**Класс ArrayList**

В классе ArrayList поддерживаются динамические массивы, расширяющиеся и

сокращающиеся по мере необходимости. В классе

ArrayList определяется массив переменной длины, который состоит из ссылок на

объекты и может динамически увеличивать и уменьшать свой размер. Массив типа

ArrayList создается с первоначальным размером. Если этот размер превышается, то

массив автоматически расширяется. А при удалении объектов из такого массива он

автоматически сокращается. Коллекции класса ArrayList широко применяются в

практике программирования на С#. Именно поэтому они рассматриваются здесь подробно.

В классе ArrayList реализуются интерфейсы ICollection, IList, IEnumerable

и ICloneable. Ниже приведены конструкторы класса ArrayList.

public ArrayList()

public ArrayList(ICollection c)

public ArrayList(int capacity)

Первый конструктор создает пустую коллекцию класса ArrayList с нулевой первоначальной

емкостью. Второй конструктор создает коллекцию типа ArrayList с количеством

инициализируемых элементов, которое определяется параметром с и равно

первоначальной емкости массива. Третий конструктор создает коллекцию, имеющую

указанную первоначальную емкость, определяемую параметром *сараcity.* В данном

случае емкость обозначает размер базового массива, используемого для хранения элементов

коллекции.

В классе ArrayList определяется ряд собственных методов, помимо тех, что уже

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

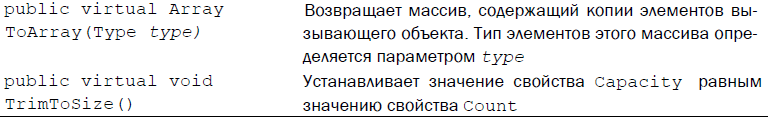
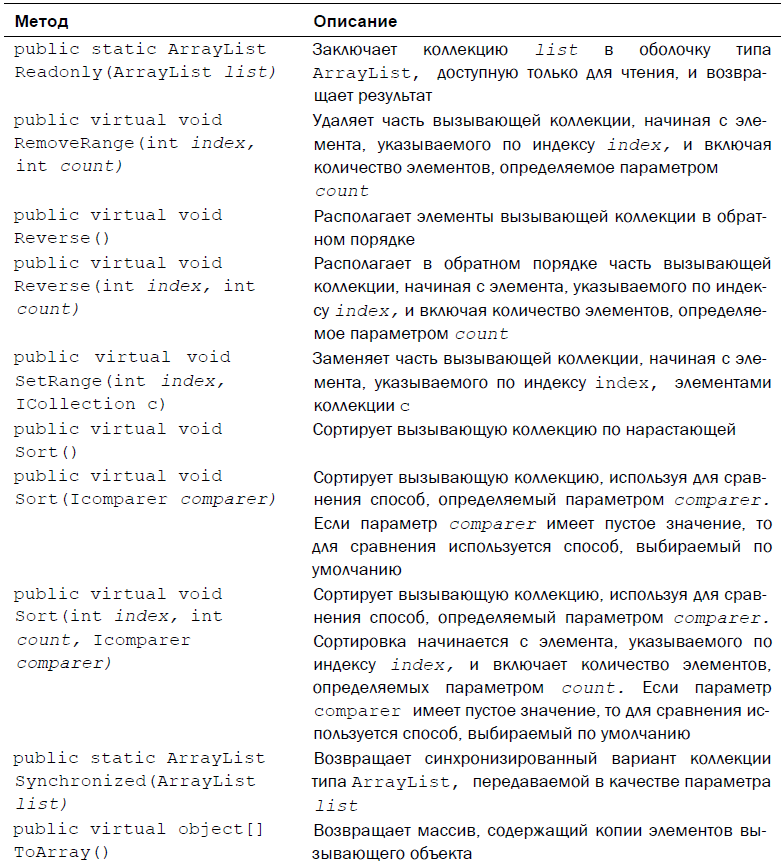
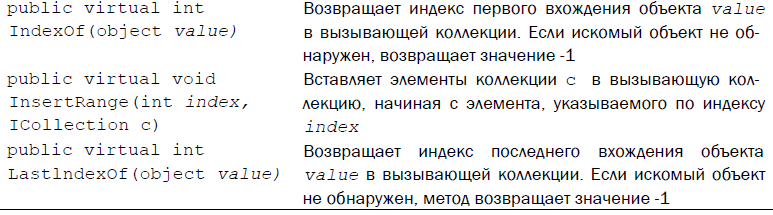
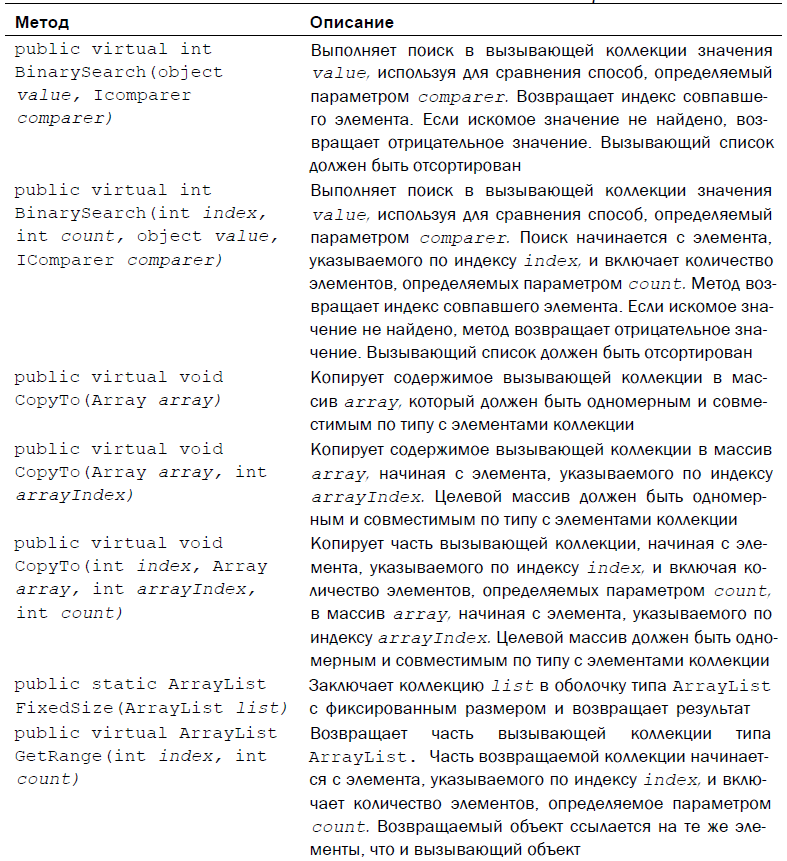
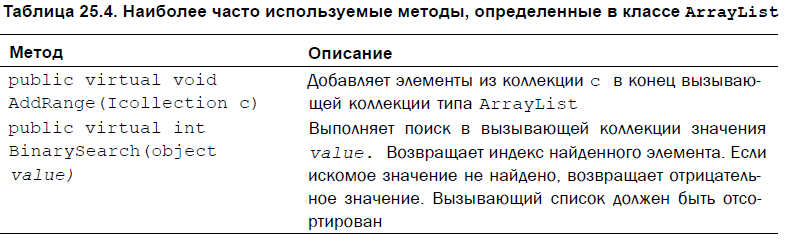
используемых методов класса ArrayList перечислены в табл. 25.4. Коллекцию класса

ArrayList можно отсортировать, вызвав метод Sort(). В этом случае поиск в отсортированной

коллекции с помощью метода BinarySearch() становится еще более

эффективным. Содержимое коллекции типа ArrayList можно также обратить, вызвав

метод Reverse().



По умолчанию коллекция типа ArrayList не синхронизирована. Для получения

синхронизированной оболочки, в которую заключается коллекция, вызывается метод

Synchronized().

В классе ArrayList имеется также приведенное ниже свойство Capacity, помимо

свойств, определенных в интерфейсах, которые в нем реализуются.

public virtual int Capacity { get; set; }

Свойство Capacity позволяет получать и устанавливать емкость вызывающей коллекции

типа ArrayList. Емкость обозначает количество элементов, которые может

содержать коллекция типа ArrayList до ее вынужденного расширения.

С другой стороны, если требуется сократить размер базового массива коллекции

типа ArrayList, то для этой цели достаточно установить меньшее значение свойства

Capacity. Но это значение не должно быть меньше значения свойства Count. Напомним,

что свойство Count определено в интерфейсе ICollection и содержит количество

объектов, хранящихся в коллекции на данный момент. Всякая попытка установить

значение свойства Capacity меньше значения свойства Count приводит к генерированию

исключения ArgumentOutOfRangeException.

В приведенном ниже примере программы демонстрируется применение класса

ArrayList. В ней сначала создается коллекция типа ArrayList, а затем в эту коллекцию

вводятся символы, после чего содержимое коллекции отображается. Некоторые

элементы затем удаляются из коллекции, и ее содержимое отображается вновь. После

этого в коллекцию вводятся дополнительные элементы, что вынуждает увеличить ее

емкость. И наконец, содержимое элементов коллекции изменяется.

(***glava25\_1***)

class ArrayListDemo

{

static void Main()

{

//create collection as dynamic array

ArrayList al = new ArrayList();

Console.WriteLine("Start qnty elements: " + al.Count);

Console.WriteLine();

Console.WriteLine("Add 6 elements");

//add elements in dyn arr

al.Add('C');

al.Add('A');

al.Add('E');

al.Add('B');

al.Add('D');

al.Add('F');

Console.WriteLine("Count elements: " + al.Count);

//show containing of dyn arr

//via index

Console.Write("Current content: ");

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

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

Console.WriteLine("\n");

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

al.Remove('F');

al.Remove('A');

Console.WriteLine("Count: " + al.Count);

//show content with foreach

Console.Write("Content: ");

foreach (char c in al)

Console.Write(c + " ");

Console.WriteLine("\n");

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

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

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

Console.WriteLine("Current size: " + al.Capacity);

Console.WriteLine("Count after add 20 elements: " + al.Count);

Console.Write("Content: ");

foreach (char c in al)

Console.Write(c + " ");

Console.WriteLine("\n");

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

al[0] = 'X';

al[1] = 'Y';

al[2] = 'Z';

Console.Write("Content: ");

foreach (char c in al)

Console.Write(c + " ");

Console.WriteLine("\n");

}

}

**Сортировка и поиск в коллекции типа ArrayList**

Коллекцию типа ArrayList можно отсортировать с помощью метода Sort().

В этом случае поиск в отсортированной коллекции с помощью метода BinarySearch()

становится еще более эффективным. Применение обоих методов демонстрируется

в приведенном ниже примере программы.

(***glava25\_2***)

class ArrayListDemo

{

static void Main()

{

//create collection as dynamic array

ArrayList al = new ArrayList();

Console.WriteLine("Add 6 elements");

//add elements in dyn arr

al.Add(55);

al.Add(43);

al.Add(-4);

al.Add(88);

al.Add(3);

al.Add(19);

Console.Write("Current content: ");

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

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

Console.WriteLine("\n");

//sort arr

al.Sort();

Console.Write("Current content: ");

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

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

Console.WriteLine("\n");

}

}

В одной и той же коллекции типа ArrayList могут храниться объекты любого

типа. Тем не менее во время сортировки и поиска в ней эти объекты приходится сравнивать.

Так, если бы список объектов в приведенном выше примере программы содержал

символьную строку, то их сравнение привело бы к исключительной ситуации.

Впрочем, для сравнения символьных строк и целых чисел можно создать специальные

методы. О таких методах сравнения речь пойдет далее в этой главе.

**Получение массива из коллекции типа ArrayList**

В работе с коллекцией типа ArrayList иногда требуется получить из ее содержимого

обычный массив. Этой цели служит метод ТоАrrау(). Для преобразования

коллекции в массив имеется несколько причин. Две из них таковы: потребность в ускорении

обработки при выполнении некоторых операций и необходимость передавать

массив методу, который не перегружается, чтобы принять коллекцию. Но независимо

от конкретной причины коллекция типа ArrayList преобразуется в обычный массив

довольно просто, как показано в приведенном ниже примере программы.

(***glava25\_3***)

class ArrayListDemo

{

static void Main()

{

//create collection as dynamic array

ArrayList al = new ArrayList();

//add elements in dyn arr

al.Add(1);

al.Add(2);

al.Add(3);

al.Add(4);

Console.Write("Current content: ");

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

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

Console.WriteLine("\n");

//get array

int[] ia = (int[])al.ToArray(typeof(int));

int sum = 0;

Console.Write("Current content: ");

for (int i = 0; i < ia.Length; i++)

sum += ia[i];

Console.WriteLine("Sum: " + sum);

}

}

В начале этой программы создается коллекция целых чисел. Затем в ней вызывается

метод ToArray() с указанием типа int получаемого массива. В итоге создается

целочисленный массив. Но поскольку Array является типом, возвращаемым методом

ToArray(), то содержимое подучаемого в итоге массива должно быть приведено к

типу int[].