*Массив* представляет собой совокупность переменных одного типа с общим для обращения к ним именем. В C# массивы могут быть как одномерными, так и многомерными, хотя чаще всего применяются одномерные массивы.

Главное преимущество массива — в организации данных таким образом, чтобы ими было проще манипулировать.

Массивами в C# можно пользоваться практически так же, как и в других языках программирования. Тем не менее у них имеется одна особенность: они реализованы в виде

объектов.

**Одномерные массивы**

*Одномерный массив* представляет собой список связанных переменных. Такие списки

часто применяются в программировании.

Для того чтобы воспользоваться массивом в программе, требуется двухэтапная процедура,

поскольку в C# массивы реализованы в виде объектов. Во-первых, необходимо

объявить переменную, которая может обращаться к массиву. И во-вторых, нужно создать

экземпляр массива, используя оператор ***new***. Так, для объявления одномерного

массива обычно применяется следующая общая форма:

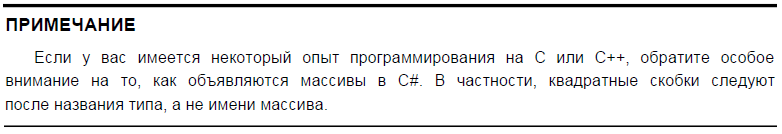
***тип[] имя\_массива = new тип[размер];***

где *тип* объявляет конкретный тип элемента массива. Тип элемента определяет тип

данных каждого элемента, составляющего массив. Обратите внимание на квадратные

скобки, которые сопровождают *тип.* Они указывают на то, что объявляется одномерный

массив. А *размер* определяет число элементов массива.



Обратимся к конкретному примеру. В приведенной ниже строке кода создается

массив типа int, который составляется из десяти элементов и связывается с переменной

ссылки на массив, именуемой sample.

int[] sample = new int[10];

Как и при создании экземпляра класса, приведенное выше объявление массива можно разделить на два отдельных оператора. Например:

int[] sample;

sample = new int[10];

В данном случае переменная sample не ссылается на какой-то определенный физический

объект, когда она создается в первом операторе. И лишь после выполнения второго оператора эта переменная ссылается на массив.

Доступ к отдельному элементу массива осуществляется по индексу: *Индекс* обозначает

положение элемента в массиве. В языке C# индекс первого элемента всех массивов

оказывается нулевым. В частности, массив sample состоит из 10 элементов с индексами

от 0 до 9. Для индексирования массива достаточно указать номер требуемого элемента

в квадратных скобках. Так, первый элемент массива sample обозначается как

sample [0], а последний его элемент — как sample[9]. (**glava7\_1\_array, glava7\_2**)

class ArrayDemo

{

static void Main()

{

int[] sample = new int[10];

int i;

for (i = 0; i < 10; i += 1)

sample[i] = i;

for (i = 0; i < 10; i += 1)

Console.WriteLine("sample [" + i + "]: " + sample[i]);

}

}

**Инициализация массива**

Mассивы могут инициализироваться, когда они создаются. Ниже приведена общая форма инициализации одномерного массива:

**тип[] имя\_массива = {vall, val2, val3, ..., valN};**

где *val1-valN* обозначают первоначальные значения, которые присваиваются по очереди,

слева направо и по порядку индексирования. Для хранения инициализаторов

массива в C# автоматически распределяется достаточный объем памяти. А необходимость

пользоваться оператором new явным образом отпадает сама собой.

(glava7\_2.1)

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 += 1)

avg += nums[i];

avg /= 10;

Console.WriteLine("Average: " + avg);

}

}

Любопытно, что при инициализации массива можно также воспользоваться оператором

new, хотя особой надобности в этом нет. Например, приведенный ниже фрагмент

кода считается верным, но избыточным для инициализации массива nums в упомянутой

выше программе.

int [] nums = new int [] { 99,10,100,18,78,23,

63,9,87,49};

Несмотря на свою избыточность, форма инициализации массива с оператором new

оказывается полезной в том случае, если новый массив присваивается уже существующей

переменной ссылки на массив. Например:

int[] nums;

nums = new int[] { 99,10,100,18,78,23,

63,9,87,49};

И последнее замечание: 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. В этом объявлении размер массива nums задается равным 10 явно.

int[] nums = new int[10]{ 99,10,100,18,78,23,

63,9,87,49};

**Соблюдение границ массива**

Границы массива в C# строго соблюдаются. Если границы массива не достигаются

или же превышаются, то возникает ошибка при выполнении.

**Многомерные массивы**

В программировании чаще всего применяются одномерные массивы, хотя и многомерные

не так уж и редки. *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.

**Двумерные массивы**

Простейшей формой многомерного массива является двумерный массив. Местоположение

любого элемента в двумерном массиве обозначается двумя индексами.

В следующей строке кода объявляется двумерный массив integer размерами 10×20.

int [,] table = new int [10, 20];

Обратите особое внимание на объявление этого массива. Как видите, оба его размера

разделяются запятой. В первой части этого объявления синтаксическое обозначение

***[,]***

означает, что создается переменная ссылки на двумерный массив. Если же память распределяется

для массива с помощью оператора new, то используется следующее синтаксическое

обозначение.

***int[10, 20]***

Для доступа к элементу двумерного массива следует указать оба индекса, разделив

их запятой. Например, в следующей строке кода элементу массива table с координатами

местоположения (3,5) присваивается значение 10.

table[3, 5] = 10;

Пример в листинге ниже.

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();

}

}

}

В данном примере элемент массива table[0,0] будет иметь значение 1, элемент

массива table [0,1] — значение 2, элемент массива table [0,2] — значение 3 и т.д.

А значение элемента массива table [2,3] окажется равным 12. На рис. 7.1 показано

схематически расположение элементов этого массива и их значений.



**Массивы трех и более измерений**

В C# допускаются массивы трех и более измерений. Ниже приведена общая форма

объявления многомерного массива.

**тип[,...,] имя\_массива = new тип [размер1, размер2, ... размерN];**

Например, в приведенном ниже объявлении создается трехмерный целочисленный

массив размерами 4×10×3.

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

А в следующем операторе элементу массива multidim с координатами местоположения

(2,4,1) присваивается значение 100.

multidim [2, 3, 1] = 100;

В листинге glava7\_4\_multiArr пример с трех мерным массивом.

class ThreeDMatrix

{

static void Main()

{

int sum = 0, n = 1;

int[,,] m = new int [3, 3, 3];

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("Sum of first diagonal: " + sum);

}

}

**Инициализация многомерных массивов**

Для инициализации многомерного массива достаточно заключить в фигурные

скобки список инициализаторов каждого его размера. Ниже в качестве примера приведена

общая форма инициализации двумерного массива:

**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.

В качестве примера ниже приведена программа, в которой двумерный массив sqrs

инициализируется числами от 1 до 10 и квадратами этих чисел.

class Squares

{

static void Main()

{

int[,] sqrs =

{

{ 1,2 },

{ 2, 4 },

{ 3, 9 },

{ 4, 16 },

{ 5, 25 },

{ 6, 36 },

{ 7, 49 },

{ 8, 64 },

{ 9, 81 },

{ 10, 100 }

};

int i, j;

for (i = 0; i < 10; i++)

{

for (j = 0; j < 2; j++)

Console.Write(sqrs[i,j] + " ");

Console.WriteLine();

}

}

}

**Ступенчатые массивы**

В приведенных выше примерах применения двумерного массива, по существу, создавался

так называемый *прямоугольный массив.* Двумерный массив можно представить

в виде таблицы, в которой длина каждой строки остается неизменной по всему массиву.

Но в C# можно также создавать специальный тип двумерного массива, называемый

*ступенчатым массивом.* Ступенчатый массив представляет собой *массив массивов,* в котором

длина каждого массива может быть разной. Следовательно, ступенчатый массив

может быть использован для составления таблицы из строк разной длины.

Например, для объявления двумерного ступенчатого массива

служит следующая общая форма:

**тип [] [] имя\_массива = new тип[размер] [];**

где ***размер***обозначает число строк в массиве.

int [] [] jagged = new int [3] [];

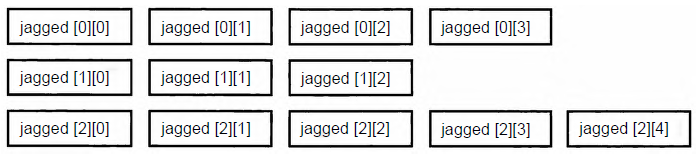
jagged [0] = new int [4];

jagged [1] = new int [3];

jagged [2] = new int [5];

После выполнения этого фрагмента кода массив ***jagged*** выглядит так, как показано

ниже.



После создания ступенчатого массива доступ к его элементам осуществляется по индексу,

указываемому в отдельных квадратных скобках. Например, в следующей строке кода

элементу массива jagged, находящемуся на позиции с координатами (2,1), присваивается

значение 10.

jagged [2][1] = 10;

Обратите внимание на синтаксические отличия в доступе к элементу ступенчатого и прямоугольного массива.

Пример в листинге ***glava7\_6\_ladArr***.

И последнее замечание: ступенчатые массивы представляют собой массивы массивов,

и поэтому они не обязательно должны состоять из одномерных массивов. Например,

в приведенной ниже строке кода создается массив двумерных массивов.

***int [] [,] jagged = new int [3] [,];***

В следующей строке кода элементу массива jagged [0] присваивается ссылка на

массив размерами 4×2.

***jagged [0] = new int [4, 2];***

А в приведенной ниже строке кода элементу массива jagged [0] [1,0] присваивается

значение переменной i.

***jagged [0] [1,0] = i;***

**Присваивание ссылок на массивы**

Присваивание значения одной переменной ссылки на массив другой переменной,

по существу, означает, что обе переменные ссылаются на один и тот же массив,

и в этом отношении массивы ничем не отличаются от любых других объектов. Такое

присваивание не приводит ни к созданию копии массива, ни к копированию содержимого

одного массива в другой. (***glava7\_7***)

Короче говоря, все просто, похоже на С++.

int [] nums1 = new int [10];

int [] nums2 = new int [10];

nums2 = nums1;//nums2 теперь указывает на массив nums1

Теперь nums2 может влиять на массив nums1, изменять значения и тд.

**Применение свойства Length**

Реализация в C# массивов в виде объектов дает целый ряд преимуществ. Одно из

них заключается в том, что с каждым массивом связано свойство Length, содержащее

число элементов, из которых может состоять массив. Следовательно, у каждого массива

имеется специальное свойство, позволяющее определить его длину. (***glava7\_8***)

class LengthDemo

{

static void Main()

{

int [] nums = new int [10];

Console.WriteLine("Length of nums array = " + nums.Length);

//using Length for initialize nums array

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

nums[i] = i \* i;

//Also for output

Console.Write("Nums Array: ");

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

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

}

}

***Length*** используется без скобочек, как будто функтор.

Когда запрашивается длина многомерного массива, то возвращается общее число

элементов, из которых может состоять массив, как в приведенном ниже примере кода.

class LengthDemo3D

{

static void Main()

{

int[,,] nums = new int[10, 5, 6];

Console.WriteLine("Длина массива nums равна " + nums.Length);

}

При выполнении этого кода получается следующий результат.

***Длина массива nums равна 300***

Как подтверждает приведенный выше результат, свойство Length содержит число

элементов, из которых может состоять массив (в данном случае — 300 (10×5×6) элементов).

Тем не менее свойство Length нельзя использовать для определения длины

массива в отдельном его измерении.

**Неявно типизированные массивы**

Как правило, неявно типизированные массивы предназначены

для применения в определенного рода вызовах, включающих в себя элементы

языка LINQ, о котором речь пойдет в главе 19. А в большинстве остальных случаев

используется "обычное" объявление массивов.

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. Ниже приведен пример объявления неявно типизированного

массива.

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

В данном примере создается массив типа int, состоящий из пяти элементов. Ссылка

на этот массив присваивается переменной vals. Следовательно, тип этой переменной

соответствует типу int массива, состоящего из пяти элементов. Обратите внимание

на то, что в левой части приведенного выше выражения отсутствуют квадратные

скобки []. А в правой части этого выражения, где происходит инициализация массива,

квадратные скобки присутствуют. В данном контексте они обязательны.

Рассмотрим еще один пример, в котором создается двумерный массив типа

double.

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

В данном случае получается массив vals размерами 2×3.

Объявлять можно также неявно типизированные ступенчатые массивы.

***var jagged = new [] {***

***new [] {1, 2, 3, 4},***

***new [] {9, 8, 7),***

***new [] {11, 12, 13, 14, 15}***

***};***

Как упоминалось выше, неявно типизированные массивы чаще всего применяются

в LINQ-ориентированных запросах. А в остальных случаях следует использовать явно

типизированные массивы.