

# C# Array class

C# provides an Array class to deal with array related operations. It provides methods for creating, manipulating, searching, and sorting elements of an array. This class works as the base class for all arrays in the .NET programming environment.

## C# Array class Signature

```
[SerializableAttribute]
[ComVisibleAttribute(true)]
public abstract class Array : ICloneable, IList, ICollection,
IEnumerable, IStructuralComparable, IStructuralEquatable
```

Note: In C#, Array is not part of collection but considered as collection because it is based on the IList interface.

## C# Array Properties

Property	Description
IsFixedSize	It is used to get a value indicating whether the Array has a fixed size or not.
IsReadOnly	It is used to check that the Array is read-only or not.
IsSynchronized	It is used to check that access to the Array is synchronized or not.
Length	It is used to get the total number of elements in all the dimensions of the Array.
LongLength	It is used to get a 64-bit integer that represents the total number of elements in all the dimensions of the Array.
Rank	It is used to get the rank (number of dimensions) of the Array.
SyncRoot	It is used to get an object that can be used to synchronize access to the Array.

## C# Array Methods

Method	Description
--------	-------------

AsReadOnly<T>(T[])	It returns a read-only wrapper for the specified array.
BinarySearch(Array,Int32,Int32,Object)	It is used to search a range of elements in a one-dimensional sorted array for a value.
BinarySearch(Array,Object)	It is used to search an entire one-dimensional sorted array for a specific element.
Clear(Array,Int32,Int32)	It is used to set a range of elements in an array to the default value.
Clone()	It is used to create a shallow copy of the Array.
Copy(Array,Array,Int32)	It is used to copy elements of an array into another array by specifying starting index.
CopyTo(Array,Int32)	It copies all the elements of the current one-dimensional array to the specified one-dimensional array starting at the specified destination array index
CreateInstance(Type,Int32)	It is used to create a one-dimensional Array of the specified Type and length.
Empty<T>()	It is used to return an empty array.
Finalize()	It is used to free resources and perform cleanup operations.
Find<T>(T[],Predicate<T>)	It is used to search for an element that matches the conditions defined by the specified predicate.
IndexOf(Array,Object)	It is used to search for the specified object and returns the index of its first occurrence in a one-dimensional array.
Initialize()	It is used to initialize every element of the value-type Array by calling the default constructor of the value type.
Reverse(Array)	It is used to reverse the sequence of the elements in the entire one-dimensional Array.

Sort(Array)	It is used to sort the elements in an entire one-dimensional Array.
ToString()	It is used to return a string that represents the current object.

## C# Array Example

```
using System;
namespace CSharpProgram
{
    class Program
    {
        static void Main(string[] args)
        {
            // Creating an array
            int[] arr = new int[6] { 5, 8, 9, 25, 0, 7 };
            // Creating an empty array
            int[] arr2 = new int[6];
            // Displaying length of array
            Console.WriteLine("length of first array: "+arr.Length);
            // Sorting array
            Array.Sort(arr);
            Console.Write("First array elements: ");
            // Displaying sorted array
            PrintArray(arr);
            // Finding index of an array element
            Console.WriteLine("\nIndex position of 25 is "+Array.IndexOf(arr,25));
            // Coping first array to empty array
            Array.Copy(arr, arr2, arr.Length);
            Console.Write("Second array elements: ");
            // Displaying second array
            PrintArray(arr2);
            Array.Reverse(arr);
            Console.Write("\nFirst Array elements in reverse order: ");
            PrintArray(arr);
        }
        // User defined method for iterating array elements
        static void PrintArray(int[] arr)
        {
            foreach (Object elem in arr)
```

```
    {  
        Console.Write(elem+" ");  
    }  
}  
}
```

**Output:**

```
length of first array: 6  
First array elements: 0 5 7 8 9 25  
Index position of 25 is 5  
Second array elements: 0 5 7 8 9 25  
First Array elements in reverse order: 25 9 8 7 5 0
```