Массивы

Описание массива и его инициализация

Массивы: *одномерные* (single-dimensional), многомерные *прямоугольные* (multidimensional), *массивы массивов — невыровненные* (jagged).

Описание:

```
char[] a;
int[,] b;
```

Создание с неявной инициализацией:

```
a = new char[5];
b = new int[3, 4];
```

Совмещение описания и инициализации:

```
char[] a = new char[5];
int[,] b = new int[3, 4];
```

Явная инициализация:

```
a = new char[] {'A', 'B', 'C', 'D', 'E'};
b = new int[,] {{1, 2, 3, 4}, {5, 6, 7, 8},
  {9, 10, 11, 12}};
```

Повторное создание:

```
a = new char[5];
...
a = new char[10];
```

Создание «на лету»:

```
M(new char[] {'A', 'B', 'C'});
```

Описание и неявная инициализация невыровненного двумерного целочисленного массива d с элементами $d_{0,0}$, $d_{1,0}$, $d_{1,1}$, $d_{2,0}$, $d_{2,1}$, $d_{2,2}$:

```
int[][] d = new int[3][];
d[0] = new int[1];
d[1] = new int[2];
d[2] = new int[3];
d[1][0] = 5;
```

Класс Array

Свойства и экземплярные методы

```
int Length { get; }
int Rank { get; }
int GetLength(int dimension);
int GetUpperBound(int dimension);
object Clone();
void CopyTo(Array a, int start);
```

Классовые методы

```
static void Clear(Array a, int start, int count);
static void Copy(Array src, Array dest, int count);
static void Copy(Array src, int srcStart, Array
    dest, int destStart, int count);
static void Reverse(Array a[, int start, int count]);
static void Sort(Array a[, Array b][, int start, int
    count][, System.Collections.IComparer cmp]);
static int IndexOf(Array a, object value[, int
    start[, int count]]);
static int LastIndexOf(Array a, object value[, int
    start[, int count]]);
```

Динамический массив: классы ArrayList и List<T>

Динамические массивы, в отличие от обычных, могут изменять свой размер после создания (с сохранением своего прежнего содержимого): System.Collections.ArrayList и System.Collections.Generic.List<T>

Наборы свойств и методов классов ArrayList и List<T> совпадают. Ниже описаны члены класса ArrayList. Всюду, где в описании члена класса ArrayList используется тип Array и object, в описании аналогичного члена класса List<T> надо использовать тип T[] и T соответственно.

Свойства

```
int Capacity { get; set; }
int Count { get; }
object [int index] { get; set; }
```

Конструкторы и другие способы создания динамического массива

```
ArrayList([int capacity]);
ArrayList(ICollection c);
object Clone();
static ArrayList Repeat(object value, int count);
```

Совместимость объектов Array и ArrayList

Обычные массивы и динамические массивы ArrayList не совместимы по присваиванию. Для преобразования обычного массива к объекту ArrayList необходимо воспользоваться соответствующим конструктором класса ArrayList.

```
object[] ToArray();
Array ToArray(Type type);
```

Пример:

```
double[] a = (double[])d.ToArray(typeof(double));
```

В классе List<T> метод ToArray не имеет параметров и возвращает массив типа T[].

```
void CopyTo(Array array[, int arrayStart]);
void CopyTo(int start, Array array, int arrayStart,
    int count);
```

Преобразование динамического массива

```
int Add(object value); // в List<T> имеет тип void
void Insert(int index, object value);
void AddRange(ICollection c);
void InsertRange(int start, ICollection c);
void SetRange(int start, ICollection c);
void Remove(object value);
void RemoveAt(int index);
void RemoveRange(int start, int count);
void Clear();
void TrimToSize();
void Reverse([int start, int count]);
void Sort([[int start, int count,]
    System.Collections.IComparer cmp]);
```

Поиск в динамическом массиве

```
bool Contains(object value);
int IndexOf(object value[, int start[, int count]]);
int LastIndexOf(object value[, int start[, int
    count]]);
```

Перебор элементов в цикле foreach

Признаком допустимости использования цикла foreach для объекта является наличие у этого объекта интерфейса IEnumerable.

Пример (a имеет тип ArrayList и содержит данные целого типа):

```
foreach (int v in a)
    s += v;
```

Без foreach:

```
for (int i = 0; i < a.Count; i++)
    s += (int)a[i];
```