

1 System.Collections.Generic.LinkedList<T>

2 Class

```
3 [ILAsm]
4 .class public serializable beforefieldinit LinkedList`1<T> extends
5 System.Object implements System.Collections.Generic ICollection`1<!0>,
6 System.Collections.Generic.IEnumerable`1<!0>,
7 System.Collections.ICollection, System.Collections.IEnumerable
8
9 [C#]
10 public class LinkedList<T>: System.Collections.Generic.ICollection<T>,
    System.Collections.Generic.IEnumerable<T>, System.Collections.ICollection
```

11 Assembly Info:

- 12 • Name: System
- 13 • Public Key: [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- 14 • Version: 4.0.0.0
- 15 • Attributes:
 - 16 ○ CLSCompliantAttribute(true)

17 Implements:

- 18 • System.Collections.Generic.ICollection<T>
- 19 • System.Collections.Generic.IEnumerable<T>
- 20 • System.Collections.ICollection
- 21 • System.Runtime.Serialization.IDeserializationCallback
- 22 • System.Runtime.Serialization.ISerializable

23 Summary

24 Represents a doubly linked list.

25 Inherits From: System.Object

26 Library: BCL

29 Description

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

33
34 System.Collections.Generic.LinkedList<T> provides separate nodes of type
35 System.Collections.Generic.LinkedListNode<T>, so insertion and removal are
36 O(1) operations.

37
38 You can remove nodes and reinsert them, either in the same list or in another list, which
39 results in no additional objects allocated on the heap. Because the list also maintains an

1 internal count, getting the `System.Collections.Generic.LinkedList<T>.Count`
2 property is an $O(1)$ operation.
3
4 Each node in a `System.Collections.Generic.LinkedList<T>` object is of the type
5 `System.Collections.Generic.LinkedListNode<T>`. Because the
6 `System.Collections.Generic.LinkedList<T>` is doubly linked, each node points
7 forward to the `System.Collections.Generic.LinkedListNode<T>.Next` node and
8 backward to the `System.Collections.Generic.LinkedListNode<T>.Previous` node.
9
10 Lists that contain reference types perform better when a node and its value are created
11 at the same time. `System.Collections.Generic.LinkedList<T>` accepts null as a
12 valid `System.Collections.Generic.LinkedListNode<T>.Value` property for
13 reference types and allows duplicate values.
14
15 If the `System.Collections.Generic.LinkedList<T>` is empty, the
16 `System.Collections.Generic.LinkedList<T>.First` and
17 `System.Collections.Generic.LinkedList<T>.Last` properties contain null.
18
19 The `System.Collections.Generic.LinkedList<T>` class does not support chaining,
20 splitting, cycles, or other features that can leave the list in an inconsistent state. The list
21 remains consistent on a single thread. The only multithreaded scenario supported by
22 `System.Collections.Generic.LinkedList<T>` is multithreaded read operations.
23

1 `LinkedList<T>()` Constructor

```
2 [ILAsm]  
3 .method public hidebysig specialname rtspecialname instance void .ctor()  
4 cil managed  
  
5 [C#]  
6 public LinkedList ()
```

7 **Summary**

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

10 **Description**

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

14
15 If the `System.Collections.Generic.LinkedList<T>` is empty, the
16 `System.Collections.Generic.LinkedList<T>.First` and
17 `System.Collections.Generic.LinkedList<T>.Last` properties contain null.

18
19 This constructor is an O(1) operation.

20

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<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
<i>collection</i>	The <code>System.Collections.IEnumerable</code> whose elements are copied to the new <code>System.Collections.Generic.LinkedList<T></code> .

Description

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

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

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

Exceptions

Exception	Condition
<code>System.ArgumentNullException</code>	<i>collection</i> is null.

28
29

1
2 **LinkedList<T> (System.Runtime.Serialization.**
3 **SerializationInfo,**
4 **System.Runtime.Serialization.StreamingCont**
5 **ext) Constructor**

```
6 [ILAsm]  
7 .method family hidebysig specialname rtspecialname instance void  
8 .ctor(class System.Runtime.Serialization.SerializationInfo info, valuetype  
9 System.Runtime.Serialization.StreamingContext context) cil managed  
  
10 [C#]  
11 protected LinkedList (System.Runtime.Serialization.SerializationInfo info,  
12 System.Runtime.Serialization.StreamingContext context)
```

13 **Summary**

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

18 **Parameters**

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

19
20 **Description**

21 System.Collections.Generic.LinkedList<T> accepts null as a valid
22 System.Collections.Generic.LinkedListNode<T>.Value for reference types and
23 allows duplicate values.
24
25 If the System.Collections.Generic.LinkedList<T> is empty, the
26 System.Collections.Generic.LinkedList<T>.First and
27 System.Collections.Generic.LinkedList<T>.Last properties contain null.
28
29 This constructor is an O(n) operation.

1
2 **LinkedList<T>.AddAfter(System.Collections.Generic.LinkedListNode<T>,**
3 **System.Collections.Generic.LinkedListNode<T**
4 **>) Method**
5

```
6 [ILAsm]  
7 .method public hidebysig instance class  
8 System.Collections.Generic.LinkedListNode`1<!0> AddAfter(class  
9 System.Collections.Generic.LinkedListNode`1<!0> node, !0 'value') cil  
10 managed  
  
11 [C#]  
12 public void AddAfter (System.Collections.Generic.LinkedListNode<T> node,  
13 System.Collections.Generic.LinkedListNode<T> newNode)
```

14 **Summary**

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

17 **Parameters**

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

18
19 **Description**

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

23
24 This method is an O(1) operation.

25 **Exceptions**

Exception	Condition
System.ArgumentNullException	<i>node</i> is null.

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

1

2

1 2 `LinkedList<T>.AddAfter(System.Collections.Generic.LinkedListNode<T>, T) Method` 3

```
4 [ILAsm]  
5 .method public hidebysig instance void AddAfter(class  
6 System.Collections.Generic.LinkedListNode`1<!0> node, class  
7 System.Collections.Generic.LinkedListNode`1<!0> newNode) cil managed  
  
8 [C#]  
9 public System.Collections.Generic.LinkedListNode<T> AddAfter  
10 (System.Collections.Generic.LinkedListNode<T> node, T value)
```

11 Summary

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

14 Parameters

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

15 16 Return Value

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

18 Description

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

22
23 This method is an O(1) operation.

24 Exceptions

Exception	Condition
<code>System.ArgumentNullException</code>	<i>node</i> is null.

System.InvalidOperationException

node is not in the current

`System.Collections.Generic.LinkedList`1<T>`.

1

2

1
2 **LinkedList<T>.AddBefore(System.Collections.Generic.LinkedListNode<T>,**
3 **System.Collections.Generic.LinkedListNode<T**
4 **>) Method**
5

```
6 [ILAsm]  
7 .method public hidebysig instance void AddBefore(class  
8 System.Collections.Generic.LinkedListNode`1<!0> node, class  
9 System.Collections.Generic.LinkedListNode`1<!0> newNode) cil managed  
  
10 [C#]  
11 public void AddBefore (System.Collections.Generic.LinkedListNode<T> node,  
12 System.Collections.Generic.LinkedListNode<T> newNode)
```

13 **Summary**

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

16 **Parameters**

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

17
18 **Description**

19 System.Collections.Generic.LinkedList`1<T> accepts null as a valid
20 System.Collections.Generic.LinkedListNode`1<T>.Value for reference types and
21 allows duplicate values.
22

23 This method is an O(1) operation.

24 **Exceptions**

Exception	Condition
System.ArgumentNullException	<i>node</i> is null.

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

1

2

1 2 **LinkedList<T>.AddBefore(System.Collections.Generic.LinkedListNode<T>, T) Method** 3

```
4 [ILAsm]  
5 .method public hidebysig instance class  
6 System.Collections.Generic.LinkedListNode`1<!0> AddBefore(class  
7 System.Collections.Generic.LinkedListNode`1<!0> node, !0 'value') cil  
8 managed  
  
9 [C#]  
10 public System.Collections.Generic.LinkedListNode<T> AddBefore  
11 (System.Collections.Generic.LinkedListNode<T> node, T value)
```

12 **Summary**

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

15 **Parameters**

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

16 17 **Return Value**

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

19 **Description**

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

23
24 This method is an O(1) operation.

25 **Exceptions**

Exception	Condition
-----------	-----------

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

1

2

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

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

```
[ILAsm]  
.method public hidebysig instance void AddFirst(class  
System.Collections.Generic.LinkedListNode`1<T> 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<T>.

Parameters

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

Description

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

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

This method is an O(1) operation.

Exceptions

Exception	Condition
System.ArgumentNullException	<i>node</i> is null.
System.InvalidOperationException	<i>node</i> belongs to another System.Collections.Generic.LinkedList<T>.

1 LinkedList<T>.AddFirst(T) Method

```
2 [ILAsm]
3 .method public hidebysig instance class
4 System.Collections.Generic.LinkedListNode`1<T> AddFirst(!0 'value') cil
5 managed
6 [C#]
7 public System.Collections.Generic.LinkedListNode<T> AddFirst (T value)
```

8 Summary

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

11 Parameters

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

12

13 Return Value

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

15 Description

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

19

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

23

24 This method is an O(1) operation.

25

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

```
[ILAsm]
.method public hidebysig instance class
System.Collections.Generic.LinkedListNode`1<T> 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<T>.

Parameters

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

Description

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

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

This method is an O(1) operation.

Exceptions

Exception	Condition
System.ArgumentNullException	<i>node</i> is null.
System.InvalidOperationException	<i>node</i> belongs to another System.Collections.Generic.LinkedList<T>.

1 **LinkedList<T>.AddLast(T) Method**

```
2 [ILAsm]  
3 .method public hidebysig instance void AddLast(class  
4 System.Collections.Generic.LinkedListNode`1<T> node) cil managed  
  
5 [C#]  
6 public System.Collections.Generic.LinkedListNode<T> AddLast (T value)
```

7 **Summary**

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

10 **Parameters**

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

11 12 **Return Value**

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

14 **Description**

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

18
19 If the System.Collections.Generic.LinkedList`1<T> is empty, the new node
20 becomes the System.Collections.Generic.LinkedList`1<T>.First and the
21 System.Collections.Generic.LinkedList`1<T>.Last.

22
23 This method is an O(1) operation.

24

1 **LinkedList<T>.Clear() Method**

```
2 [ILAsm]  
3 .method public hidebysig newslot virtual final instance void Clear() cil  
4 managed  
5 [C#]  
6 public void Clear ()
```

7 **Summary**

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

9 **Description**

10 `System.Collections.Generic.LinkedList<T>.Count` is set to zero, and references
11 to other objects from elements of the collection are also released.
12 `System.Collections.Generic.LinkedList<T>.First` and
13 `System.Collections.Generic.LinkedList<T>.Last` are set to null.
14
15 This method is an $O(n)$ operation, where n is
16 `System.Collections.Generic.LinkedList<T>.Count`.

17

1 `LinkedList<T>.Contains(T)` Method

```
2 [ILAsm]  
3 .method public hidebysig newslot virtual final instance bool Contains(!0  
4 'value') cil managed  
  
5 [C#]  
6 public bool Contains (T value)
```

7 **Summary**

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

9 **Parameters**

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

10

11 **Return Value**

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

14 **Description**

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

17

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

2 Method

```
3 [ILAsm]  
4 .method public hidebysig newslot virtual final instance void CopyTo(!0[]  
5 array, int32 index) cil managed  
  
6 [C#]  
7 public void CopyTo (T[] array, int index)
```

8 Summary

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

11 Parameters

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

12

13 Description

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

15
16 The elements are copied to the `System.Array` in the same order in which the
17 enumerator iterates through the `System.Collections.Generic.LinkedList<T>`.

18
19 This method is an $O(n)$ operation, where n is
20 `System.Collections.Generic.LinkedList<T>.Count`.

21 Exceptions

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

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

1

2

1 LinkedList<T>.Find(T) Method

```
2 [ILAsm]
3 .method public hidebysig instance class
4 System.Collections.Generic.LinkedListNode`1<T> Find(!0 'value') cil
5 managed
6 [C#]
7 public System.Collections.Generic.LinkedListNode<T> Find (T value)
```

8 Summary

9 Finds the first node that contains the specified value.

10 Parameters

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

11

12 Return Value

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

15 Description

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

19

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

22

1 `LinkedList<T>.FindLast(T)` Method

```
2 [ILAsm]  
3 .method public hidebysig instance class  
4 System.Collections.Generic.LinkedListNode`1<T> FindLast(!0 'value') cil  
5 managed  
  
6 [C#]  
7 public System.Collections.Generic.LinkedListNode<T> FindLast (T value)
```

8 **Summary**

9 Finds the last node that contains the specified value.

10 **Parameters**

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

11 **Return Value**

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

15 **Description**

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

19
20 This method performs a linear search; therefore, this method is an $O(n)$ operation,
21 where n is `System.Collections.Generic.LinkedList`1<T>.Count`.

22

1 `LinkedList<T>.GetEnumerator()` Method

```
2 [ILAsm]  
3 .method public hidebysig instance valuetype  
4 System.Collections.Generic.LinkedList`1/Enumerator<!0> GetEnumerator() cil  
5 managed  
  
6 [C#]  
7 public System.Collections.Generic.LinkedList<T>.Enumerator GetEnumerator  
8 ()
```

9 **Summary**

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

12 **Return Value**

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

15 **Usage**

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

18

19

1 `LinkedList<T>.Remove(T)` Method

```
2 [ILAsm]  
3 .method public hidebysig instance void Remove(class  
4 System.Collections.Generic.LinkedListNode`1<!0> node) cil managed  
  
5 [C#]  
6 public bool Remove (T value)
```

7 **Summary**

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

10 **Parameters**

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

11

12 **Return Value**

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

16 **Description**

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

19

1 `LinkedList<T>.RemoveFirst()` Method

```
2 [ILAsm]  
3 .method public hidebysig instance void RemoveFirst() cil managed  
  
4 [C#]  
5 public void RemoveFirst ()
```

6 **Summary**

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

8 **Description**

9 This method is an O(1) operation.

10 **Exceptions**

Exception	Condition
System.InvalidOperationException	The <code>System.Collections.Generic.LinkedList<T></code> is empty.

11

12

1 `LinkedList<T>.RemoveLast()` Method

```
2 [ILAsm]  
3 .method public hidebysig instance void RemoveLast() cil managed  
4 [C#]  
5 public void RemoveLast ()
```

6 **Summary**

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

8 **Description**

9 This method is an O(1) operation.

10 **Exceptions**

Exception	Condition
System.InvalidOperationException	The <code>System.Collections.Generic.LinkedList<T></code> is empty.

11

12

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

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
<i>value</i>	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.

1
2 **LinkedList<T>.System.Collections.Generic.IE**
3 **numerable<T>.GetEnumerator() Method**

```
4 [ILAsm]  
5 .method private hidebysig newslot virtual final instance class  
6 System.Collections.Generic.IEnumerator`1<!0>  
7 System.Collections.Generic.IEnumerable<T>.GetEnumerator() cil managed  
  
8 [C#]  
9 System.Collections.Generic.IEnumerator<T> IEnumerable<T>.GetEnumerator ()
```

10 **Summary**

11 Returns an enumerator that iterates through a collection.

12 **Return Value**

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

15 **Usage**

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

18

19

1
2 **LinkedList<T>.System.Collections.ICollection.**
3 **CopyTo(System.Array, System.Int32) Method**

```
4 [ILAsm]  
5 .method private hidebysig newslot virtual final instance void  
6 System.Collections.ICollection.CopyTo(class System.Array array, int32  
7 index) cil managed  
  
8 [C#]  
9 void ICollection.CopyTo (Array array, int index)
```

10 **Summary**

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

13 **Parameters**

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

14
15 **Description**

16 [Note: If the type of the source `System.Collections.ICollection` cannot be cast
17 automatically to the type of the destination *array*, the non-generic implementations of
18 `System.Collections.ICollection.CopyTo` throw `System.InvalidCastException`,
19 whereas the generic implementations throw `System.ArgumentException`.
20]

21]
22
23 This method is an $O(n)$ operation, where n is
24 `System.Collections.Generic.LinkedList<T>.Count`.

25 **Exceptions**

Exception	Condition
System.ArgumentNullException	<i>array</i> is null.

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

1

2

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

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>`.

1 **LinkedList<T>.Count Property**

2 [ILAsm]
3 **.property instance int32 Count**

4 [C#]
5 **public int Count { get; }**

6 **Summary**

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

9 **Property Value**

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

12 **Description**

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

14

1 **LinkedList<T>.First Property**

```
2 [ILAsm]  
3 .property instance class System.Collections.Generic.LinkedListNode`1<!0>  
4 First  
5 [C#]  
6 public System.Collections.Generic.LinkedListNode<T> First { get; }
```

7 **Summary**

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

9 **Property Value**

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

12 **Description**

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

16
17 If the System.Collections.Generic.LinkedList`1<T> is empty, the
18 System.Collections.Generic.LinkedList`1<T>.First and
19 System.Collections.Generic.LinkedList`1<T>.Last properties contain null.

20
21 Retrieving the value of this property is an O(1) operation.

22

1 **LinkedList<T>.Last Property**

```
2 [ILAsm]  
3 .property instance class System.Collections.Generic.LinkedListNode`1<!0>  
4 Last  
5 [C#]  
6 public System.Collections.Generic.LinkedListNode<T> Last { get; }
```

7 **Summary**

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

9 **Property Value**

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

12 **Description**

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

16
17 If the `System.Collections.Generic.LinkedList`1<T>` is empty, the
18 `System.Collections.Generic.LinkedList`1<T>.First` and
19 `System.Collections.Generic.LinkedList`1<T>.Last` properties contain null.

20
21 Retrieving the value of this property is an O(1) operation.

22

1 2 **LinkedList<T>.System.Collections.Generic.ICollection<T>.IsReadOnly Property** 3

```
4 [ILAsm]  
5 .property instance bool  
6 System.Collections.Generic.ICollection<T>.IsReadOnly  
7 [C#]  
8 bool System.Collections.Generic.ICollection<T>.IsReadOnly { get; }
```

9 **Summary**

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

12 **Property Value**

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

16 **Description**

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

19
20 A collection that is read-only is simply a collection with a wrapper that prevents
21 modifying the collection; therefore, if changes are made to the underlying collection, the
22 read-only collection reflects those changes.

23
24 Retrieving the value of this property is an O(1) operation.

25

1 2 **LinkedList<T>.System.Collections.ICollection.** 3 **IsSynchronized Property**

```
4 [ILAsm]  
5 .property instance bool System.Collections.ICollection.IsSynchronized  
6 [C#]  
7 bool System.Collections.ICollection.IsSynchronized { get; }
```

8 **Summary**

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

11 **Property Value**

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

15 **Description**

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

18
19 Enumerating through a collection is intrinsically not a thread-safe procedure. To
20 guarantee thread safety during enumeration, you can lock the collection during the
21 entire enumeration. To allow the collection to be accessed by multiple threads for
22 reading and writing, you must implement your own synchronization.

23
24 `System.Collections.ICollection.SyncRoot` returns an object that can be used to
25 synchronize access to the `System.Collections.ICollection`. Synchronization is
26 effective only if all threads lock this object before accessing the collection.

27
28 Retrieving the value of this property is an O(1) operation.

29

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<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.