# Массивы. Класс Array

*Массив* представляет собой совокупность переменных одного типа с общим *для* обращения к ним именем.

В C# выделяют:

- - одномерные массивы;
- - многомерные массивы.

Одномерный массив представляет собой список связанных переменных.

Объявление :

Тип [] имя_массива = new тип [размер];

где *тип* объявляет конкретный тип элемента массива. Квадратные скобки указывают на то, что объявляется одномерный массив. А размер определяет число элементов массива.

**Пример:**

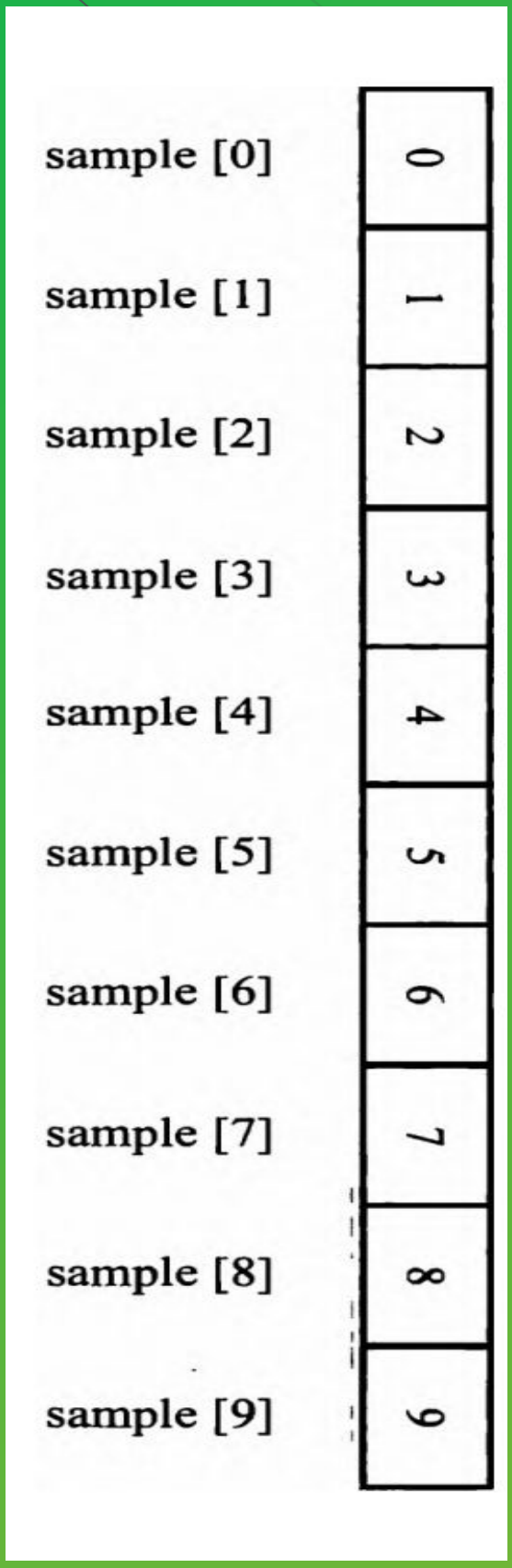
```
int[] sample = new int[10];

int[] sample;
sample = new int[10];
```

Доступ к отдельному элементу массива осуществляется по индексу. Индекс обозначает положение элемента в массиве.

В языке C# индекс первого элемента всех массивов оказывается нулевым.

```
using System;
class ArrayDemo  {
static void Main()    {
int[]   sample = new  int[10];
int  i;
for(i = 0;   i <  10;   i = i+1) sample[i]   =  i;
for(i = 0;   i <  10;   i = i+1)
Console.WriteLine("sample["  + i  + "]:   "  +  sample[i]);
```

```
sample[0]: 0
sample[1]: 1
sample[2]: 2
sample[3]: 3
sample[4]: 4
sample[5]: 5
sample[6]: 6
sample[7]: 7
sample[8]: 8
sample[9]: 9
```


sample [0]
sample [1]
sample [2]
sample [3]
sample [4]
sample [5]
sample [6]
sample [7]
sample [8]
sample [9]
0
1
2
3
4
5
6
7
8
9

*Инициализация массива*

Тип[] имя_массива = {val1, val2, val3,  ....  ,valN};

где val1-valN обозначают первоначальные значения, которые присваиваются по очереди, слева направо и по порядку индексирования

```
//  Вычислить среднее арифметическое ряда значений.
using System;
class Average   {
static void Main ()   {
int[]   nums = { 99, 10, 100, 18, 78, 23,63, 9, 87, 49  };
int avg =  0;
for(int  i=0; i < 10; i++)
avg = avg  +  nums[i];
avg = avg  /  10;
Console.WriteLine("Среднее:  "  + avg); }}
```

При инициализации массива его размер можно указывать явным образом, но этот размер должен совпадать с числом инициализаторов. В качестве примера ниже приведен еще один способ инициализации массива ***nums***.

```
int[] nums = new int[ ] { 99, 10, 100, 18, 78,  23, 63,   9,   87,   49  };
```

```
int[] nums = new int[10] { 99, 10, 100, 18, 78,  23, 63,   9,   87,   49  };
```

Обязательно нужно указать!

```
int[] nums;
nums = new int[] { 99, 10, 100, 18, 78, 23, 63, 9, 87,49 };
```

Границы массива в C# строго соблюдаются. Если границы массива не достигаются или же превышаются, то возникает ошибка при выполнении.

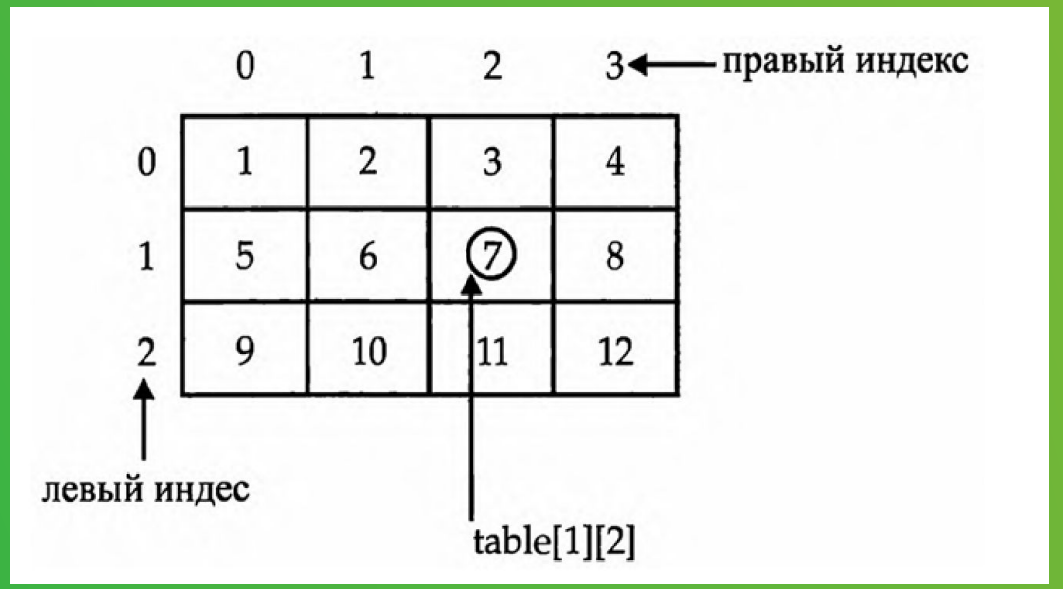
```
//  Продемонстрировать  превышение  границ массива.
using System;
class ArrayErr   {
static void Main ()   {
int[] sample = new int[10];
int  i;
//  Воссоздать превышение  границ массива.
for(i  = 0;   i  <  100;   i = i+1)
sample[i]   = i; } }
```

*Многомерным* называется такой массив, который отличается двумя или более измерениями, причем доступ к каждому элементу такого массива осуществляется с помощью определенной комбинации двух или более индексов.

Простейшей формой многомерного массива является *двумерный* массив. Местоположение любого элемента в двумерном массиве обозначается двумя индексами.

```
int[,]   table = new  int[10,   20];
table[3,   5]   =  10;
```

```
// Продемонстрировать двумерный массив.
using System;
class TwoD  {
static void Main ()   {
int  t,   i;
int[,]   table  = new int[3,   4];
for(t=0;   t <  3;   ++t)    {
for(i=0;   i  < 4;   ++i) {
table[t,i] = (t*4)+i+1;
Console.Write(table[t,i]   +  "  ");}
Console.WriteLine();
} } }
```



## *Массивы трех и более измерений*

Тип [,...,] *имя_массива = new тип [размер1, размер2, ... размерN]:*

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

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

```
// Суммировать  значения по одной из диагоналей матрицы 3x3x3.
using System;
class ThreeCMatrix   {
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[1,1,1] + m[2, 2, 2];
Console.WriteLine("Сумма значений по первой диагонали:    "   + sum);
}}
```

## *Инициализация многомерных массивов*

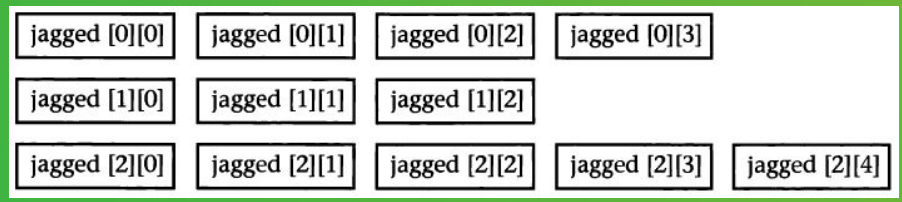
*тип[,...,] имя_массива* = {
{val, val, val, .... val},
{val, val, val, .... val},

{val, val, val, .... val}
};

где *val* обозначает инициализирующее значение, а каждый внутренний блок — отдельный ряд

Ступенчатый массив представляет собой *массив массивов,* в ко тором длина каждого массива может быть разной.

```
int[][] jagged = new int[3][];
jagged[0] = new int[4];
jagged[1] = new int[3];
jagged[2] = new int[5];
```




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

```
using System;
class Jagged {
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;
// Сохранить значения в первом массиве.
for(i=0; i < 4; i++)
jagged[0][i] = i;
// Сохранить значения во втором массиве.
for(i=0; i < 3; i++)
jagged[1][i] = i;
// Сохранить значения в третьем массиве.
```

Присваивание значения одной переменной ссылки на массив другой переменной, по существу, означает, что обе переменные ссылаются на один и тот же массив

```
// Присваивание ссылок на массивы.
using System;
class AssignARef {
static void Main() {
int i;
int[] numsl = new int [10];
int[] nums2 = new int[10];
for(i=0; i < 10; i++) numsl[i] = i;
for(i=0; i < 10; i++) nums2[i] = -i;
Console.Write("Содержимое массива numsl: ");
for(i=0; i < 10; i++)
Console.Write(numsl[i] + " ");
Console.WriteLine() ;
Console.Write("Содержимое массива nums2: ");
for(i=0; i < 10; i++)
```

свойство Length определяет длину массива

```
// Использовать свойство Length массива.
using System;
class LengthDemo {
static void Main() {
int[] nums = new int[10];
Console.WriteLine("Длина массива nums равна " + nums.Length);
// Использовать свойство Length для инициализации массива nums.
for(int i=0; i < nums.Length; i++)
nums[i] = i * i;
// А теперь воспользоваться свойством Length
// для вывода содержимого массива nums.
Console.Write("Содержимое массива nums: ");
for(int i=0; i < nums.Length; i++)
Console.Write(nums[i] + " ");
Console.WriteLine();
} }
```

Когда запрашивается длина многомерного массива, то возвращается общее число элементов, из которых может состоять массив

```
// Использовать свойство Length трехмерного массива.
using System;
class LengthDemo3D {
static void Main() {
int[,,] nums = new int[10, 5, 6];
Console.WriteLine("Длина массива nums равна " + nums.Length) ; } }
```

с помощью свойства Length можно получить длину каждого массива, составлявшего ступенчатый массив

```
// Продемонстрировать применение свойства Length  при обращении со ступенчатыми массивами.
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++)
```

*Неявно типизированный массив* объявляется с помощью ключевого слова var, но без последующих квадратных скобок [ ]. Кроме того, неявно типизированный массив должен быть непременно инициализирован, поскольку по типу инициализаторов определяется тип элементов данного массива. Все инициализаторы должны быть одного и того же согласованного типа.

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

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

```
// Продемонстрировать неявно типизированный ступенчатый массив.
using System;
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();
} } }
```

# *Оператор цикла foreach*

Оператор foreach служит для циклического обращения к элементам коллекции. В C# определено несколько видов коллекций, каждая из которых является массивом.

foreach {тип имя_переменной_цикла in коллекция) оператор;

*тип имя_переменной_цикла* обозначает тип и имя переменной управления циклом, которая получает значение следующего элемента коллекции на каждом шаге выполнения цикла foreach. А коллекция обозначает циклически опрашиваемый массив.

Тип переменной цикла должен соответствовать типу элемента массива.

Переменная цикла в операторе foreach служит только для чтения. Это означает, что, присваивая этой переменной новое значение, нельзя изменить содержимое массива

```
// Использовать оператор цикла foreach.
using System;
class ForeachDemo {
static void Main() {
int sum = 0;
int[] nums = new int [10];
// Задать первоначальные значения элементов массива nums.
for(int i = 0; i < 10; i++)
nums[i] = i;
// Использовать цикл foreach для вывода значений элементов массива и подсчета их суммы,
foreach(int x in nums) {
Console.WriteLine("Значение элемента равно: " + x);
sum += x;
}
Console.WriteLine("Сумма равна: " + sum); } }
```

цикл foreach повторяется до тех пор, пока не будут опрошены все элементы массива, его можно завершить преждевременно, воспользовавшись оператором break

```
// Использовать оператор break для преждевременного завершения цикла foreach.
using System;
class ForeachDemo {
static void Main() {
int sum = 0;
int[] nums = new int [10];
// Задать первоначальные значения элементов массива nums.
for(int i = 0; i < 10; i++)
nums[i] = i;
// Использовать цикл foreach для вывода значений
// элементов массива и подсчета их суммы.
foreach(int x in nums) {
Console.WriteLine("Значение элемента равно: " + x) ;
```

Оператор цикла foreach можно также использовать для циклического обращения к элементам многомерного массива. В этом случае элементы многомерного массива возвращаются по порядку следования строк от первой до последней

```
// Использовать оператор цикла foreach для обращения к двумерному массиву.
using System;
class ForeachDemo2 {
static void Main() {
int sum = 0;
int[,] nums = new int"[3,5];

// Задать первоначальные значения элементов массива nums.
for (int i = 0; i < 3; i++)
for (int j=0; j < 5; j++)
nums[i,j] = (i+1)* (j + 1);
```

```
// Поиск в массиве с помощью оператора цикла foreach.
using System;
class Search {
static void Main() {
int[] nums = new int[10];
int val;
bool found = false;
// Задать первоначальные значения элементов массива nums.
for (int i = 0; i < 10; i++)
nums[i] = i;
val = 5;
// Использовать цикл foreach для поиска заданного
// значения в массиве nums.
foreach(int x in nums) {
if(x == val) {
found = true;
break;
} }
if(found)
Console.WriteLine("Значение найдено!");
} }
```