic.LinkedList`1<T>.

Return Value

An System.Collections.Generic.LinkedList`1<T>.Enumerator for the System.Collections.Generic.LinkedList`1<T>.

Usage

For a detailed description regarding the use of an enumerator, see `System.Collections.Generic.IEnumerator<T>`.

LinkedList<T>.Remove(System.Collections.Generic.LinkedListNode<T>) Method

```
[ILAsm]
.method public hidebysig newslot virtual final instance bool Remove(!0
'value') cil managed

[C#]
public void Remove (System.Collections.Generic.LinkedListNode<T> node)
```

Summary

Removes the specified node from the System.Collections.Generic.LinkedList`1<T>.

Parameters

Parameter	Description
node	The System.Collections.Generic.LinkedListNode`1<T> to remove from the System.Collections.Generic.LinkedList`1<T>.

Description

This method is an O(1) operation.

Exceptions

Exception	Condition
System.ArgumentNullException	*node* is `null`.
System.InvalidOperationException	*node* is not in the current System.Collections.Generic.LinkedList`1<T>.

LinkedList<T>.Remove(T) Method

[ILAsm]
```
.method public hidebysig instance void Remove(class
System.Collections.Generic.LinkedListNode`1<!0> node) cil managed
```

[C#]
```
public bool Remove (T value)
```

Summary

Removes the first occurrence of the specified value from the `System.Collections.Generic.LinkedList`1<T>`.

Parameters

Parameter	Description
value	The value to remove from the `System.Collections.Generic.LinkedList`1<T>`.

Return Value

`true` if the element containing *value* is successfully removed; otherwise, `false`. This method also returns `false` if *value* was not found in the original `System.Collections.Generic.LinkedList`1<T>`.

Description

This method performs a linear search; therefore, this method is an O(*n*) operation, where *n* is `System.Collections.Generic.LinkedList`1<T>.Count`.

LinkedList<T>.RemoveFirst() Method

[ILAsm]
```
.method public hidebysig instance void RemoveFirst() cil managed
```

[C#]
```
public void RemoveFirst ()
```

Summary

Removes the node at the start of the ``System.Collections.Generic.LinkedList`1<T>``.

Description

This method is an O(1) operation.

Exceptions

Exception	Condition
System.InvalidOperationException	The ``System.Collections.Generic.LinkedList`1<T>`` is empty.

LinkedList<T>.RemoveLast() Method

```
[ILAsm]
.method public hidebysig instance void RemoveLast() cil managed

[C#]
public void RemoveLast ()
```

Summary

Removes the node at the end of the System.Collections.Generic.LinkedList`1<T>.

Description

This method is an O(1) operation.

Exceptions

Exception	Condition
System.InvalidOperationException	The System.Collections.Generic.LinkedList`1<T> is empty.

LinkedList<T>.System.Collections.Generic.ICollection<T>.Add(T) Method

[ILAsm]
```
.method private hidebysig newslot virtual final instance void
System.Collections.Generic.ICollection<T>.Add(!0 'value') cil managed
```

[C#]
```
void ICollection<T>.Add (T value)
```

Summary

Adds an item at the end of the System.Collections.Generic.ICollection`1<T>.

Parameters

Parameter	Description
value	The value to add at the end of the System.Collections.Generic.ICollection`1<T>.

Description

System.Collections.Generic.LinkedList`1<T> accepts null as a valid System.Collections.Generic.LinkedListNode`1<T>.Value for reference types and allows duplicate values.

If the System.Collections.Generic.LinkedList`1<T> is empty, the new node becomes the System.Collections.Generic.LinkedList`1<T>.First and the System.Collections.Generic.LinkedList`1<T>.Last.

This method is an O(1) operation.

LinkedList<T>.System.Collections.Generic.IEnumerable<T>.GetEnumerator() Method

[ILAsm]

```
.method private hidebysig newslot virtual final instance class
System.Collections.Generic.IEnumerator`1<!0>
System.Collections.Generic.IEnumerable<T>.GetEnumerator() cil managed
```

[C#]

```
System.Collections.Generic.IEnumerator<T> IEnumerable<T>.GetEnumerator ()
```

Summary

Returns an enumerator that iterates through a collection.

Return Value

An ``System.Collections.Generic.IEnumerator`1<T>`` that can be used to iterate through the collection.

Usage

For a detailed description regarding the use of an enumerator, see `System.Collections.Generic.IEnumerator<T>`.

LinkedList<T>.System.Collections.ICollection.CopyTo(System.Array, System.Int32) Method

```
[ILAsm]
.method private hidebysig newslot virtual final instance void
System.Collections.ICollection.CopyTo(class System.Array array, int32
index) cil managed

[C#]
void ICollection.CopyTo (Array array, int index)
```

Summary

Copies the elements of the `System.Collections.ICollection` to an `System.Array`, starting at a particular `System.Array` index.

Parameters

Parameter	Description
array	The one-dimensional `System.Array` that is the destination of the elements copied from `System.Collections.ICollection`. The `System.Array` must have zero-based indexing.
index	The zero-based index in *array* at which copying begins.

Description

[*Note:* If the type of the source `System.Collections.ICollection` cannot be cast automatically to the type of the destination *array*, the non-generic implementations of `System.Collections.ICollection.CopyTo` throw `System.InvalidCastException`, whereas the generic implementations throw `System.ArgumentException`.

]

This method is an O(*n*) operation, where *n* is ``System.Collections.Generic.LinkedList`1<T>.Count``.

Exceptions

Exception	Condition
System.ArgumentNullException	*array* is `null`.

System.ArgumentOutOfRangeException	*index* is less than zero.
System.ArgumentException	*array* is multidimensional. -or- *array* does not have zero-based indexing. -or- The number of elements in the source `System.Collections.ICollection` is greater than the available space from *index* to the end of the destination *array*. -or- The type of the source `System.Collections.ICollection` cannot be cast automatically to the type of the destination *array*.

LinkedList<T>.System.Collections.IEnumerable.GetEnumerator() Method

```
[ILAsm]
.method private hidebysig newslot virtual final instance class
System.Collections.IEnumerator
System.Collections.IEnumerable.GetEnumerator() cil managed

[C#]
System.Collections.IEnumerator IEnumerable.GetEnumerator ()
```

Summary

Returns an enumerator that iterates through the linked list as a collection.

Return Value

An `System.Collections.IEnumerator` that can be used to iterate through the linked list as a collection.

Usage

For a detailed description regarding the use of an enumerator, see `System.Collections.Generic.IEnumerator<T>`.

LinkedList<T>.Count Property

[ILAsm]
```
.property instance int32 Count
```

[C#]
```
public int Count { get; }
```

Summary

Gets the number of nodes actually contained in the ``System.Collections.Generic.LinkedList`1<T>``.

Property Value

The number of nodes actually contained in the ``System.Collections.Generic.LinkedList`1<T>``.

Description

Retrieving the value of this property is an O(1) operation.

LinkedList<T>.First Property

```
[ILAsm]
.property instance class System.Collections.Generic.LinkedListNode`1<!0>
First

[C#]
public System.Collections.Generic.LinkedListNode<T> First { get; }
```

Summary

Gets the first node of the `System.Collections.Generic.LinkedList`1<T>`.

Property Value

The first `System.Collections.Generic.LinkedListNode`1<T>` of the `System.Collections.Generic.LinkedList`1<T>`.

Description

`System.Collections.Generic.LinkedList`1<T>` accepts `null` as a valid `System.Collections.Generic.LinkedListNode`1<T>.Value` for reference types and allows duplicate values.

If the `System.Collections.Generic.LinkedList`1<T>` is empty, the `System.Collections.Generic.LinkedList`1<T>.First` and `System.Collections.Generic.LinkedList`1<T>.Last` properties contain `null`.

Retrieving the value of this property is an O(1) operation.

LinkedList<T>.Last Property

```
[ILAsm]
.property instance class System.Collections.Generic.LinkedListNode`1<!0> 
Last

[C#]
public System.Collections.Generic.LinkedListNode<T> Last { get; }
```

Summary

Gets the last node of the `System.Collections.Generic.LinkedList`1<T>`.

Property Value

The last `System.Collections.Generic.LinkedListNode`1<T>` of the `System.Collections.Generic.LinkedList`1<T>`.

Description

`System.Collections.Generic.LinkedList`1<T>` accepts `null` as a valid `System.Collections.Generic.LinkedListNode`1<T>.Value` for reference types and allows duplicate values.

If the `System.Collections.Generic.LinkedList`1<T>` is empty, the `System.Collections.Generic.LinkedList`1<T>.First` and `System.Collections.Generic.LinkedList`1<T>.Last` properties contain `null`.

Retrieving the value of this property is an O(1) operation.

LinkedList<T>.System.Collections.Generic.ICollection<T>.IsReadOnly Property

```
[ILAsm]
.property instance bool
System.Collections.Generic.ICollection<T>.IsReadOnly

[C#]
bool System.Collections.Generic.ICollection<T>.IsReadOnly { get; }
```

Summary

Gets a value indicating whether the System.Collections.Generic.ICollection`1<T> is read-only.

Property Value

true if the System.Collections.Generic.ICollection`1<T> is read-only; otherwise, false. In the default implementation of System.Collections.Generic.LinkedList`1<T>, this property always returns `false`.

Description

A collection that is read-only does not allow the addition, removal, or modification of elements after the collection is created.

A collection that is read-only is simply a collection with a wrapper that prevents modifying the collection; therefore, if changes are made to the underlying collection, the read-only collection reflects those changes.

Retrieving the value of this property is an O(1) operation.

LinkedList<T>.System.Collections.ICollection.IsSynchronized Property

```
[ILAsm]
.property instance bool System.Collections.ICollection.IsSynchronized

[C#]
bool System.Collections.ICollection.IsSynchronized { get; }
```

Summary

Gets a value indicating whether access to the `System.Collections.ICollection` is synchronized (thread safe).

Property Value

`true` if access to the `System.Collections.ICollection` is synchronized (thread safe); otherwise, `false`. In the default implementation of ``System.Collections.Generic.LinkedList`1<T>``, this property always returns `false`.

Description

Default implementations of collections in `System.Collections.Generic` are not synchronized.

Enumerating through a collection is intrinsically not a thread-safe procedure. To guarantee thread safety during enumeration, you can lock the collection during the entire enumeration. To allow the collection to be accessed by multiple threads for reading and writing, you must implement your own synchronization.

`System.Collections.ICollection.SyncRoot` returns an object that can be used to synchronize access to the `System.Collections.ICollection`. Synchronization is effective only if all threads lock this object before accessing the collection.

Retrieving the value of this property is an O(1) operation.

LinkedList<T>.System.Collections.ICollection.SyncRoot Property

[ILAsm]
```
.property instance object System.Collections.ICollection.SyncRoot
```

[C#]
```
object System.Collections.ICollection.SyncRoot { get; }
```

Summary

Gets an object that can be used to synchronize access to the `System.Collections.ICollection`.

Property Value

An object that can be used to synchronize access to the `System.Collections.ICollection`. In the default implementation of ``System.Collections.Generic.LinkedList`1<T>``, this property always returns the current instance.

Description

Default implementations of collections in `System.Collections.Generic` are not synchronized.

Enumerating through a collection is intrinsically not a thread-safe procedure. To guarantee thread safety during enumeration, you can lock the collection during the entire enumeration. To allow the collection to be accessed by multiple threads for reading and writing, you must implement your own synchronization.

`System.Collections.ICollection.SyncRoot` returns an object that can be used to synchronize access to the `System.Collections.ICollection`. Synchronization is effective only if all threads lock this object before accessing the collection. The following code shows the use of the `System.Collections.ICollection.SyncRoot` property for C#.

```
ICollection ic =...;
lock (ic.SyncRoot) {
   // Access the collection.
}
```

Retrieving the value of this property is an O(1) operation.