**Класс List<T>**

В классе List<T> реализуется обобщенный динамический массив. Он ничем

принципиально не отличается от класса необобщенной коллекции ArrayList.

В этом классе реализуются интерфейсы ICollection, ICollection<T>, IList,

IList<T>, IEnumerable и IEnumerable<T>. У класса List<T> имеются следующие

конструкторы.

public List()

public List(IEnumerable<T> collection)

public List(int capacity)

Первый конструктор создает пустую коллекцию класса List с выбираемой по

умолчанию первоначальной емкостью. Второй конструктор создает коллекцию типа

List с количеством инициализируемых элементов, которое определяется параметром

*collection* и равно первоначальной емкости массива. Третий конструктор создает

коллекцию типа List, имеющую первоначальную емкость, задаваемую параметром

*capacity.* В данном случае емкость обозначает размер базового массива, используемого

для хранения элементов коллекции. Емкость коллекции, создаваемой в виде динамического

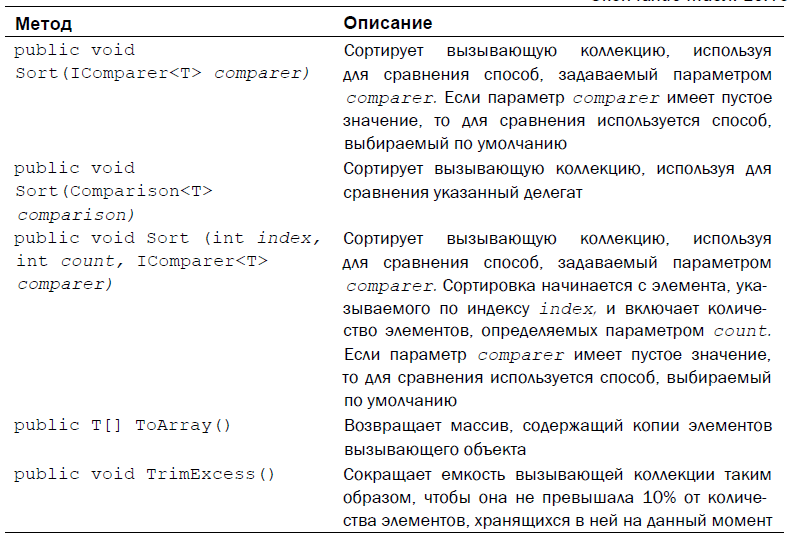
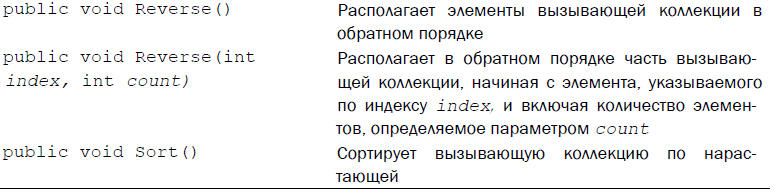
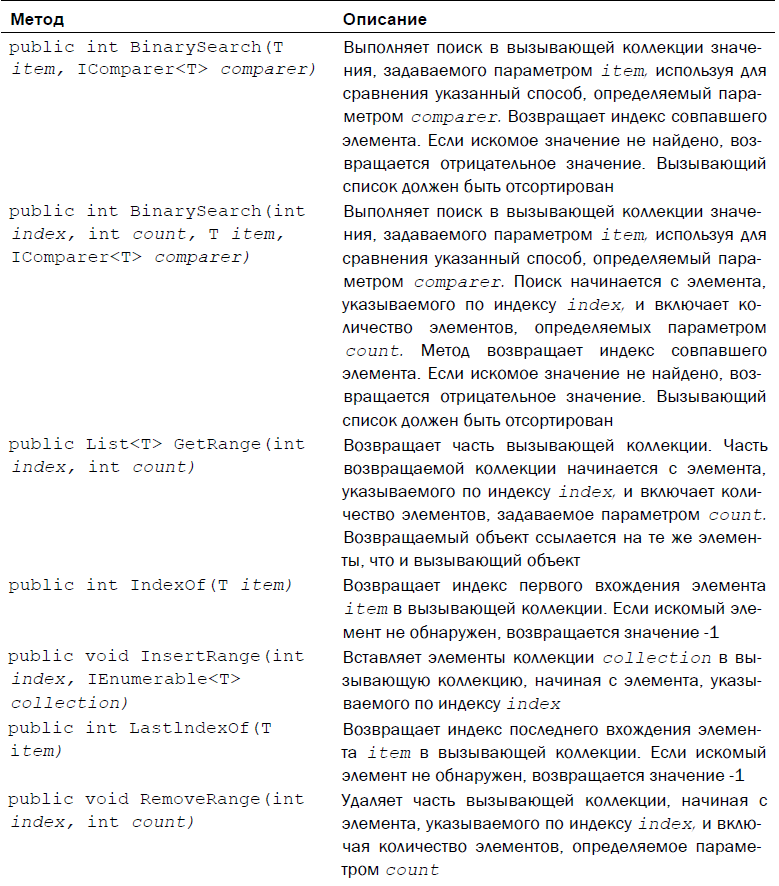
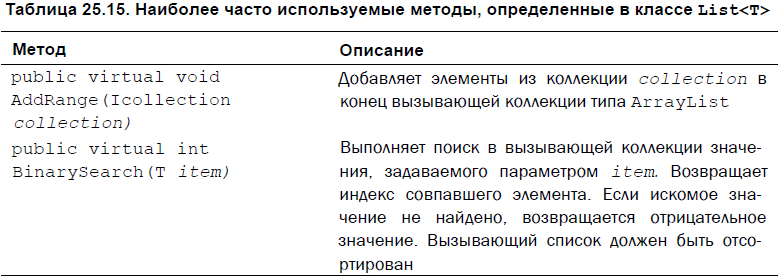
массива, может увеличиваться автоматически по мере добавления в нее

элементов.

В классе List<T> определяется ряд собственных методов, помимо тех, что уже объявлены

в интерфейсах, которые в нем реализуются. Некоторые из наиболее часто используемых

методов этого класса перечислены в табл. 25.15.



В классе List<T> определяется также собственное свойство Capacity, помимо тех,

что уже объявлены в интерфейсах, которые в нем реализуются. Это свойство объявляется

следующим образом.

public int Capacity { get; set; }

Свойство Capacity позволяет установить и получить емкость вызывающей коллекции

в качестве динамического массива. Эта емкость равна количеству элементов, которые

может содержать коллекция до ее вынужденного расширения.

В классе List<T> реализуется также приведенный ниже индексатор, определенный

в интерфейсе IList<T>.

public Т this[int index] { get; set; }

С помощью этого индексатора устанавливается и получается значение элемента

коллекции, указываемое по индексу *index.*

В приведенном ниже примере программы демонстрируется применение класса

List<T>. Это измененный вариант примера, демонстрировавшего ранее класс

ArrayList. Единственное изменение, которое потребовалось для этого, заключалось

в замене класса ArrayList классом List, а также в использовании параметров обобщенного

типа.

(***glava25\_9***)

class GenListDemo

{

static void Main()

{

//create collection as dyn array

List<char> lst = new List<char>();

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

Console.WriteLine();

Console.WriteLine("Add 6 elements.");

lst.Add('C');

lst.Add('A');

lst.Add('E');

lst.Add('B');

lst.Add('D');

lst.Add('F');

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

//show content via indexes

Console.Write("Content of lst: ");

for (int i = 0; i < lst.Count; i++)

Console.Write(lst[i] + " ");

Console.WriteLine("\n");

Console.WriteLine("Remove 2 elements: ");

lst.Remove('F');

lst.Remove('A');

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

//show content via foreach

Console.Write("Content of lst: ");

foreach (var c in lst)

Console.Write(c + " ");

Console.WriteLine("\n");

Console.WriteLine("Add more 20 elements.");

for (int i = 0; i < 20; i++)

lst.Add((char)('a' + i));

Console.WriteLine("Capacity of lst: " + lst.Capacity);

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

Console.Write("Content of lst: ");

foreach (var c in lst)

Console.Write(c + " ");

Console.WriteLine("\n");

//change via index

Console.WriteLine("Change first 3 elements.");

lst[0] = 'X';

lst[1] = 'Y';

lst[2] = 'Z';

Console.Write("Content of lst: ");

foreach (var c in lst)

Console.Write(c + " ");

Console.WriteLine("\n");

}

}