6. Arrays

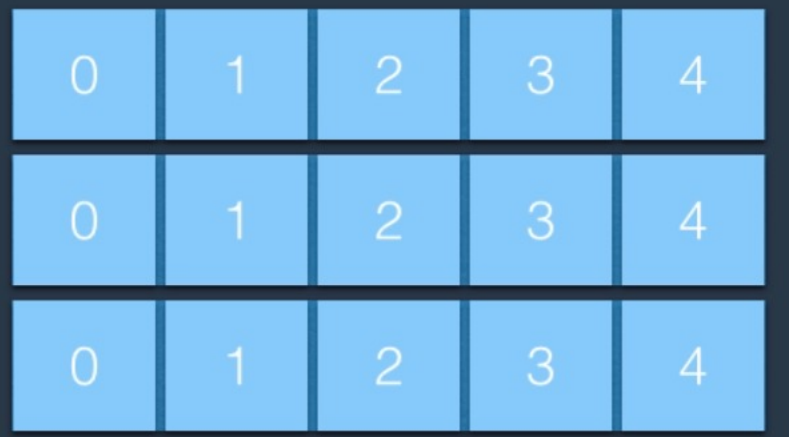
- Quick review of arrays
- Types of Arrays in C#
- Array methods

Array	Represents a fixed number of variables of a particular type.

Types of Arrays

Single Dimension

Multi Dimension



Single Dimension Arrays

```
var numbers = new int[5];

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

Multi Dimension Arrays

Syntax (Rectangular 2D)

```
var matrix = new int[3, 5];
```

```
var matrix = new int[3, 5]
{
    { 1, 2, 3, 4, 5 },
    { 6, 7, 8, 9, 10 },
    { 11, 12, 13, 14, 15 }
};
```

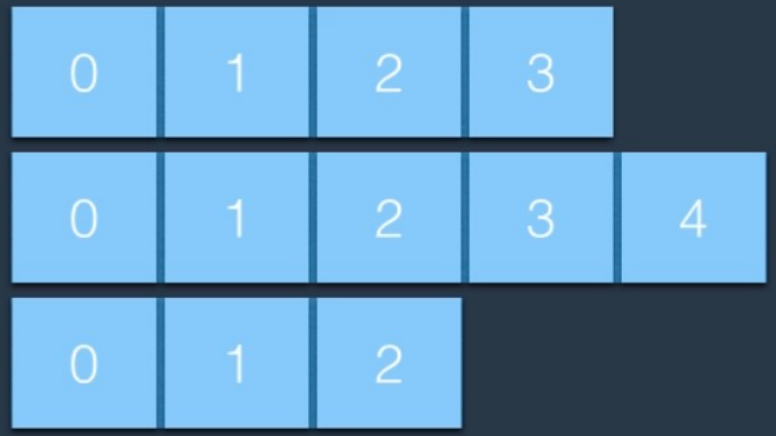
```
var element = matrix[0, 0];
```

Syntax (Rectangular 3D)

```
var colors = new int[3, 5, 4];
```

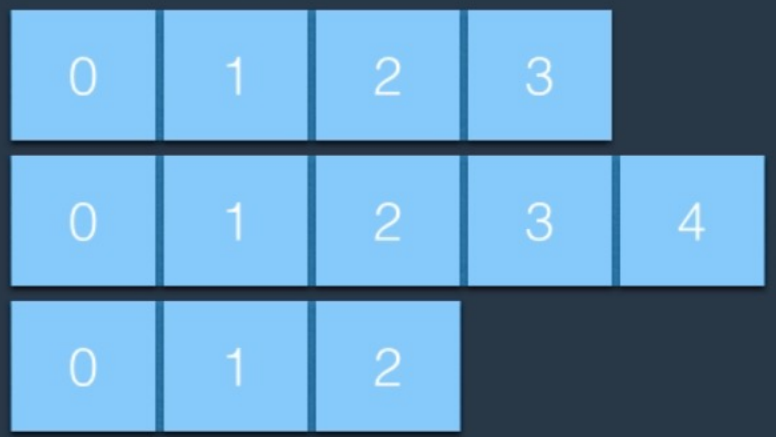
Syntax (Jagged)

Jagged



Syntax (Jagged)

Jagged



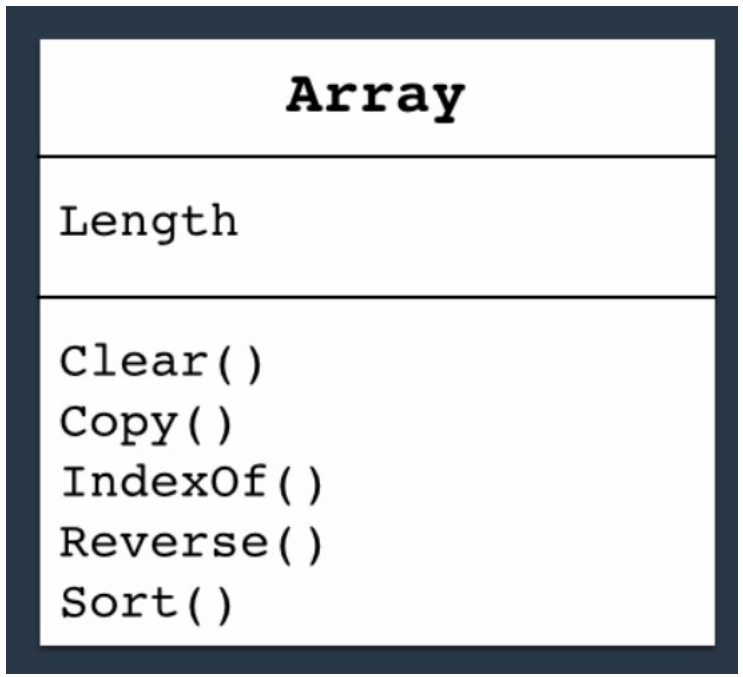

```
var array = new int[3][];


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


array[0][0] = 1;
```

Jagged

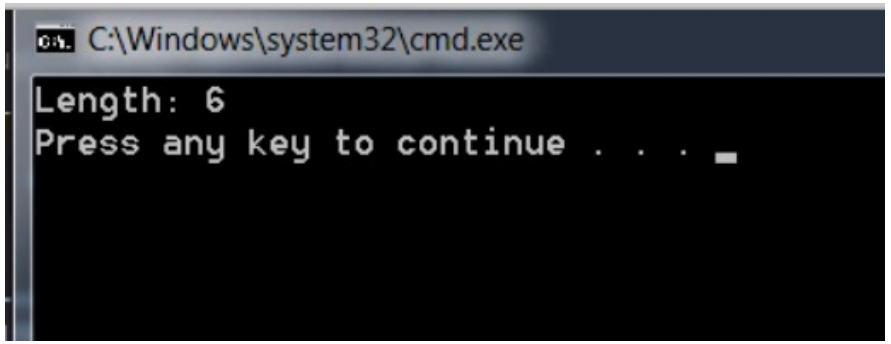
```
var array = new int[3][];
```

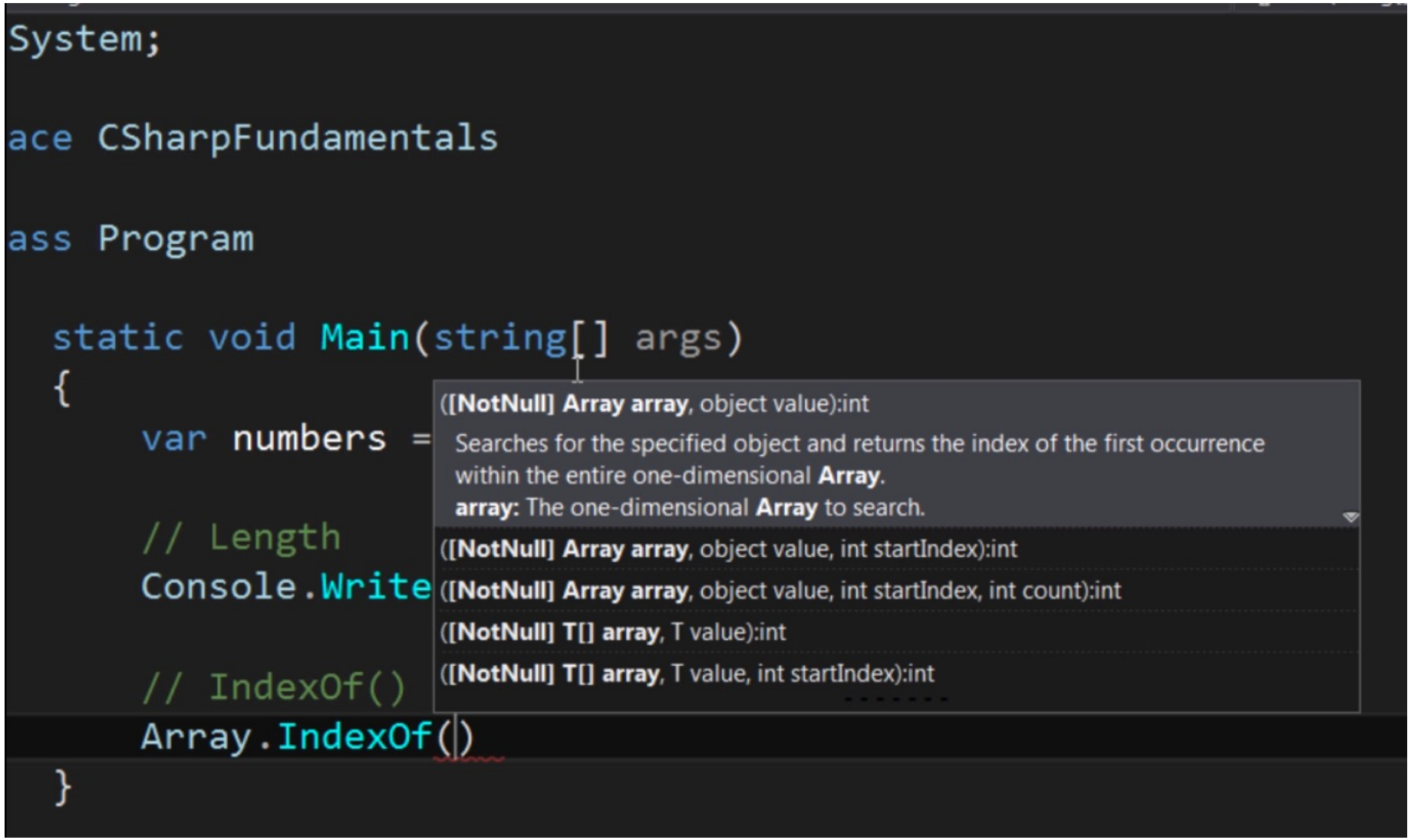
Rectangular

```
var array = new int[3, 5];
```


Array
Length
Clear()
Copy()
IndexOf()
Reverse()
Sort()

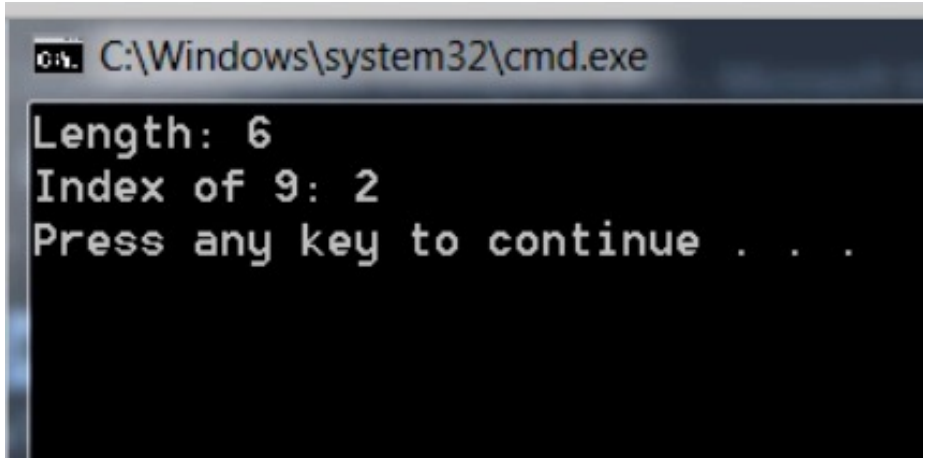
```
using System;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var numbers = new[] { 3, 7, 9, 2, 14, 6 };

            // Length
            Console.WriteLine("Length: " + numbers.Length);
        }
    }
}
```

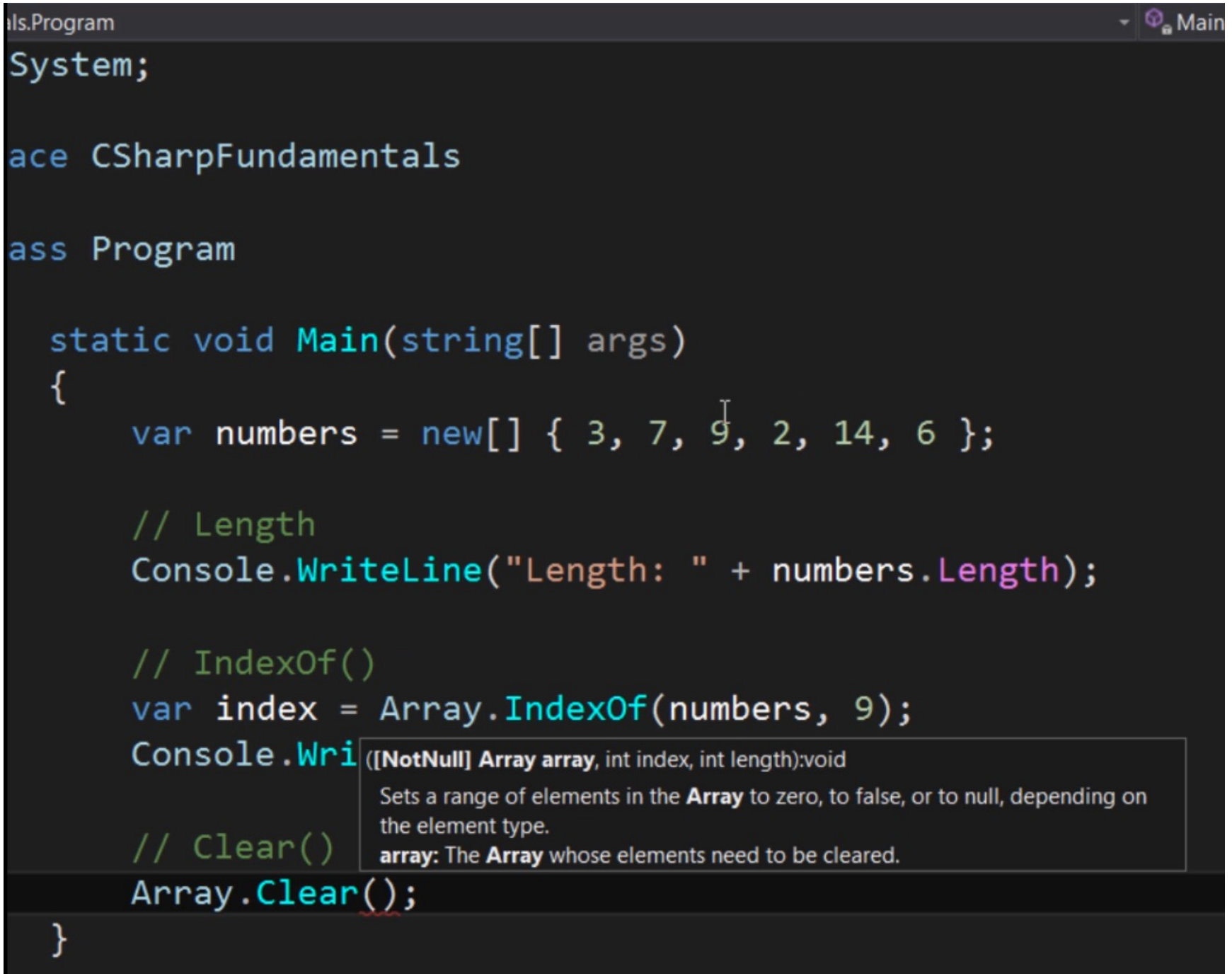



System;
ace CSharpFundamentals
ass Program
static void Main(string[] args)
{
var numbers =
// Length
Console.Write
// IndexOf()
Array.IndexOf()
}
([NotNull] Array array, object value):int
Searches for the specified object and returns the index of the first occurrence within the entire one-dimensional Array.
array: The one-dimensional Array to search.
([NotNull] Array array, object value, int startIndex):int
([NotNull] Array array, object value, int startIndex, int count):int
([NotNull] T[] array, T value):int
([NotNull] T[] array, T value, int startIndex):int

CSharpFundamentals.Program

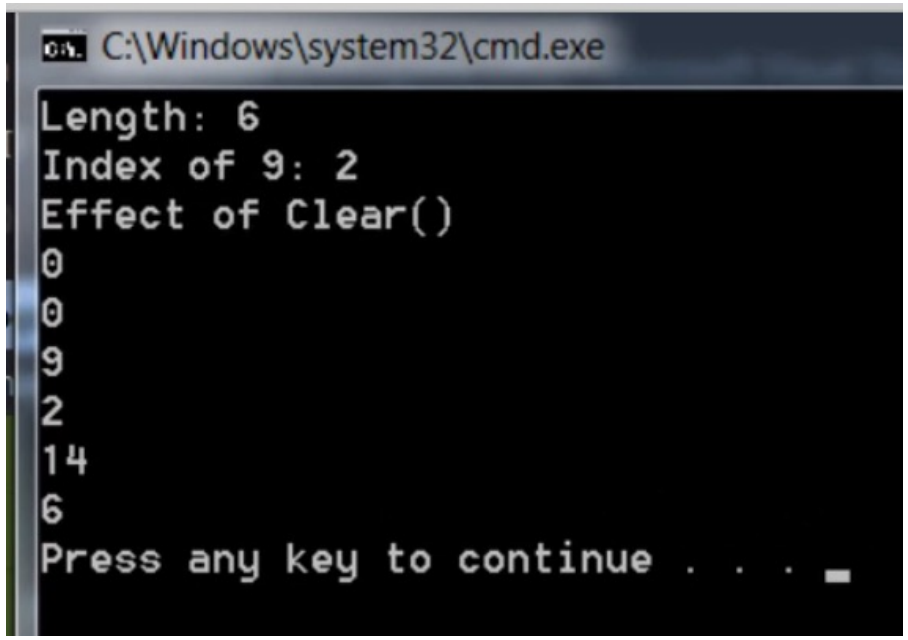
```
using System;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var numbers = new[] { 3, 7, 9, 2, 14, 6 };

            // Length
            Console.WriteLine("Length: " + numbers.Length);

            // IndexOf()
            var index = Array.IndexOf(numbers, 9);
            Console.WriteLine("Index of 9: " + index);
        }
    }
}
```




ls.Program
Main
System;
ace CSharpFundamentals
ass Program
static void Main(string[] args)
{
var numbers = new[] { 3, 7, 9, 2, 14, 6 };
// Length
Console.WriteLine("Length: " + numbers.Length);
// IndexOf()
var index = Array.IndexOf(numbers, 9);
Console.Wri
([NotNull] Array array, int index, int length):void
Sets a range of elements in the Array to zero, to false, or to null, depending on the element type.
array: The Array whose elements need to be cleared.
// Clear()
Array.Clear();
}

SharpFundamentals.Program

```
using System;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var numbers = new[] { 3, 7, 9, 2, 14, 6 };

            // Length
            Console.WriteLine("Length: " + numbers.Length);

            // IndexOf()
            var index = Array.IndexOf(numbers, 9);
            Console.WriteLine("Index of 9: " + index);

            // Clear()
            Array.Clear(numbers, 0, 2);

            Console.WriteLine("Effect of Clear()");
            foreach (var n in numbers)
                Console.WriteLine(n);
        }
    }
}
```



```
ace CSharpFundamentals

ass Program

  static void Main(string[] args)
  {
      var numbers = new[] { 3, 7, 9, 2, 14, 6 };

      // Length
      Console.WriteLine("Length: " + numbers.Length);

      // IndexOf()
      var index = Array.IndexOf(numbers, 9);
      Console.WriteLine("Index of 9: " + index);

      // Clear()
      Array.Clea

      Console.Wr
      foreach (v
          Consol

      // Copy()
      Array.Copy();
  }
```

([NotNull] Array sourceArray, [NotNull] Array destinationArray, int length):void

Copies a range of elements from an **Array** starting at the first element and pastes them into another **Array** starting at the first element. The length is specified as a 32-bit integer.

sourceArray: The **Array** that contains the data to copy.

([NotNull] Array sourceArray, [NotNull] Array destinationArray, long length):void

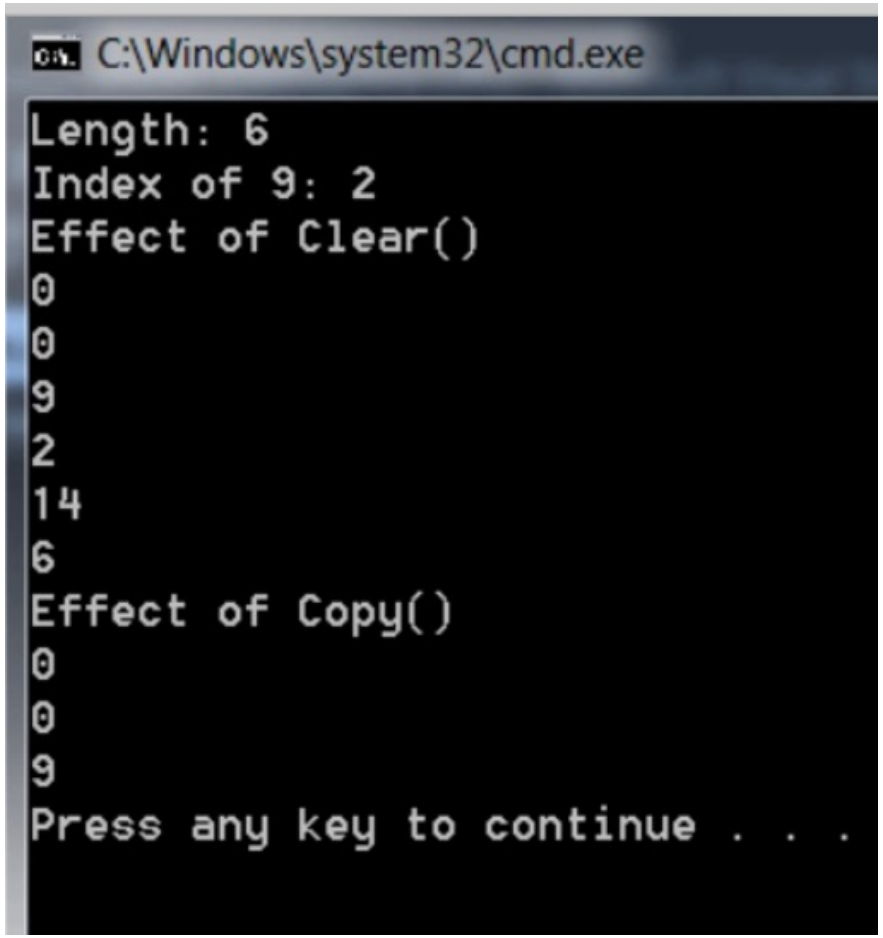
([NotNull] Array sourceArray, int sourceIndex, [NotNull] Array destinationArray, int destinationIndex, int length):void

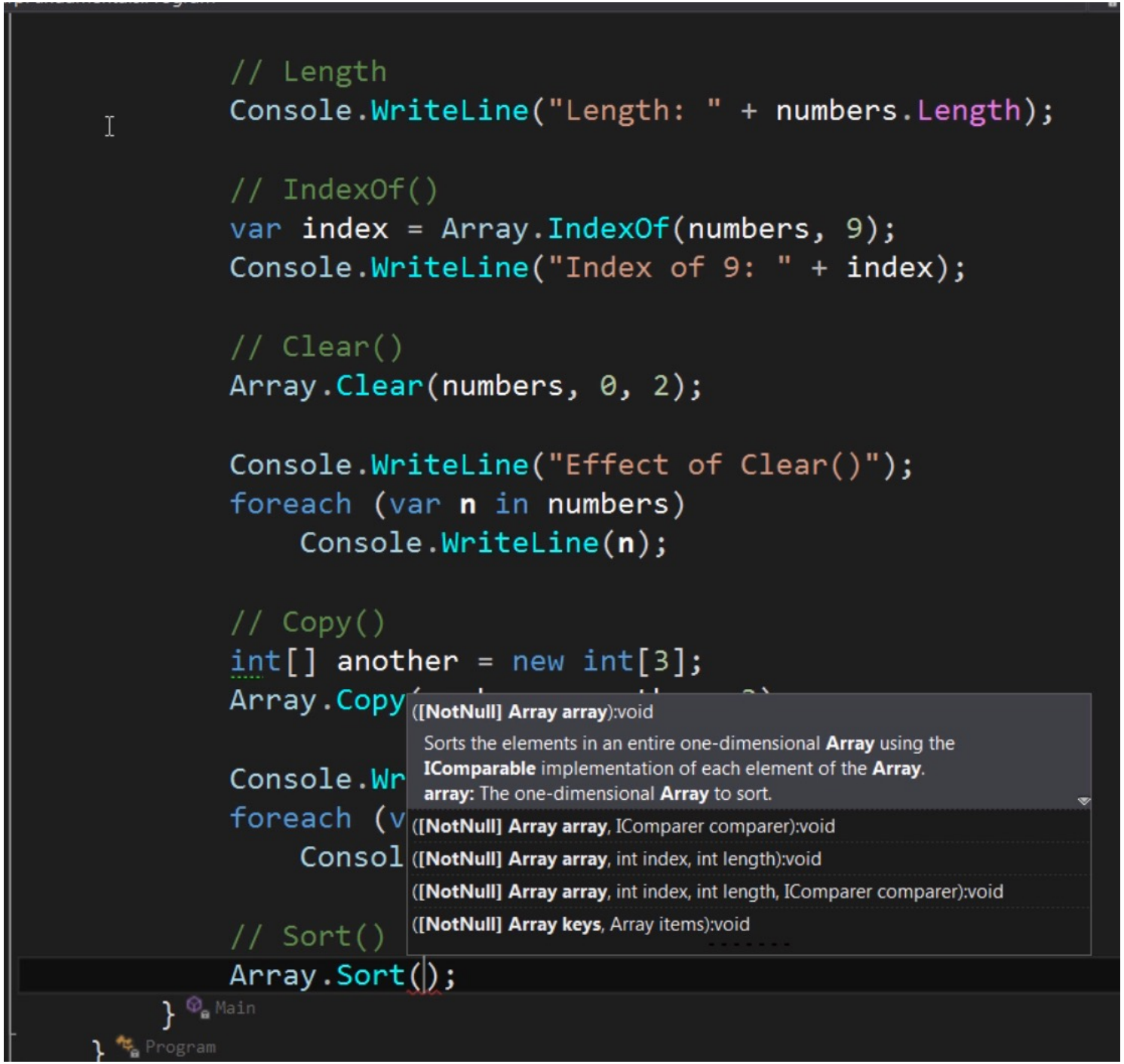
([NotNull] Array sourceArray, long sourceIndex, [NotNull] Array destinationArray, long destinationIndex, long length):void

```
            // Length
            Console.WriteLine("Length: " + numbers.Length);

            // IndexOf()
            var index = Array.IndexOf(numbers, 9);
            Console.WriteLine("Index of 9: " + index);

            // Clear()
            Array.Clear(numbers, 0, 2);

            Console.WriteLine("Effect of Clear()");
            foreach (var n in numbers)
                Console.WriteLine(n);

            // Copy()
            int[] another = new int[3];
            Array.Copy(numbers, another, 3);

            Console.WriteLine("Effect of Copy()");
            foreach (var n in another)
                Console.WriteLine(n);
        }
    }
}
```

Main

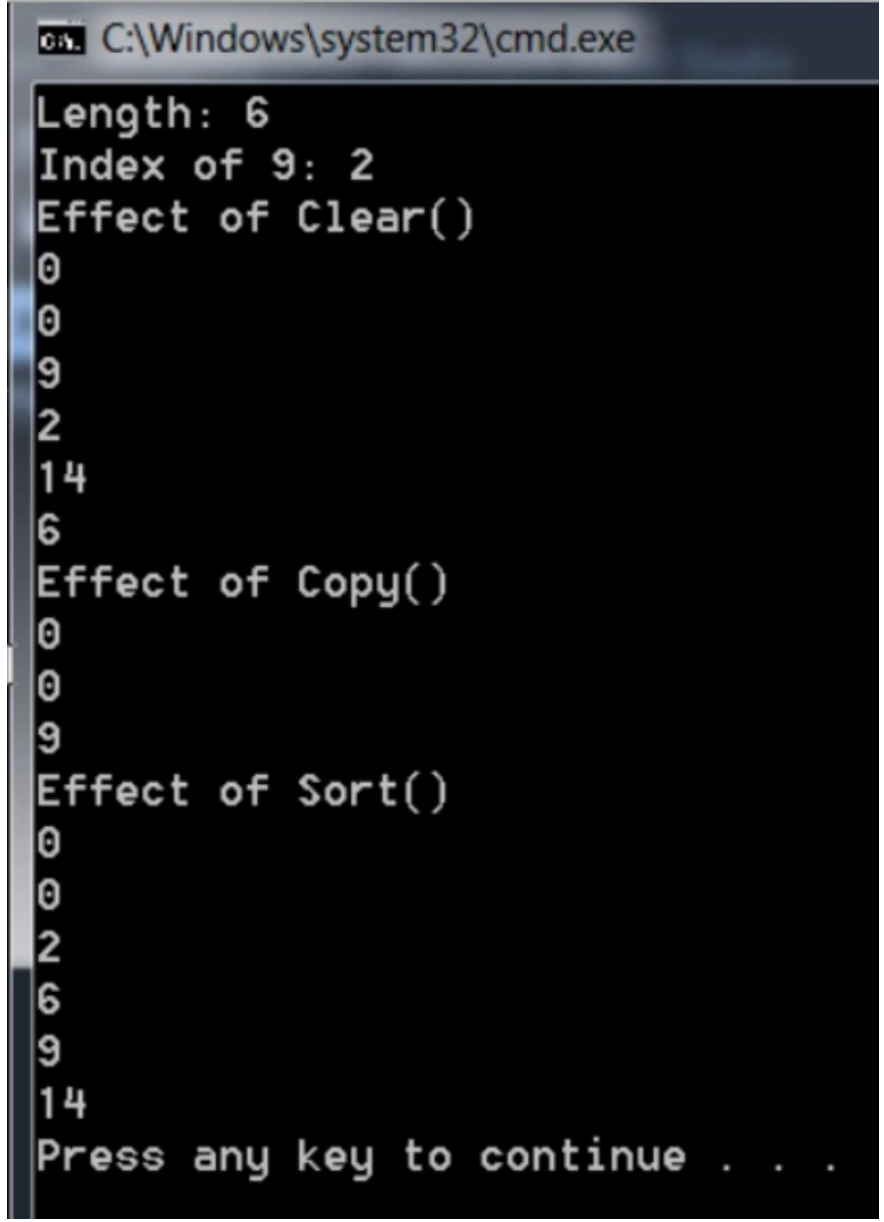
Program




// Length
Console.WriteLine("Length: " + numbers.Length);
// IndexOf()
var index = Array.IndexOf(numbers, 9);
Console.WriteLine("Index of 9: " + index);
// Clear()
Array.Clear(numbers, 0, 2);
Console.WriteLine("Effect of Clear()");
foreach (var n in numbers)
Console.WriteLine(n);
// Copy()
int[] another = new int[3];
Array.Copy
Console.Wr
foreach (v
Consol
// Sort()
Array.Sort();
([NotNull] Array array):void
Sorts the elements in an entire one-dimensional Array using the IComparable implementation of each element of the Array.
array: The one-dimensional Array to sort.
([NotNull] Array array, IComparer comparer):void
([NotNull] Array array, int index, int length):void
([NotNull] Array array, int index, int length, IComparer comparer):void
([NotNull] Array keys, Array items):void
Main
Program

```
// Clear()
Array.Clear(numbers, 0, 2);

Console.WriteLine("Effect of Clear()");
foreach (var n in numbers)
    Console.WriteLine(n);

// Copy()
int[] another = new int[3];
Array.Copy(numbers, another, 3);

Console.WriteLine("Effect of Copy()");
foreach (var n in another)
    Console.WriteLine(n);

// Sort()
Array.Sort(numbers);

Console.WriteLine("Effect of Sort()");
foreach (var n in numbers)
    Console.WriteLine(n);
}
```

Program



```
        // Copy()
        int[] another = new int[3];
        Array.Copy(numbers, another, 3);

        Console.WriteLine("Effect of Copy()");
        foreach (var n in another)
            Console.WriteLine(n);

        // Sort()
        Array.Sort(numbers);

        Console.WriteLine("Effect of Sort()");
        foreach (var n in numbers)
            Console.WriteLine(n);

        // Reverse()
        Array.Reverse(numbers);

        Console.WriteLine("Effect of Reverse()");
        foreach (var n in numbers)
            Console.WriteLine(n);
    } Main
} Program
}
```

```
0
9
2
14
6
Effect of Copy()
0
0
9
Effect of Sort()
0
0
2
6
9
14
Effect of Reverse()
14
9
6
2
0
0
Press any key to continue . . .
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            var numbers = new[] { 3, 7, 9, 2, 14, 6 };

            // Length
            Console.WriteLine("Length: " + numbers.Length);

            // IndexOf()
            var index = Array.IndexOf(numbers, 9);
            Console.Writ

            // Clear()
            Array.Clear(

            Console.Writ
            foreach (var n in numbers)
                Console.WriteLine(n);

            // Copy()
            int[] another = new int[3];
            Array.Copy(numbers, another, 3);

            Console.WriteLine("Effect of Copy()");
            foreach (var n in another)
```

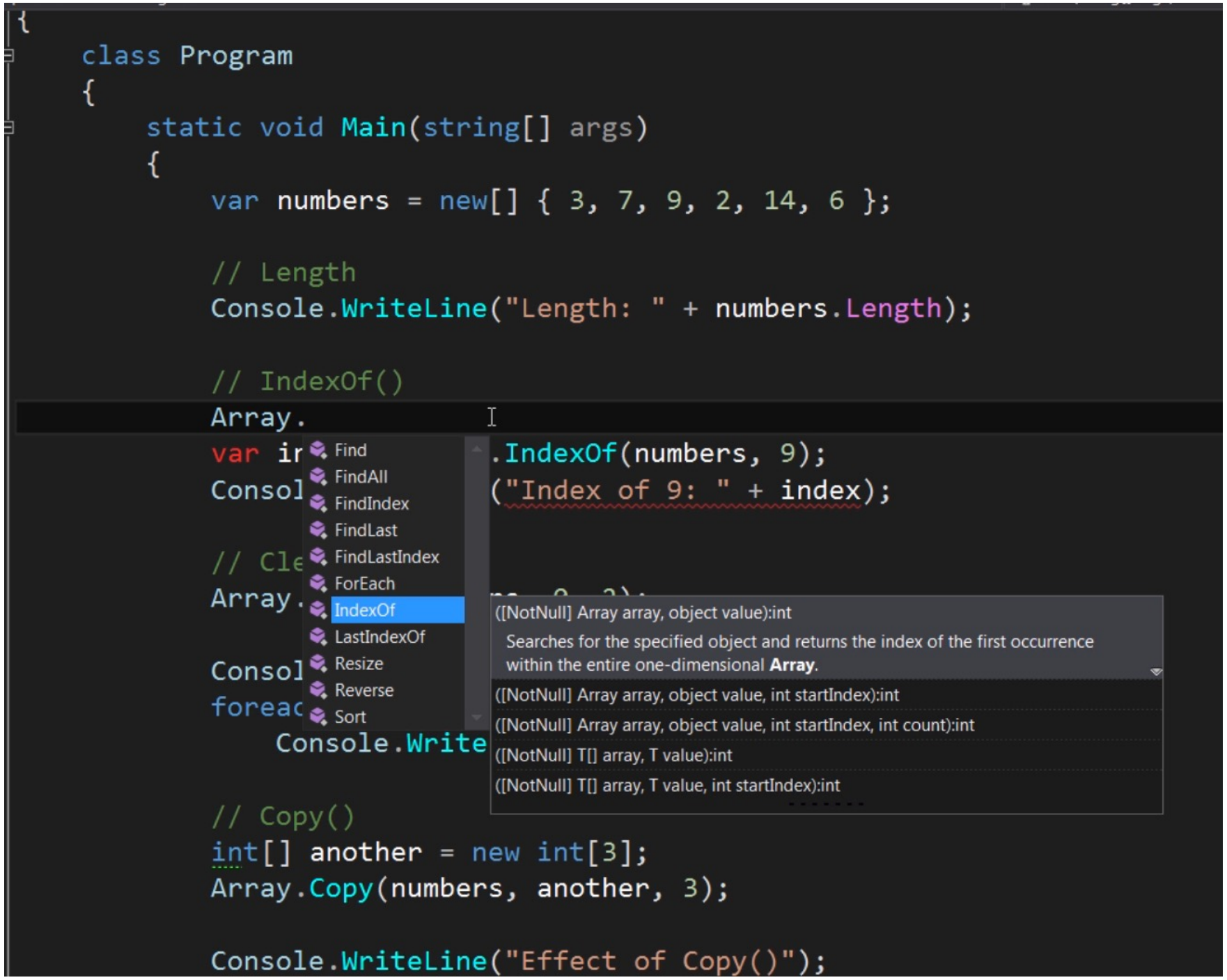
(class) System.Array

Provides methods for creating, manipulating, searching, and sorting arrays, thereby serving as the base class for all arrays in the common language runtime.

(method) int System.Array.IndexOf<int>(int[] array, int value)

Searches for the specified object and returns the index of the first occurrence within the entire **Array**.

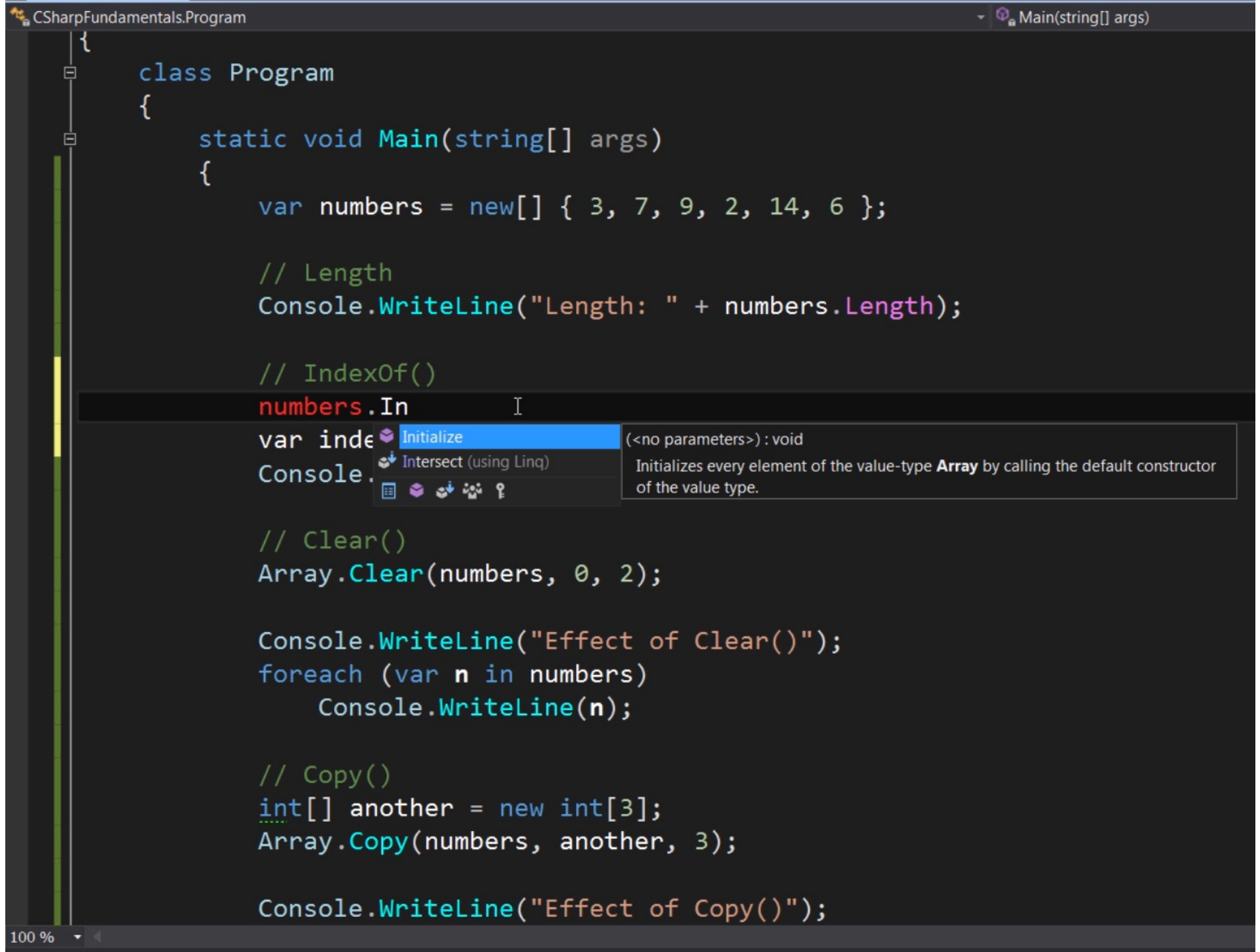
Exceptions:
System.ArgumentNullException: **array** is null.

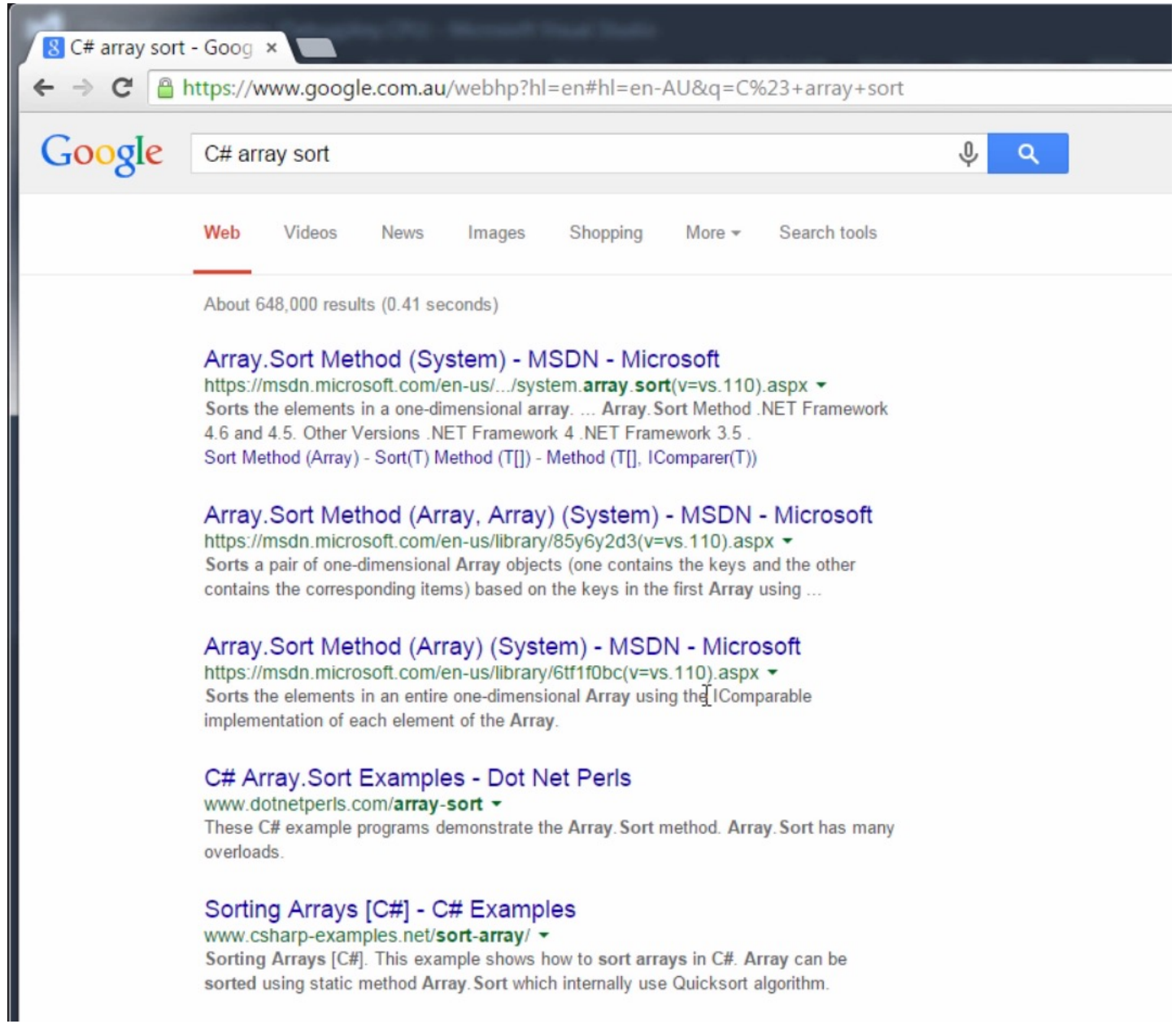

{
class Program
{
static void Main(string[] args)
{
var numbers = new[] { 3, 7, 9, 2, 14, 6 };
// Length
Console.WriteLine("Length: " + numbers.Length);
// IndexOf()
Array.
var ir
Consol
.IndexOf(numbers, 9);
("Index of 9: " + index);
// Cle
Array.
Consol
foreac
Console.Write
Find
FindAll
FindIndex
FindLast
FindLastIndex
ForEach
IndexOf
LastIndexOf
Resize
Reverse
Sort
([NotNull] Array array, object value):int
Searches for the specified object and returns the index of the first occurrence within the entire one-dimensional Array.
([NotNull] Array array, object value, int startIndex):int
([NotNull] Array array, object value, int startIndex, int count):int
([NotNull] T[] array, T value):int
([NotNull] T[] array, T value, int startIndex):int
// Copy()
int[] another = new int[3];
Array.Copy(numbers, another, 3);
Console.WriteLine("Effect of Copy()");

```
{
    class Program
    {
        static void Main(string[] args)
        {
            var numbers = new[] { 3, 7, 9, 2, 14, 6 };

            // Length
            Console.WriteLine("Length: " + numbers.Length);

            // IndexOf()
            numbers.
            var inde                                    exOf(numbers, 9);
            Console.                                    dex of 9: " + index);

            // Clear
            Array.Cl

            Console.                                    fect of Clear()");
            foreach                                     ers)
                Cons                                    (n);

            // Copy()
            int[] another = new int[3];
            Array.Copy(numbers, another, 3);

            Console.WriteLine("Effect of Copy()");
```

GetUpperBound
GetValue
Initialize
IsFixedSize
IsReadOnly
IsSynchronized
Length
LongLength
MemberwiseClone
Rank
SetValue

Property int System.Array.Length
Gets a 32-bit integer that represents the total number of elements in all the dimensions of the **Array**.


CSharpFundamentals.Program
Main(string[] args)
{
class Program
{
static void Main(string[] args)
{
var numbers = new[] { 3, 7, 9, 2, 14, 6 };
// Length
Console.WriteLine("Length: " + numbers.Length);
// IndexOf()
numbers.In
var inde
Console.
Initialize
Intersect (using Linq)
(<no parameters>) : void
Initializes every element of the value-type Array by calling the default constructor of the value type.
// Clear()
Array.Clear(numbers, 0, 2);
Console.WriteLine("Effect of Clear()");
foreach (var n in numbers)
Console.WriteLine(n);
// Copy()
int[] another = new int[3];
Array.Copy(numbers, another, 3);
Console.WriteLine("Effect of Copy()");
100 %


C# array sort - Goog
https://www.google.com.au/webhp?hl=en#hl=en-AU&q=C%23+array+sort
Google
C# array sort
Web
Videos
News
Images
Shopping
More
Search tools
About 648,000 results (0.41 seconds)
Array.Sort Method (System) - MSDN - Microsoft
https://msdn.microsoft.com/en-us/.../system.array.sort(v=vs.110).aspx
Sorts the elements in a one-dimensional array. ... Array.Sort Method .NET Framework 4.6 and 4.5. Other Versions .NET Framework 4 .NET Framework 3.5 .
Sort Method (Array) - Sort(T) Method (T[]) - Method (T[], IComparer(T))
Array.Sort Method (Array, Array) (System) - MSDN - Microsoft
https://msdn.microsoft.com/en-us/library/85y6y2d3(v=vs.110).aspx
Sorts a pair of one-dimensional Array objects (one contains the keys and the other contains the corresponding items) based on the keys in the first Array using ...
Array.Sort Method (Array) (System) - MSDN - Microsoft
https://msdn.microsoft.com/en-us/library/6tf1f0bc(v=vs.110).aspx
Sorts the elements in an entire one-dimensional Array using the IComparable implementation of each element of the Array.
C# Array.Sort Examples - Dot Net Perls
www.dotnetperls.com/array-sort
These C# example programs demonstrate the Array.Sort method. Array.Sort has many overloads.
Sorting Arrays [C#] - C# Examples
www.csharp-examples.net/sort-array/
Sorting Arrays [C#]. This example shows how to sort arrays in C#. Array can be sorted using static method Array.Sort which internally use Quicksort algorithm.

Array.Sort Method

.NET Framework 4.6 and 4.5 | Other Versions

Sorts the elements in a one-dimensional array.

This member is overloaded. For complete information about this member, including syntax, usage, and examples, click a name in the overload list.

Overload List

Click to collapse. Double-click to collapse all.

	Name	Description
	Sort<T>(T[])	Sorts the elements in an entire Array using the IComparable<T> generic interface implementation of each element of the Array.
	Sort(Array)	Sorts the elements in an entire one-dimensional Array using the IComparable implementation of each element of the Array.
	Sort<T>(T[], IComparer<T>)	Sorts the elements in an Array using the specified IComparer<T> generic interface.
	Sort<T>(T[], Comparison<T>)	Sorts the elements in an Array using the specified Comparison<T>.
	Sort<TKey, TValue>(TKey[], TValue[])	Sorts a pair of Array objects (one contains the keys and the other contains the corresponding items) based on the keys in the first Array using the IComparable<T> generic interface implementation of each key.
	Sort(Array, Array)	Sorts a pair of one-dimensional Array objects (one contains the keys and the other contains the corresponding items) based on the keys in the first Array using the IComparable implementation of each key.
	Sort(Array, IComparer)	Sorts the elements in a one-dimensional Array using the specified IComparer.
	Sort<T>(T[], Int32, Int32)	Sorts the elements in a range of elements in an Array using the IComparable<T> generic interface implementation of each element of the Array.
	Sort<TKey, TValue>(TKey[], TValue[], IComparer<TKey>)	Sorts a pair of Array objects (one contains the keys and the other contains the corresponding items) based on the keys in the first Array using the specified IComparer<T> generic interface.

Sort(Array, IComparer)	Sorts the elements in a one-dimensional Array using the specified IComparer.
Sort<T>(T[], Int32, Int32)	Sorts the elements in a range of elements in an Array using the IComparable<T> generic interface implementation of each element of the Array.
Sort<TKey, TValue>(TKey[], TValue[], IComparer<TKey>)	Sorts a pair of Array objects (one contains the keys and the other contains the corresponding items) based on the keys in the first Array using the specified IComparer<T> generic interface.
Sort(Array, Array, IComparer)	Sorts a pair of one-dimensional Array objects (one contains the keys and the other contains the corresponding items) based on the keys in the first Array using the specified IComparer.
Sort(Array, Int32, Int32)	Sorts the elements in a range of elements in a one-dimensional Array using the IComparable implementation of each element of the Array.
Sort<T>(T[], Int32, Int32, IComparer<T>)	Sorts the elements in a range of elements in an Array using the specified IComparer<T> generic interface.
Sort<TKey, TValue>(TKey[], TValue[], Int32, Int32)	Sorts a range of elements in a pair of Array objects (one contains the keys and the other contains the corresponding items) based on the keys in the first Array using the IComparable<T> generic interface implementation of each key.
Sort(Array, Array, Int32, Int32)	Sorts a range of elements in a pair of one-dimensional Array objects (one contains the keys and the other contains the corresponding items) based on the keys in the first Array using the IComparable implementation of each key.
Sort(Array, Int32, Int32, IComparer)	Sorts the elements in a range of elements in a one-dimensional Array using the specified IComparer.
Sort<TKey, TValue>(TKey[], TValue[], Int32, Int32, IComparer<TKey>)	Sorts a range of elements in a pair of Array objects (one contains the keys and the other contains the corresponding items) based on the keys in the first Array using the specified IComparer<T> generic interface.
Sort(Array, Array, Int32, Int32, IComparer)	Sorts a range of elements in a pair of one-dimensional Array objects (one contains the keys and the other contains the corresponding items) based on the keys in the first Array using the specified IComparer.

Top

See Also

Reference

Array Class
System Namespace

Inheritance Hierarchy

System.Object
System.Array

Namespace: System
Assembly: mscorlib (in mscorlib.dll)

Syntax

C# | C++ | F# | JScript | VB

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

The Array type exposes the following members.

Properties

	Name	Description
	IsFixedSize	Gets a value indicating whether the Array has a fixed size.
	IsReadOnly	Gets a value indicating whether the Array is read-only.
	IsSynchronized	Gets a value indicating whether access to the Array is synchronized (thread safe).
	Length	Gets a 32-bit integer that represents the total number of elements in all the dimensions of the Array.
	LongLength	Gets a 64-bit integer that represents the total number of elements in all the dimensions of the Array.
	Rank	Gets the rank (number of dimensions) of the Array. For example, a one-dimensional array returns 1, a two-dimensional array returns 2, and so on.
	SyncRoot	Gets an object that can be used to synchronize access to the Array.

Top

Methods