Массивы являются коллекциями объектов одного типа. Поскольку длина массивов практически не ограничена, они могут использоваться для хранения тысяч или даже миллионов объектов, но размер массива должен быть указан при его создании. Каждый элемент массива доступен по числовому индексу, указывающему позицию или ячейку, в которой объект хранится в массиве. Массивы могут хранить как [ссылочные типы](http://msdn.microsoft.com/ru-ru/library/490f96s2%28v%3Dvs.90%29.aspx), так и [типы значений](http://msdn.microsoft.com/ru-ru/library/s1ax56ch%28v%3Dvs.90%29.aspx).

### Одномерные массивы

Массив является индексированной коллекцией объектов. Одномерный массив объектов объявляется следующим образом.

type[] arrayName;

Часто элементы в массиве инициализируются в это же время, как показано ниже.

C#

int[] array = new int[5];

Значение по умолчанию числовых элементов массива задано равным нулю, а элементы ссылок имеют значение **null**, но значения можно инициализировать при создании массива следующим образом.

C#

int[] array1 = new int[] { 1, 3, 5, 7, 9 };

Или так.

C#

int[] array2 = {1, 3, 5, 7, 9};

Индексация массивов начинается с нуля, поэтому номер первого элемента массива равен 0.

C#

string[] days = {"Sun", "Mon", "Tue", "Wed", "Thr", "Fri", "Sat"};

System.Console.WriteLine(days[0]); // Outputs "Sun"

### Многомерные массивы

Концептуально, многомерный массив с двумя измерениями напоминает сетку. Многомерный массив с тремя измерениями напоминает куб.

C#

// declare multidimension array (two dimensions)

int[,] array2D = new int[2,3];

// declare and initialize multidimension array

int[,] array2D2 = { {1, 2, 3}, {4, 5, 6} };

// write elements in a multidimensional array

for (int i=0; i<2; i++)

{

 for (int j=0; j<3; j++)

 {

 array2D[i,j] = (i + 1) \* (j + 1);

 }

}

// read elements in a multidimensional array

for (int i=0; i<2; i++)

{

 for (int j=0; j<3; j++)

 {

 System.Console.Write(array2D[i,j]);

 }

 System.Console.WriteLine();

}

### Массивы массивов

Одним из вариантов многомерного массива является массив массивов. Массив массивов представляет собой одномерный массив, в котором каждый элемент является массивом. Элементы массива не обязаны иметь одинаковый размер.

Объявить массив массивов можно следующим образом

C#

int[][] jaggedArray = new int[3][];

Создание массива трех массивов. Эти массивы можно инициализировать следующим образом.

C#

jaggedArray[0] = new int[5];

jaggedArray[1] = new int[4];

jaggedArray[2] = new int[2];

### Использование оператора foreach

Оператор [foreach](http://msdn.microsoft.com/ru-ru/library/ttw7t8t6%28v%3Dvs.90%29.aspx) часто используется для доступа к каждому элементу, хранимому в массиве.

C#

int[] numbers = { 4, 5, 6, 1, 2, 3, -2, -1, 0 };

foreach (int i in numbers)

{

 System.Console.Write("{0} ", i);

}

//Output: 4 5 6 1 2 3 -2 -1 0

### Массивы объектов

Создание массива объектов в отличие от создания массива простых типов данных, например целочисленных, происходит в два этапа. Сначала необходимо объявить массив, а затем создать объекты для хранения в нем. В этом примере создается класс, определяющий аудио компакт-диск. Затем создается массив для хранения 20 аудио компакт-дисков.

C#

namespace CDCollection

{

 // Define a CD type.

 class CD

 {

 private string album;

 private string artist;

 private int rating;

 public string Album

 {

 get {return album;}

 set {album = value;}

 }

 public string Artist

 {

 get {return artist;}

 set {artist = value;}

 }

 public int Rating

 {

 get {return rating;}

 set {rating = value;}

 }

 }

 class Program

 {

 static void Main(string[] args)

 {

 // Create the array to store the CDs.

 CD[] cdLibrary = new CD[20];

 // Populate the CD library with CD objects.

 for (int i=0; i<20; i++)

 {

 cdLibrary[i] = new CD();

 }

 // Assign details to the first album.

 cdLibrary[0].Album = "See";

 cdLibrary[0].Artist = "The Sharp Band";

 cdLibrary[0].Rating = 10;

 }

 }

}

Коллекции

Массив является одним из многочисленных вариантов хранения набора данных, используемых C#. Вариант выбора зависит от нескольких факторов, например от планируемого способа управления или доступа к элементам. Например, *список* работает, как правило, быстрее массива при добавлении элемента в начало или в середину коллекции. Другие типы классов коллекций, такие как хэш-таблица, связный список и стек, имеют свои преимущества. Дополнительные сведения см. в разделах [System.Collections](http://msdn.microsoft.com/ru-ru/library/system.collections%28v%3Dvs.90%29.aspx) и [System.Collections.Generic](http://msdn.microsoft.com/ru-ru/library/system.collections.generic%28v%3Dvs.90%29.aspx).

В следующем примере показано использование класса [List<T>](http://msdn.microsoft.com/ru-ru/library/6sh2ey19%28v%3Dvs.90%29.aspx). Обратите внимание, что в отличие от класса [Array](http://msdn.microsoft.com/ru-ru/library/system.array%28v%3Dvs.90%29.aspx), элементы могут вставляться в середину списка. В этом примере показано ограничение списка, который может содержать только текстовые элементы.

C#

public class TestCollections

{

 public static void TestList()

 {

 System.Collections.Generic.List<string> sandwich = new System.Collections.Generic.List<string>();

 sandwich.Add("bacon");

 sandwich.Add("tomato");

 sandwich.Insert(1, "lettuce");

 foreach (string ingredient in sandwich)

 {

 System.Console.WriteLine(ingredient);

 }

 }

}