**Arrays (C# Programming Guide)**

* 5 minutes to read

You can store multiple variables of the same type in an array data structure. You declare an array by specifying the type of its elements. If you want the array to store elements of any type, you can specify object as its type. In the unified type system of C#, all types, predefined and user-defined, reference types and value types, inherit directly or indirectly from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object).

C#Copy

type[] arrayName;

**Example**

The following example creates single-dimensional, multidimensional, and jagged arrays:

C#Copy

class TestArraysClass

{

static void Main()

{

// Declare a single-dimensional array of 5 integers.

int[] array1 = new int[5];

// Declare and set array element values.

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

// Alternative syntax.

int[] array3 = { 1, 2, 3, 4, 5, 6 };

// Declare a two dimensional array.

int[,] multiDimensionalArray1 = new int[2, 3];

// Declare and set array element values.

int[,] multiDimensionalArray2 = { { 1, 2, 3 }, { 4, 5, 6 } };

// Declare a jagged array.

int[][] jaggedArray = new int[6][];

// Set the values of the first array in the jagged array structure.

jaggedArray[0] = new int[4] { 1, 2, 3, 4 };

}

}

**Array overview**

An array has the following properties:

* An array can be [single-dimensional](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/single-dimensional-arrays), [multidimensional](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/multidimensional-arrays) or [jagged](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/jagged-arrays).
* The number of dimensions and the length of each dimension are established when the array instance is created. These values can't be changed during the lifetime of the instance.
* The default values of numeric array elements are set to zero, and reference elements are set to null.
* A jagged array is an array of arrays, and therefore its elements are reference types and are initialized to null.
* Arrays are zero indexed: an array with n elements is indexed from 0 to n-1.
* Array elements can be of any type, including an array type.
* Array types are [reference types](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/reference-types) derived from the abstract base type [Array](https://docs.microsoft.com/en-us/dotnet/api/system.array). All arrays implement [IList](https://docs.microsoft.com/en-us/dotnet/api/system.collections.ilist), and [IEnumerable](https://docs.microsoft.com/en-us/dotnet/api/system.collections.ienumerable). You can use [foreach](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/statements/iteration-statements#the-foreach-statement) iteration arrays in C# . Since single-dimension arrays also implement [IList<T>](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.ilist-1), and [IEnumerable<T>](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.ienumerable-1).

**Arrays as Objects**

In C#, arrays are actually objects, and not just addressable regions of contiguous memory as in C and C++. [Array](https://docs.microsoft.com/en-us/dotnet/api/system.array) is the abstract base type of all array types. You can use the properties and other class members that [Array](https://docs.microsoft.com/en-us/dotnet/api/system.array) has. An example of this is using the [Length](https://docs.microsoft.com/en-us/dotnet/api/system.array.length) property to get the length of an array. The following code assigns the length of the numbers array, which is 5, to a variable called lengthOfNumbers:

C#Copy

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

int lengthOfNumbers = numbers.Length;

The [Array](https://docs.microsoft.com/en-us/dotnet/api/system.array) class provides many other useful methods and properties for sorting, searching, and copying arrays. The following example uses the [Rank](https://docs.microsoft.com/en-us/dotnet/api/system.array.rank) property to display the number of dimensions of an array.

C#Copy

class TestArraysClass

{

static void Main()

{

// Declare and initialize an array.

int[,] theArray = new int[5, 10];

System.Console.WriteLine("The array has {0} dimensions.", theArray.Rank);

}

}

// Output: The array has 2 dimensions.

**See also**

* [How to use single-dimensional arrays](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/single-dimensional-arrays)
* [How to use multi-dimensional arrays](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/multidimensional-arrays)
* [How to use jagged arrays](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/jagged-arrays)
* [Using foreach with arrays](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/using-foreach-with-arrays)
* [Passing arrays as arguments](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/passing-arrays-as-arguments)
* [Implicitly typed arrays](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/implicitly-typed-arrays)
* [C# Programming Guide](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/)
* [Collections](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/collections)

For more information, see the [C# Language Specification](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction). The language specification is the definitive source for C# syntax and usage.

**Recommended content**

**[Multidimensional Arrays - C# Programming Guide](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/multidimensional-arrays)**

Arrays in C# can have more than one dimension. This example declaration creates a two-dimensional array of four rows and two columns.

**[Math.Pow(Double, Double) Method (System)](https://docs.microsoft.com/en-us/dotnet/api/system.math.pow)**

Returns a specified number raised to the specified power.

**[Using foreach with arrays - C# Programming Guide](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/using-foreach-with-arrays)**

The foreach statement in C# iterates through the elements of an array. For single-dimensional arrays, foreach processes elements in increasing index order.

**[Array.Length Property (System)](https://docs.microsoft.com/en-us/dotnet/api/system.array.length)**

Gets the total number of elements in all the dimensions of the Array.

**[List<T>.Count Property (System.Collections.Generic)](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1.count)**

Gets the number of elements contained in the List&lt;T&gt;.

**[return statement - C# Reference](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/return)**

return statement - C# Reference

**[Passing arrays as arguments - C# Programming Guide](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/passing-arrays-as-arguments)**

Arrays in C# can be passed as arguments to method parameters. Because arrays are reference types, the method can change the value of the elements.

**[for statement - C# reference](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/for)**

for statement - C# reference