Лекция 16

Массивы. В
вод и вывод одномерных и двухмерных массивов. Операции над массивами

Цель: 1 Освоить правила ввода и вывода одномерных и двумерных массивов.

2 Научиться осуществлять простейшие операции над массивами.

Массив — это структурированный **тип** данных, содержащий **определенное число *переменных* (*элементов*) одинакового базового типа**, доступ к которым осуществляется с помощью ***порядковых номеров*** (***индексов***) и общего имени массива.

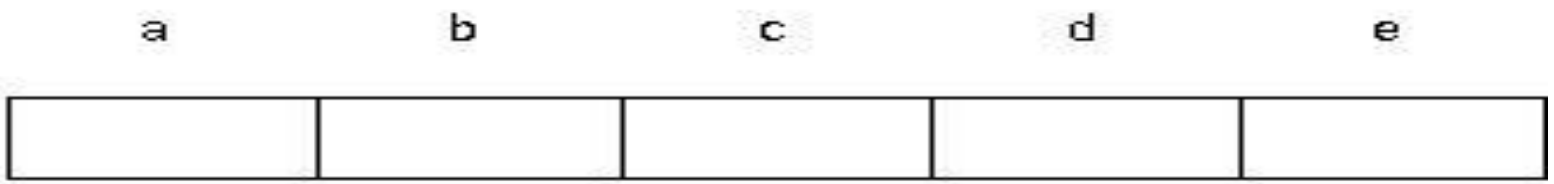
Все массивы в **C#** имеют общий базовый *класс* **Array**, определенный в пространстве имен **System**.

Массив в **C#** относится к *ссылочным типам данных*, то есть располагается в динамической области памяти, поэтому *создание массива* начинается с выделения памяти под его элементы с помощью оператора **new**.

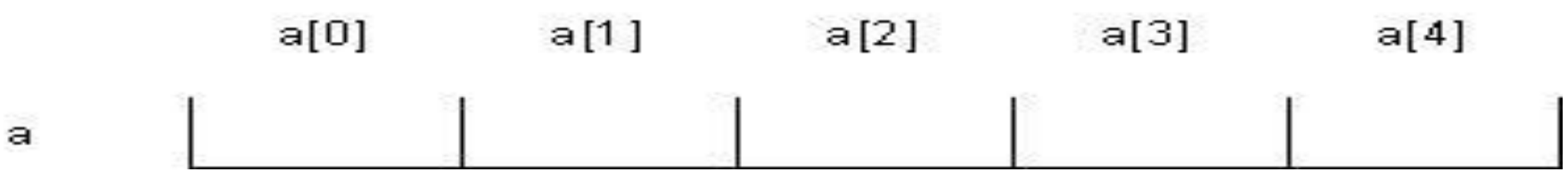
Элементам массива присваиваются индексы в диапазоне от **0** до **Length -** 1. Элементами массива могут быть величины как значимых, так и ссылочных **базовых типов** (в том числе массивы).

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

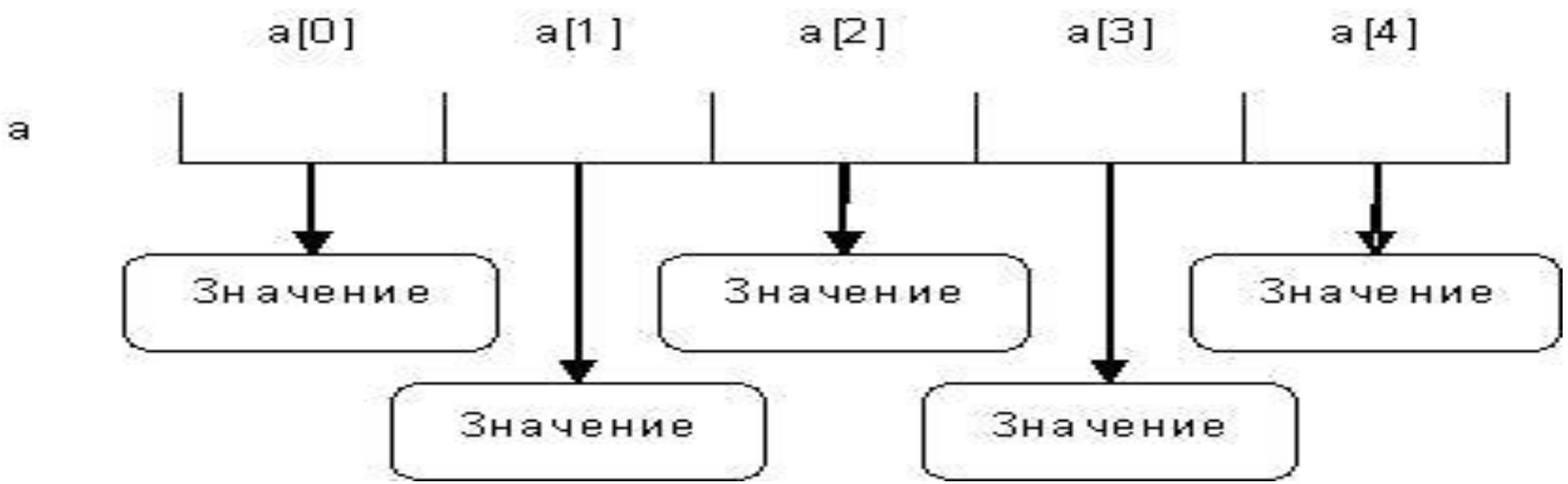
Пять простых переменных



Массив из пяти элементов значимого типа:



Массив из пяти элементов ссылочного типа:



Одномерные массивы

тип_переменной [*размерность*] название_массива;

Размерность может задаваться не только **константой**, но и **константным выражением**. Результат вычисления этого выражения должен быть неотрицательным, а его тип должен иметь *неявное преобразование* к **int**, **uint**, **long** или **ulong**.

Параметр ***тип_переменной*** определяет базовый тип данных каждого элемента, составляющего массив.

Перед тем, как к массивам можно будет получить доступ, они должны быть обязательно **инициализированы**. Это можно сделать при описании неявно с помощью операции **new** (нулями) или явно любыми значениями.

Варианты описания одномерного массива

тип[] имя; // объявление без выделения памяти
имя = **new** тип [размерность];

тип[] имя = **new** тип [размерность];

тип[] имя = {список_инициализаторов};

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

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

Примеры описаний одномерного массива

```
int[] a;                                        // 1
w = new int[6];                                 // 2
```

```
int[] b = new int[4];                           // 3
int[] d = new int[] { 61, 2, 5, -9 };           // 4
int[] e = new int[4] { 61, 2, 5, -9 };          // 5
int[] c = { 61, 2, 5, -9 };                     // 6
```

```
const int arSize = 5;                           // 7
int[] myAr = new int [arSize] {5, 9, 10, 2, 99}; // 8
```

Примеры описаний одномерного массива

Оператор 5 эквивалентен следующему коду:

```
int[] e = new int[4];
e[0] = 61;
e[1] = 2;
e[2] = 5;
e[3] = -9;
```

Размерность (должна быть константой) и ***количество инициализаторов*** должны обязательно совпадать:

```
int[] e = new int[10] { 61, 2, 5, -9 };  // ошибка
```

Одномерные массивы

Индекс (номер элемента) обозначает положение элемента в массиве. *Элементы массива нумеруются с нуля*, поэтому индекс первого элемента будет равен 0, а максимальный номер элемента всегда на единицу меньше размерности.

Для *обращения к элементу массива* после имени массива указывается **индекс** в квадратных скобках, например, чтобы обратиться к четвертому элементу в массиве, надо использовать индекс 3: myAr[3].

myAr[i] - обращение к элементу с индексом i

Пример 1

```
using System;
class Test{
   static void Main() {
      int[] a = new int[10] { 3, 12, 5, -9, 8, -4, 7, -4,
   13, 10 };
     for (int i = 0; i < a.Length; i++)
     Console.WriteLine("a[{0}] = {1}", i, a[i]);
   }
}
```

Пример 2

```
using System;
namespace ConsoleApplication1{
   class Class1{
      static void Main(){
        const int n = 6;
        int[] a = new int[n];
        a[0] = 3;          a[1] = 12;          a[2] = 5;
        a[3] = -9;         a[4] = 8;           a[5] = -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
```

Пример 2

```
        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 = 1; i < n; ++i)
            if (a[i] > max) max = a[i];
        Console.WriteLine("Максимальный элемент = " +
    max);
      }
    }
}
```

Пример 3

```
using System;
class Average {
    static void Main() {
        int[] nums = new int[10];
        int n = 10;
        double avg = 0;
        Console.WriteLine( "Введите элементы массива:"
   );
        for (int i = 0; i < n; ++i)  {
            Console.Write( " nums [" + (i+1) + "] = ");
            nums [i] = Convert.ToInt32(Console.ReadLine());
        }
```

Пример 3

```
        Console.WriteLine();
        foreach (int a in nums) avg = avg + a;
        avg = avg / n;
        Console.WriteLine("Среднее значение массива: " +
    avg);
        Console.ReadKey();
     }
}
```

```
int[] b = nums;  // b и nums указывают на один и тот же массив
Console.WriteLine(" массив b:");
for (int i = 0; i < n; ++i)   Console.Write("\t" + b[i]);
```

Многомерные массивы

Число измерений называется ***рангом*** типа массива и определяется как сумма единицы и числа запятых, указанных в квадратных скобках типа массива.

Выделение памяти для одно-, двух- и трехмерного массивов:

```
int[] a1 = new int[10];
int[ , ] a2 = new int[10, 5];
int[ , , ] a3 = new int[10, 5, 2];
```

В **C#** существуют разновидности многомерных массивов: **прямоугольные** и **ступенчатые** (невыровненные, неравномерные, зубчатые).

Прямоугольный массив

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 };

Прямоугольный массив

Примеры описаний прямоугольного массива:

```
int[,] a;                                   // 1 элементов нет
a = new int[2, 3];                          // распределение памяти
int[,] b = new int[2, 3];                                   // 2
int[,] c = {{1, 2, 3}, {4, 5, 6}};                          // 3
int[,] c = new int[,] {{1, 2, 3}, {4, 5, 6}};               // 4
int[,] d = new int[2,3] {{1, 2, 3}, {4, 5, 6}};             // 5
const int firstIdx = 2;
const int secondIdx = 3;
string[,] arr = new string[firstIdx, secondIdx];            // 6
```

Прямоугольный массив

Инициализация прямоугольного массива:

```
тип[ , ] имя_массива = {
    {val, val, val, ..., val},
    {val, val, val, ..., val},
    {val, val, val, ..., val}
};
```

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

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

Пример 4

```
using System;
namespace ConsoleApplication1{
    class Class1{
        static void Main(){
            const int m = 3, n = 4;
            int[,] a = new int[m, n] {
                { 2, -2, 8, 9 },
                {-4, -5, 6,-2 },
                { 7, 0, 1, 1 }
             };
```

Пример 4

```
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 nPosE1;
```

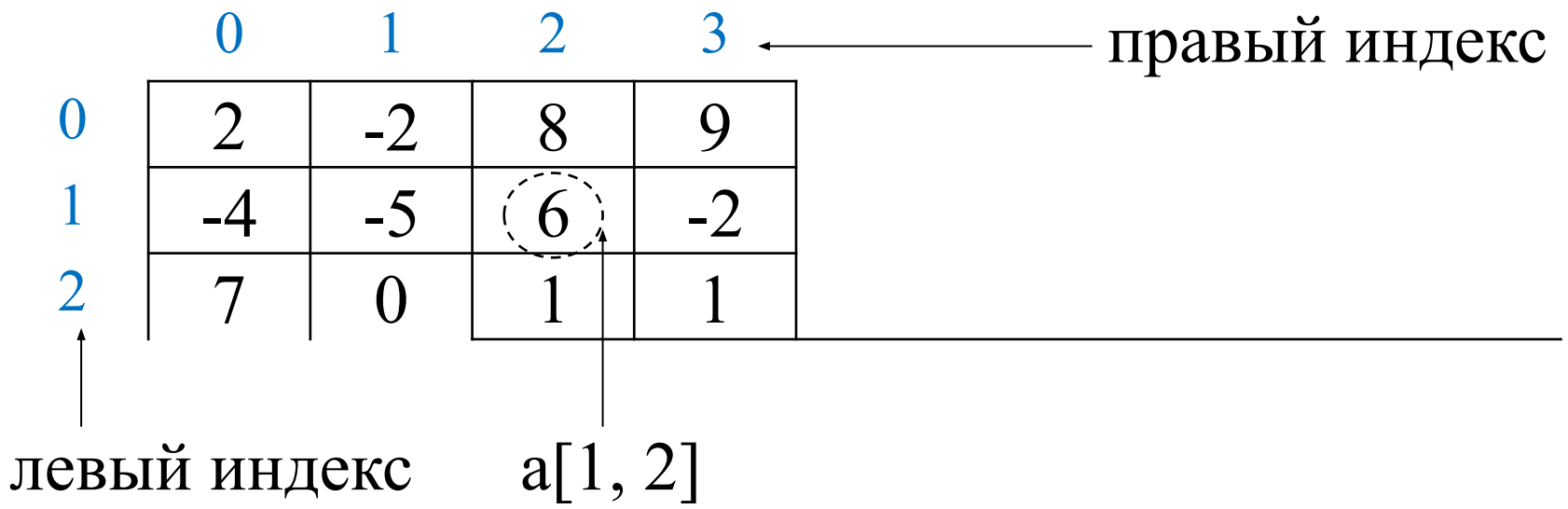
Пример 4

```
        for ( int i = 0; i < m; ++i )  {
            nPosE1 = 0;
            for ( int j = 0; j < n; ++j ) {
                sum += a[i, j];
                if ( a[i, j] > 0 ) ++nPosE1;
            }
            Console.WriteLine("В строке {0} {1}
              положительных элементов", i, nPosE1);
        }
        Console.WriteLine("Среднее арифметическое
             всех элементов: " + sum / m / n);
    }
}
}
```

Пример 4

В данном примере элемент массива **a** [0,0] будет иметь значение 2, элемент массива **a** [0,1] — значение -2, элемент массива **a** [0,2] — значение 8 и т.д.

А значение элемента массива **a** [2,3] окажется равным 1.



Многомерный массив

В **C#** допускаются массивы **трех и более измерений**.

тип[, . . .,] имя_массива = **new** тип[размер1, размер2, . . . размеры];

Создание трехмерного целочисленного массива размером 4x10x3:

```
int[ , , ] multidim = new int[4, 10, 3];
```

Элементу массива **multidim** с координатами местоположения (2,4,1) присваивается значение 100:

```
multidim[2, 4, 1] = 100;
```

Пример 5

```
class ThreeDMatrix {
    static void Main() {
        int[ , , ] m = new int[3, 3,3];
        int sum = 0;
        int n = 1;
        for (int x=0; x < 3; x++)
            for (int y=0; y < 3; y++)
                for (int z=0; z < 3; z++)
                    m[x, y, z] = n++;
        sum =m[0, 0, 0] + m[l, 1, 1] +m[2, 2, 2];
        Console.WriteLine("Сумма значений по первой
   диагонали: " + sum);
    }
}
```

Ступенчатые (невыровненные, неравномерные, зубчатые) массивы

В ступенчатых массивах (**jagged array**, или **массив массивов**) количество элементов в разных строках может различаться. В памяти ступенчатый массив хранится иначе, чем прямоугольный: в виде нескольких внутренних массивов, каждый из которых имеет свой размер.

Для объявления двумерного ступенчатого массива служит следующая общая форма:

тип[] [] имя_массива = **new** тип [размер] [];

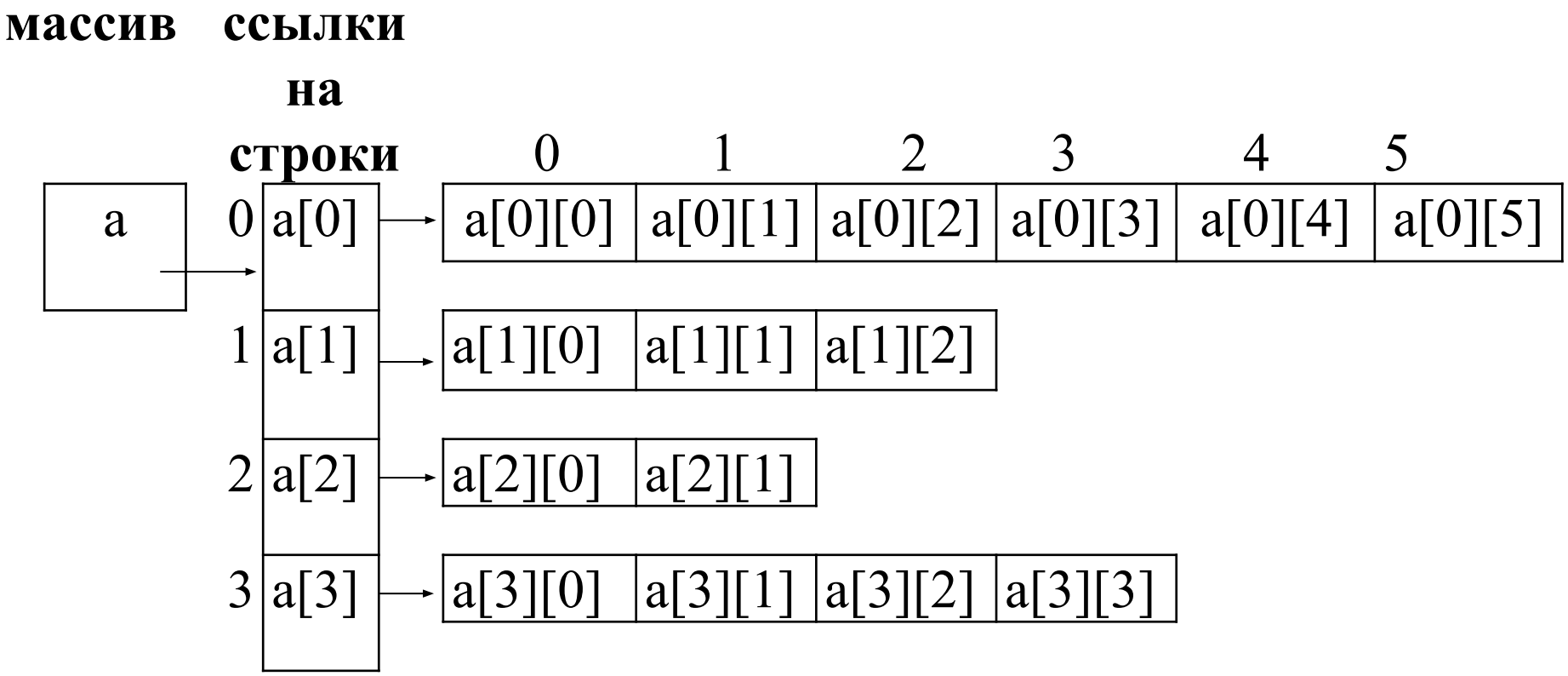
где **размер** обозначает число строк в массиве. Память для самих строк распределяется индивидуально, и поэтому длина строк может быть разной.

Ступенчатые (невыровненные, неравномерные, зубчатые) массивы

Например, в объявлении ступенчатого массива **a** память сначала распределяется для его первого измерения автоматически, а затем для второго измерения вручную.

```
int[][] a = new int[4][];
a[0] = new int[6];
a[1] = new int[3];
a[2] = new int[2];
a[3] = new int[4];
```

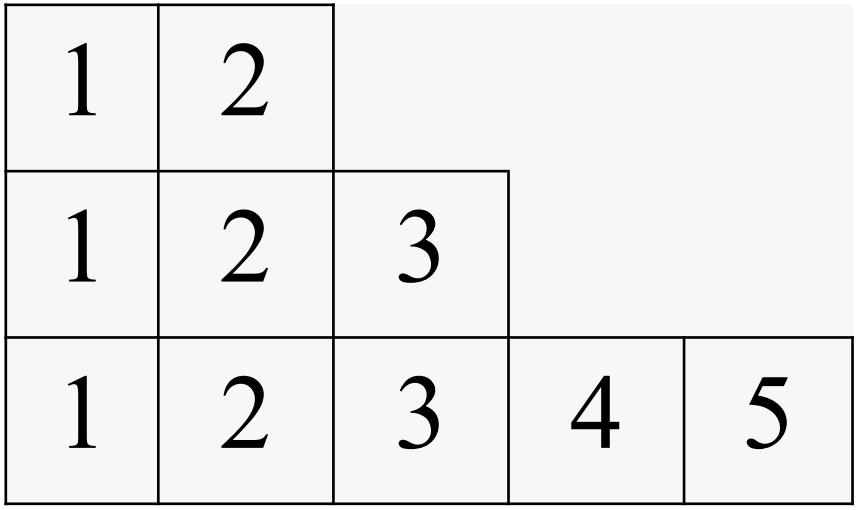
Ступенчатые (невыровненные, неравномерные, зубчатые) массивы



Ступенчатые (невыровненные, неравномерные, зубчатые) массивы

Зубчатый массив при описании может быть инициализирован:

```
int[][] nums = new int[3][];
nums[0] = new int[2] { 1, 2 };
nums[1] = new int[3] { 1, 2, 3 };
nums[2] = new int[5] { 1, 2, 3, 4, 5 };
```



Элемент массива **nums** на позиции с координатами (2,1) находится в третьей строке и во втором столбце:

```
nums [2][1] = 10;
```

Пример 6

```
static void Main() {
    int[][] jagged = new int[3][];
    jagged[0] = new int[4];
    jagged[1] = new int[3];
    jagged[2] = new int[5];
    int i;
    for (i=0; i < 4; i++)  jagged[0][i] = i;
    for (i=0; i < 3; i++) jagged[1][i] = i;
    for (i=0; i < 5; i++) jagged[2][i] = i;
    Console.WriteLine( "Исходный массив:" );
    foreach (int [] mas in jagged)   {
    foreach (int x in mas)  Console.Write( "\t" + x );
    Console.WriteLine();
}
```

Ступенчатые (невыровненные, неравномерные, зубчатые) массивы

Ступенчатые массивы представляют собой **массивы массивов**, и поэтому они не обязательно должны состоять из одномерных массивов.

Например, можно создать массив двумерных массивов.

```
int[] [,] nums = new int[3] [,];
```

В следующей строке кода элементу массива **nums** [0] присваивается ссылка на массив размерами 4x2.

```
nums [0] = new int [4, 2];
```

В приведенной ниже строке кода элементу массива **nums** [0] [1,0] присваивается значение переменной i.

```
nums [0] [1,0] = i;
```

При описании такой массив так же можно инициализировать.

Пример 7

```
using System;
class Lego {
    static void Main() {
        int[][,] nums = new int[3][,] {
            new int[,] { {1,2}, {3,4} },
            new int[,] { {1,2,-1}, {3,6,2}, {-6,4,-2} },
            new int[,] { {1,2}, {3,5}, {8, 13}, {2,-5} }
        };
        int k = 0;
        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 2; j++)
                Console.WriteLine("nums [" + k + "][" + i +
", " + j + "] = " + nums[k][i, j]);
        k = 1;
```

Пример 7

```
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                Console.WriteLine("nums [" + k + "][" + i + ", " + j +
"] = " + nums[k][i, j]);
        k = 2;
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 2; j++)
                Console.WriteLine("nums [" + k + "][" + i + ", " + j +
"] = " + nums[k][i, j]);
        // или вывод каждого двумерного массива одной строкой
        foreach (int[,] mas in nums){
            foreach (int x in mas) Console.Write("\t" + x);
            Console.WriteLine();
        }
    }
}
```

Класс System.Array

Метод или свойство	Описание
BinarySearch()	Перегруженный открытый статический метод, выполняющий поиск в одномерном упорядоченном массиве
Clear()	Открытый статический метод, который приравнивает диапазон элементов массива к нулю или к нулевой ссылке
Copy()	Перегруженный открытый статический метод, копирующий фрагмент одного массива в другой
CreateInstance()	Перегруженный открытый статический метод, создающий новый экземпляр массива
IndexOf()	Перегруженный открытый статический метод, возвращающий индекс (смещение от начала) первого значения в одномерном массиве

Класс System.Array

Метод или свойство	Описание
LastIndexOf()	Перегруженный открытый статический метод, возвращающий индекс последнего значения в одномерном массиве
Reverse()	Перегруженный открытый статический метод, меняющий порядок следования элементов одномерного массива на обратный
Sort()	Перегруженный открытый статический метод, сортирующий элементы одномерного массива
IsFixedSize	Открытое свойство, возвращающее логическое значение, которое сообщает, фиксирован ли размер массива
IsReadOnly	Открытое свойство, возвращающее логическое значение, которое сообщает, доступен ли массив только для чтения

Класс System.Array

Метод или свойство	Описание
IsSynchronized	Открытое свойство, возвращающее логическое значение, которое сообщает, обеспечивает ли массив безопасную работу с несколькими потоками
Length	Открытое свойство, возвращающее длину массива
Rank	Открытое свойство, возвращающее число измерений массива
SyncRoot	О ткрытое свойство, возвращающее объект, с помощью которого можно синхронизировать доступ к массиву
GetEnumerator()	Открытый метод, возвращающий объект типа IEnumerator

Класс System.Array

Метод или свойство	Описание
GetLength()	Открытый метод, возвращающий длину указанного измерения массива
GetLowerBound()	Открытый метод, возвращающий нижнюю границу указанного измерения массива
GetUpperBound()	Открытый метод, возвращающий верхнюю границу указанного измерения массива
GetValue()	Открытый метод, возвращающий значение элемента массива
Initialize()	Инициализирует все элементы массива (имеющие размерный тип), вызывая для каждого конструктор по умолчанию
SetValue()	Перегруженный открытый метод, присваивающий значение указанному элементу массива

Пример 8

```
using System;
namespace ConsoleApplication1{
    class Class1{
      static void Main(){
          int[] a = {24, 50, 18, 3, 16, -7, 9, -1};
          PrintArray( "Исходный массив:", a );
          Console.WriteLine("Поиск первого вхождения
    элемента 18:  индекс =  " + Array.IndexOf(a, 18));
          Array.Sort(a);
          PrintArray( "Упорядоченный массив:", a );
          Console.WriteLine("Двоичный поиск в
    отсортированном массиве элемента 18: индекс =  " +
    Array.BinarySearch(a, 18));
      }
```

Пример 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();
    }
  }
}
```

Методы **Sort**, **IndexOf** и **BinarySearch** являются статическими, поэтому к ним обращаются через имя класса, а не экземпляра, и передают в них имя массива.

Пример 9

```
namespace Multidimensional{
  class MainApp{
    static void Main(string[] args] {
      string[ , ] arr;
      const int firstldx = 2;
      const int secondldx = 3;
      arr = new string[firstldx, secondldx];
      for (int i = 0; i < firstldx; i++){
          Console.WriteLine((i+1) + " строка массива:");
          for (int j = 0; j < secondldx; j++){
              Console.WriteLine((j+1) + " строка - ");
              arr[i,j] = Console.ReadLine();
          }
          Console.WriteLine();
      }
```

Пример 9

```
        int Rank = arr.Rank;
        Console.WriteLine("Массив arr имеет ранг {0}",
Rank);  // 2
         int Len = arr.Length;
         Console.WriteLine("Количество элементов в
массиве: {0}", Len);  // 6
         for (int i = 0; i < firstldx; i++) {
            for (int j = 0; j < secondldx; j++)
                Console.Write(arr[i,j] + "\t");
             Console.WriteLine();
         }
    }
 }
```

Пример 10

```
using System;
class Jagged {
    static void Main() {
        int[][] network_nodes = new int[4][];
        network_nodes[0] = new int[3];
        network_nodes[1] = new int[7];
        network_nodes[2] = new int[2];
        network_nodes[3] = new int[5];
        int i, j;
       // Сфабриковать данные об использовании ЦП
        for (i=0; i < network_nodes.Length; i++)
            for (j=0; j < network_nodes[i].Length; j++)
                network_nodes[i][j] = i * j + 70;
```

Пример 10

```
        Console.WriteLine("Общее количество узлов
сети: " + network_nodes.Length + "\n");
        for (i=0; i < network_nodes.Length; i++) {
            for (j=0; j < network_nodes[i].Length; j++) {
                Console.Write("Использование в узле сети
" + i + " ЦП " + j + ": ");
                Console.Write(network_nodes[i] [j] + "% ");
                Console.WriteLine();
            }
            Console.WriteLine() ;
        }
    }
}
```

Пример 11

```
using System;
namespace ConsoleApplication1{
   class Monster {
      public int a;
      public int b;
      public void Pas(){
         Console.WriteLine(a+" + "+b+" = "+(a+b));
      }
   }
```

Пример 11

```
class Class1{
   static void Main(){
      Random rnd = new Random();
      const int n = 5;
      Monster[] stado = new Monster[n];
      for ( int i = 0; i < n; ++i )  {
         stado[i] = new Monster( rnd.Next(1, 100),
rnd.Next(1, 200));
      }
      foreach ( Monster x in stado )  x.Pas();
   }
}
}
```

Неявно типизированные массивы

Неявно типизированный массив объявляется с помощью ключевого слова **var**, но без последующих квадратных скобок [ ], поскольку по типу инициализаторов определяется тип элементов данного массива.

Создание массива типа **int**, состоящего из пяти элементов:

```
var vals = new[] { 1, 2, 3, 4, 5 };
```

Создание двумерного массива типа **double**размерами 2x3:

```
var vals = new[,] {{1.1, 2.2}, {3.3, 4.4},{ 5.5, 6.6}};
```

Пример 12

```
class Jagged {
    static void Main() {
        var jagged = new[] {
            new[] { 1, 2, 3, 4 },
            new[] { 9, 8, 7 },
            new[] { 11, 12, 13, 14, 15 }
        };
        for (int j = 0; j < jagged.Length; j++) {
            for (int i = 0; i < jagged[j].Length; i++)
                Console.Write(jagged[j][i] + " ");
            Console.WriteLine();
    }
}
```

Пример 13

```
namespace SortApp{
    class Program  {
        static void Main(string[] args) {
            // ввод чисел
            double [] nums = new double [7];
            Console.WriteLine("Введите семь чисел");
            for (int i = 0; i < nums.Length; i++) {
                Console.Write("{0}-е число: ", i + 1);
                nums[i] = double.Parse(Console.ReadLine());
            }
            double temp;
```

Пример 13

```
        for (int i = 0; i < nums.Length-1; i++) {
            for (int j = i + 1; j < nums.Length; j++)  {
                if (nums[i] > nums[j]) {
                    temp = nums[i];
                    nums[i] = nums[j];
                    nums[j] = temp;
                }
            }
        }
        Console.WriteLine("Вывод отсортированного массива");
        foreach ( int x in nums )  Console.WriteLine(x + "\t");
        Console.ReadLine();
    }
}
```

Пример 14

```
using System;
namespace Sorting{
    class TestApp{
        public static void Main(){
            int[] arr;
            arr = new int[10];
            Console.WriteLine("Исходный массив: ");
            for (int i = 0; i < arr.Length; i++) {
                arr[i]= 10 - i;
                Console.Write("{0} ", arr [i]);
            }
            bool bSort = false;
```

Пример 14

```
        do{
            bSott = false;
            for (int i = 1; i < arr.Length; i++)
                if (arr [i] < arr[i-1]) {
                    int c = arr[i];          arr[i] = arr[i -1];
                    arr[i-1] = c;            bSort = true;
                }
        } while(bSort);
        Console.WriteLine("Массив:");
        for (int i = 0; i < arr.Length; i++)
            Console.Write{"{0} ", arr[i]);
    }
}
}
```

Ключевое слово params

Ключевое слово **params** позволяет передавать методу переменное количество параметров без обязательного явного создания массива:

```
public void DisplayVals(params int[] intVals)
```

Метод обрабатывает эту конструкцию так, словно массив целых был явно создан и передан в качестве аргумента. Элементы такого массива можно перебрать в цикле, как и элементы любого другого целочисленного массива:

```
foreach (int i in intVals) {
    Console.WriteLine("DisplayVals {0}", i);
}
```

Ключевое слово params

При этом вызывающий метод вовсе не обязан создавать массив явно.

Будет достаточно, если он передаст целые числа, а компилятор соберет аргументы в массив для метода DisplayVals():

```
t.DisplayVals(5,6,7,8);
```

Впрочем, если программист предпочтет передать массив, ничто не помешает сделать это:

```
int [] Arr = new int[5] {1,2,3,4,5};
t.DisplayVals(Arr);
```

Пример 15

```
namespace Programming_CSharp {
    public class Tester {
      static void Main(){
          Tester t = new Tester();
          t.DisplayVals(5,6,7,8);
          int []Arr = new int[5] {1,2,3,4,5};
          t.DisplayVals(Arr);
      }
      public void DisplayVals(params int[] intVals) {
        foreach (int i in intVals) {
          Console. WriteLine("DisplayVals {0}", i);
        }
        Console. WriteLine();
    }
  }
}
```

Контрольные вопросы

1. Что такое массив?
2. Является ли размерность частью описания массива?
3. Может ли размерность массива описана переменной (а не константой)?
4. Можно ли изменить размерность массива после выделения памяти под него?
5. Какие виды массивов существуют в C#?
6. Что происходит, если количество инициализаторов массива не соответствует заявленной размерности?
7. Что происходит при присваивании массивов?
8. В чем отличие сортировки массива методом пузырька от линейного метода?
9. Опишите основные методы и свойства класса System.Array.

Домашнее задание

Задание 1: Напишите программу, выполняющую подсчет числа отрицательных элементов двумерного массива, содержащего 6x8 целых чисел, значения которого находятся в интервале с границами [-50; 40].

Задание 2: Напишите программу, определяющую максимальный элемент двумерного массива, содержащего 4x5 вещественных чисел, введенных пользователем.