Массивы

Определения

- *Массив* — ограниченная совокупность однотипных величин.
- Элементы массива имеют одно и то же имя, а различаются по порядковому номеру (*индексу*).
- Массив относится к ссылочным типам данных (располагается в хипе), поэтому *создание массива* начинается с выделения памяти под его элементы. *Элементами массива* могут быть величины как значимых, так и ссылочных типов (в том числе массивы), например:

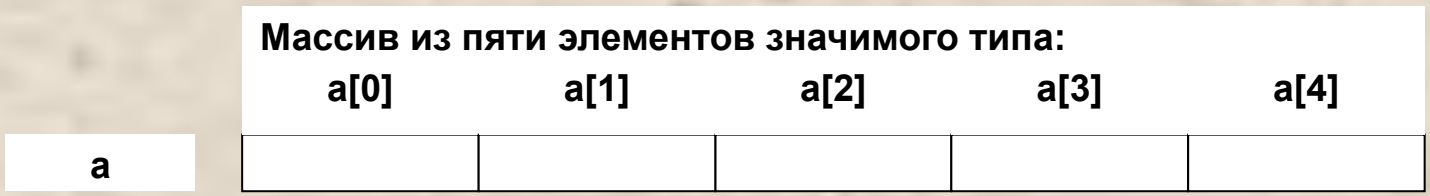
```
int[]    w = new int[10];          // массив из 10 целых чисел
string[] z = new string[100];   // массив из 100 строк
```

- Массив значимых типов хранит значения, массив ссылочных типов — ссылки на элементы.
- Всем элементам при создании массива присваиваются *значения по умолчанию*: нули для значимых типов и null для ссылочных.

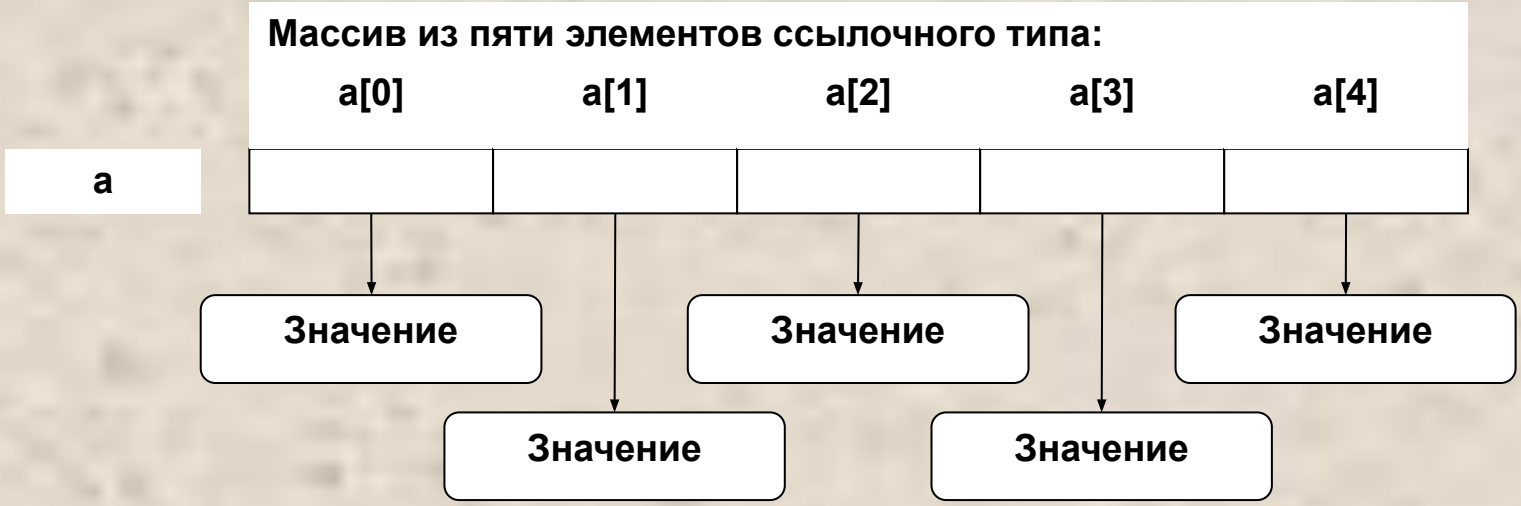
Размещение массивов в памяти

Пять простых переменных:

Массив из пяти элементов значимого типа:



Массив из пяти элементов ссылочного типа:



Размерность массива

- Количество элементов в массиве (*размерность*) задается при выделении памяти и не может быть изменена впоследствии. Она может задаваться выражением:

```
short n = …;
string[] z = new string[n + 1];
```

- Размерность не является частью типа массива.
- Элементы массива нумеруются *с нуля*.
- Для *обращения к элементу массива* после имени массива указывается номер элемента в квадратных скобках, например:

```
w[4]      z[i]
```

- С элементом массива можно делать все, что допустимо для переменных того же типа.
- При работе с массивом автоматически выполняется *контроль выхода за его границы*: если значение индекса выходит за границы массива, генерируется исключение IndexOutOfRangeException.

Действия с массивами

- Массивы одного типа можно *присваивать* друг другу. При этом происходит присваивание ссылок, а не элементов:

```
int[] a = new int[10];
int[] b = a;        // b и a указывают на один и тот же массив
```

- Все остальные действия выполняются с элементами массива по отдельности:

```
for ( int i = 0; i < n; ++i ) Console.Write( "\t" + a[i] );
```

Одномерные массивы

- Варианты описания массива:

тип[] имя;

тип[] имя = new тип [размерность];

тип[] имя = { список_инициализаторов };

тип[] имя = new тип [] { список_инициализаторов };

тип[] имя = new тип [размерность] { список_инициализаторов };

- Примеры описаний (один пример на каждый вариант описания, соответственно):

```
int[] a;                                    //  элементов нет
int[] b = new int[4];                       //  элементы равны 0
int[] c = { 61, 2, 5, -9 };                 //  new подразумевается
int[] d = new int[] { 61, 2, 5, -9 };   // размерность вычисляется
int[] e = new int[4] { 61, 2, 5, -9 };      // избыточное описание
```

Пример

Для массива, состоящего из 6 целочисленных элементов, программа определяет:

- сумму и количество отрицательных элементов;
- максимальный элемент.

Программа

```
const int n = 6;
int[] a = new int[n] { 3, 12, 5, -9, 8, -4 };

Console.WriteLine( "Исходный массив:" );
for ( int i = 0; i < n; ++i ) Console.Write( "\t" + a[i] );
Console.WriteLine();

long sum = 0;              // сумма отрицательных элементов
int  num = 0;              // количество отрицательных элементов
for ( int i = 0; i < n; ++i )
      if ( a[i] < 0 ) {
          sum += a[i];  ++num;
      }
 Console.WriteLine( "Сумма  отрицательных = " + sum );
 Console.WriteLine( "Кол-во отрицательных = " + num );

 int max = a[0];            // максимальный элемент
 for ( int i = 0; i < n; ++i )
          if ( a[i] > max ) max = a[i];
 Console.WriteLine( "Максимальный элемент = " + max );
```

Оператор foreach (упрощенно)

- Применяется для перебора элементов массива. Синтаксис:

foreach (тип имя in имя_массива) тело_цикла

- *Имя* задает локальную по отношению к циклу переменную, которая будет по очереди принимать все значения из массива, например:

```
int[] a =  { 24, 50, 18, 3, 16, -7, 9, -1 };
foreach ( int x in a ) Console.WriteLine( x );
```

Программа с использованием foreach

```
int[] a = { 3, 12, 5, -9, 8, -4 };
Console.WriteLine( "Исходный массив:" );
foreach ( int elem in a )
      Console.Write( "\t" + elem );
Console.WriteLine();

long sum = 0;        // сумма отрицательных элементов
int  num = 0;         // количество отрицательных элементов
foreach ( int elem in a )
        if ( elem < 0 ) {
              sum += elem;  ++num;
        }
Console.WriteLine( "sum = " + sum );
Console.WriteLine( "num = " + num );

int max = a[0];             // максимальный элемент
foreach ( int elem in a )
       if ( elem > max ) max = elem;
Console.WriteLine( "max = " + max );
```

```
for ( int i = 0; i < n; ++i )
       if ( a[i] < 0 ) {
       sum += a[i];  ++num;
       }
```

System.Array

- Все массивы в C# имеют общий базовый класс Array, определенный в пространстве имен System.
- Некоторые элементы класса Array:
  - Length (Свойство) - Количество элементов массива (по всем размерностям)
  - BinarySearch (Статический метод) - Двоичный поиск в отсортированном массиве
  - IndexOf – (Статический метод) - Поиск первого вхождения элемента в одномерный массив
  - Sort (Статический метод) - Упорядочивание элементов одномерного массива

Использование методов класса Array

```
static void Main()
        {
            int[] a =  { 24, 50, 18, 3, 16, -7, 9, -1 };
            PrintArray( "Исходный массив:", a );
            Console.WriteLine( Array.IndexOf( a, 18 ) );
            Array.Sort(a);
            PrintArray( "Упорядоченный массив:", a );
            Console.WriteLine( Array.BinarySearch( a, 1 8) );
        }
public static void PrintArray( string header, int[] a )
        {
            Console.WriteLine( header );
            for ( int i = 0; i < a.Length; ++i )
                Console.Write( "\t" + a[i] );
            Console.WriteLine();
        }
```

Прямоугольные массивы

- *Прямоугольный массив* имеет более одного измерения. Чаще всего в программах используются двумерные массивы. Варианты описания двумерного массива:

тип[,] имя;

тип[,] имя = new тип [разм_1, **разм_2]**;

тип[,] имя = { список_инициализаторов };

тип[,] имя = new тип [,] { список_инициализаторов };

тип[,] имя = new тип [разм_1, разм_2] { список_инициализаторов };

- Примеры описаний (один пример на каждый вариант описания):

```
int[,] a;                                   // элементов нет
int[,] b = new int[2, 3];                   // элементы равны 0
int[,] c = {{1, 2, 3}, {4, 5, 6}};          // new подразумевается
int[,] c = new int[,] {{1, 2, 3}, {4, 5, 6}};   // разм-сть вычисляется
int[,] d = new int[2,3] {{1, 2, 3}, {4, 5, 6}}; // избыточное описание
```

- К элементу двумерного массива обращаются, указывая номера строки и столбца, на пересечении которых он расположен:

a[1, 4] b[i, j] b[j, i]

- Компилятор воспринимает как номер строки первый индекс, как бы он ни был обозначен в программе.

- Пример: программа, которая для целочисленной матрицы размером 3 x 4 определяет среднее арифметическое ее элементов и количество положительных элементов в каждой строке.

```
const int m = 3, n = 4;
int[,] a = new int[m, n] {
    { 2,-2, 8, 9 },
    {-4,-5, 6,-2 },
    { 7, 0, 1, 1 }
 };
Console.WriteLine( "Исходный массив:" );
for ( int i = 0; i < m; ++i )
{
    for ( int j = 0; j < n; ++j )
        Console.Write( "\t" + a[i, j] );
    Console.WriteLine();
}
```

```
double sum = 0;
int nPosEl;
for ( int i = 0; i < m; ++i )
{
    nPosEl = 0;
    for ( int j = 0; j < n; ++j )
    {
        sum += a[i, j];
        if ( a[i, j] > 0 ) ++nPosEl;
    }
    Console.WriteLine( "В строке {0} {1} положит-х элементов",
                        i, nPosEl  );
}
Console.WriteLine( "Среднее арифметическое всех элементов: "
                    + sum / m / n );
```