**Класс LinkedList<T>**

В классе LinkedList<T> создается коллекция в виде обобщенного двунаправленного

списка. В этом классе реализуются интерфейсы ICollection, ICollection<T>,

IEnumerable, IEnumerable<T>, ISerializable и IDeserializationCallback.

В двух последних интерфейсах поддерживается сериализация списка. В классе

LinkedList<T> определяются два приведенных ниже открытых конструктора.

public LinkedList()

public LinkedList(IEnumerable<T> collection)

В первом конструкторе создается пустой связный список, а во втором конструкторе

— список, инициализируемый элементами из коллекции *collection.*

Как и в большинстве других реализаций связных списков, в классе LinkedList<T>

инкапсулируются значения, хранящиеся в *узлах* списка, где находятся также ссылки на

предыдущие и последующие элементы списка. Эти узлы представляют собой объекты

класса LinkedListNode<T>. В классе LinkedListNode<T> предоставляются четыре

следующих свойства.

public LinkedListNode<T> Next { get; }

public LinkedListNode<T> Previous { get; }

public LinkedList<T> List { get; }

public T Value { get; set; }

С помощью свойств Next и Previous получаются ссылки на предыдущий и последующий

узлы списка соответственно, что дает возможность обходить список в обоих

направлениях. Если же предыдущий или последующий узел отсутствует, то возвращается

пустая ссылка. Для получения ссылки на сам список служит свойство List.

А с помощью свойства Value можно устанавливать и получать значение, находящееся

в узле списка.

В классе LinkedList<T> определяется немало методов. В табл. 25.16 приведены

наиболее часто используемые методы данного класса. Кроме того, в классе

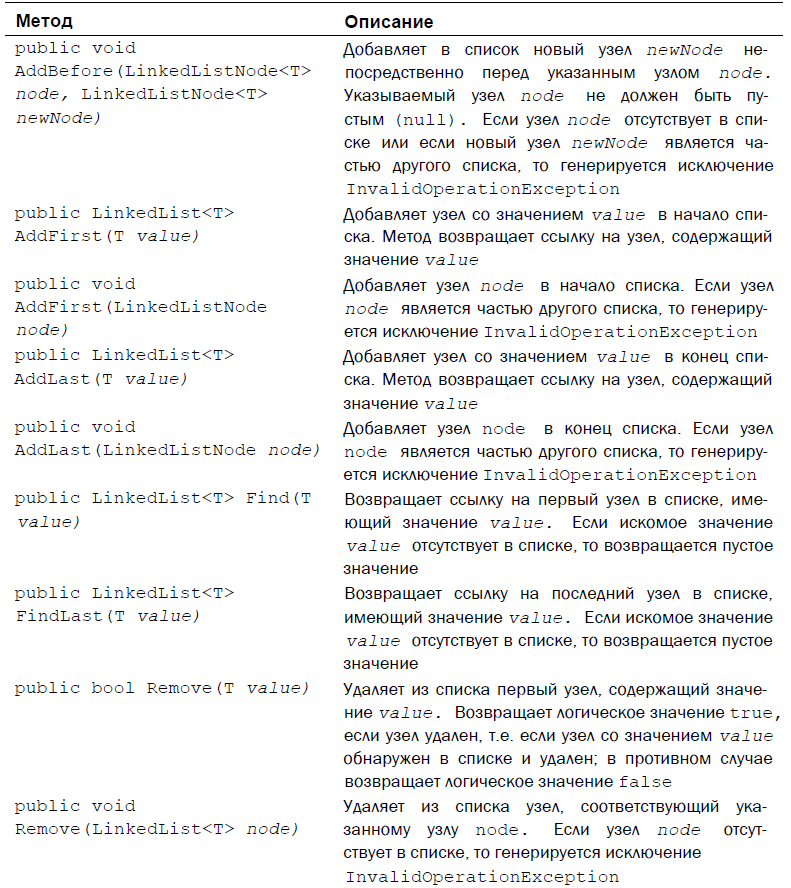
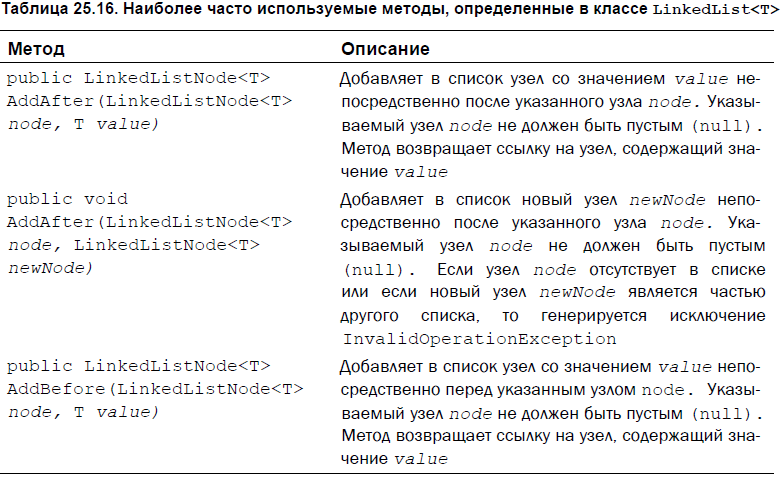
LinkedList<T> определяются собственные свойства, помимо тех, что уже

объявлены в интерфейсах, которые в нем реализуются. Эти свойства приведены

ниже.

public LinkedListNode<T> First { get; }

public LinkedListNode<T> Last { get; }



(***glava25\_10***)

class GenLinkedListDemo

{

static void Main()

{

//make linked list

LinkedList<char> ll = new LinkedList<char>();

Console.WriteLine("Count of ll: " + ll.Count);

Console.WriteLine();

Console.WriteLine("Add in list 5 elements");

ll.AddFirst('A');

ll.AddFirst('B');

ll.AddFirst('C');

ll.AddFirst('D');

ll.AddFirst('E');

Console.WriteLine("Count of ll: " + ll.Count);

//show manually

LinkedListNode<char> node;

Console.Write("Show ll via links: ");

for (node = ll.First; node != null; node = node.Next)

Console.Write(node.Value + " ");

Console.WriteLine("\n");

Console.Write("Show content via foreach: ");

foreach (var ch in ll)

Console.Write(ch + " ");

Console.WriteLine("\n");

Console.Write("Show ll via links backwards: ");

for (node = ll.Last; node != null; node = node.Previous)

Console.Write(node.Value + " ");

Console.WriteLine("\n");

//remove elements from linkedlist

ll.Remove('C');

ll.Remove('A');

Console.WriteLine("Count of ll: " + ll.Count);

foreach (var ch in ll)

Console.Write(ch + " ");

Console.WriteLine("\n");

//3 elements end of list

ll.AddLast('X');

ll.AddLast('Y');

ll.AddLast('Z');

foreach (var ch in ll)

Console.Write(ch + " ");

Console.WriteLine("\n");

}

}

Самое примечательное в этой программе — это обход списка в прямом и обратном

направлении, следуя по ссылкам, предоставляемым свойствами Next и Previous.

Двунаправленный характер подобных связных списков имеет особое значение для

приложений, управляющих базами данных, где нередко требуется перемещаться по

списку в обоих направлениях.