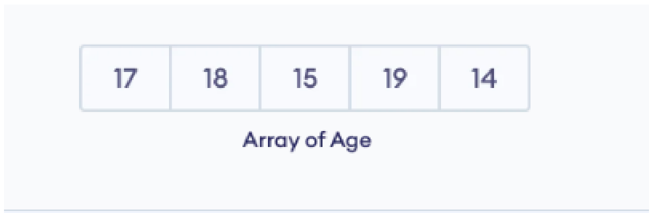
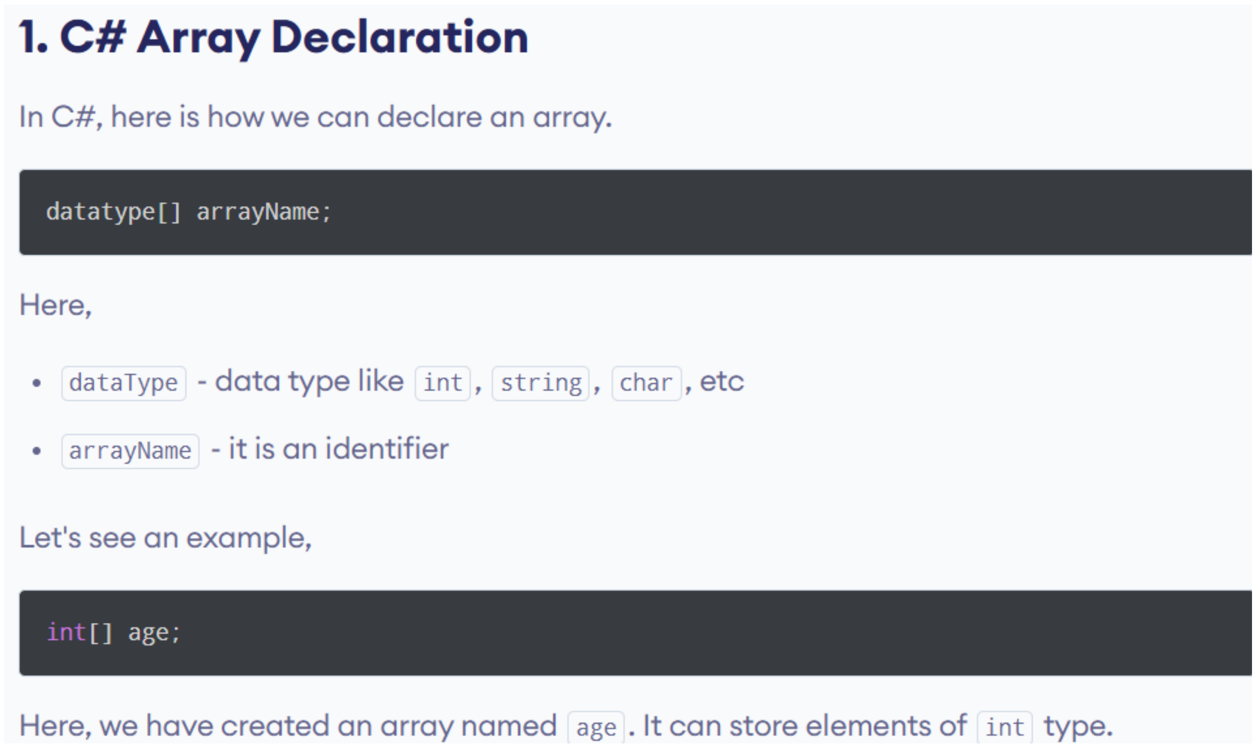
C# Arrays

An array is a collection of similar types of data. For example,

Suppose we need to record the age of 5 students. Instead of creating 5 separate variables, we can simply create an array:



Elements of an Array



1. C# Array Declaration

In C#, here is how we can declare an array.

```
datatype[] arrayName;
```

Here,

- `dataType` - data type like `int`, `string`, `char`, etc
- `arrayName` - it is an identifier

Let's see an example,

```
int[] age;
```

Here, we have created an array named `age`. It can store elements of `int` type.

But how many elements can it store?

To define the number of elements that an array can hold, we have to allocate memory for the array in C#. For example,

```
// declare an array
int[] age;

// allocate memory for array
age = new int[5];
```

Here, `new int[5]` represents that the array can store 5 elements. We can also say the size/length of the array is 5.

> **Note:** We can also declare and allocate the memory of an array in a single line. For example,
>
> ```
> int[] age = new int[5];
> ```

2. Array initialization in C#

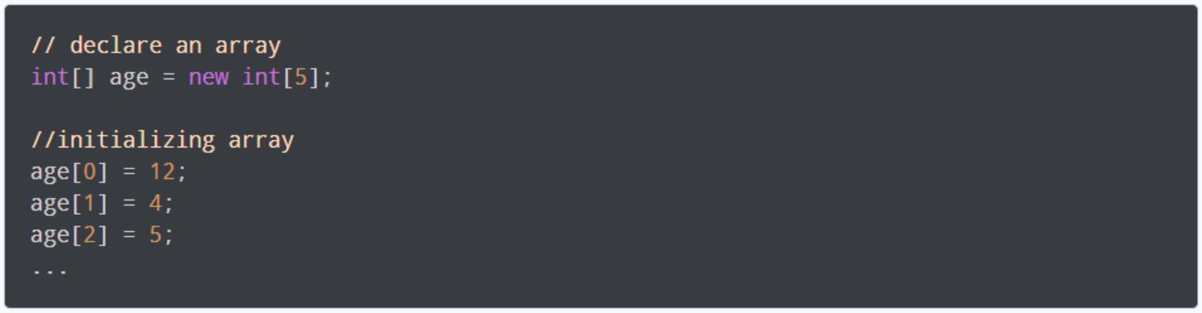
In C#, we can initialize an array during the declaration. For example,

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

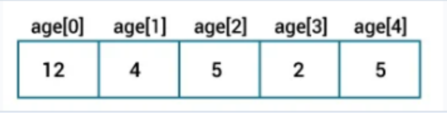
Here, we have created an array named numbers and initialized it with values **1**, **2**, **3**, **4**, and **5** inside the curly braces.

Note that we have not provided the size of the array. In this case, the C# automatically specifies the size by counting the number of elements in the array (i.e. 5).

In an array, we use an **index number** to determine the position of each array element. We can use the index number to initialize an array in C#. For example,



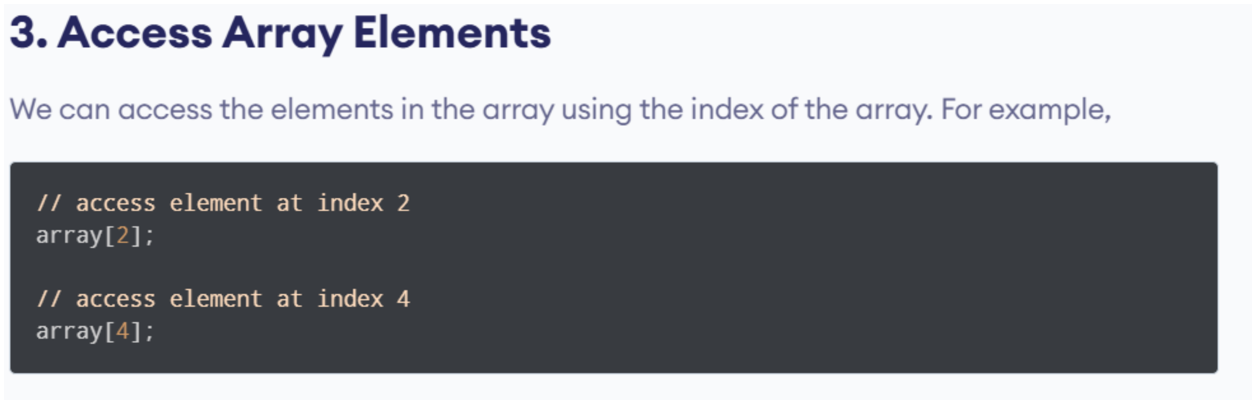
```
// declare an array
int[] age = new int[5];

//initializing array
age[0] = 12;
age[1] = 4;
age[2] = 5;
...
```



C# Array Initialization

Note:

- An array index always starts at 0. That is, the first element of an array is at index 0.
- If the size of an array is 5, the index of the last element will be at 4 (5 - 1).



3. Access Array Elements

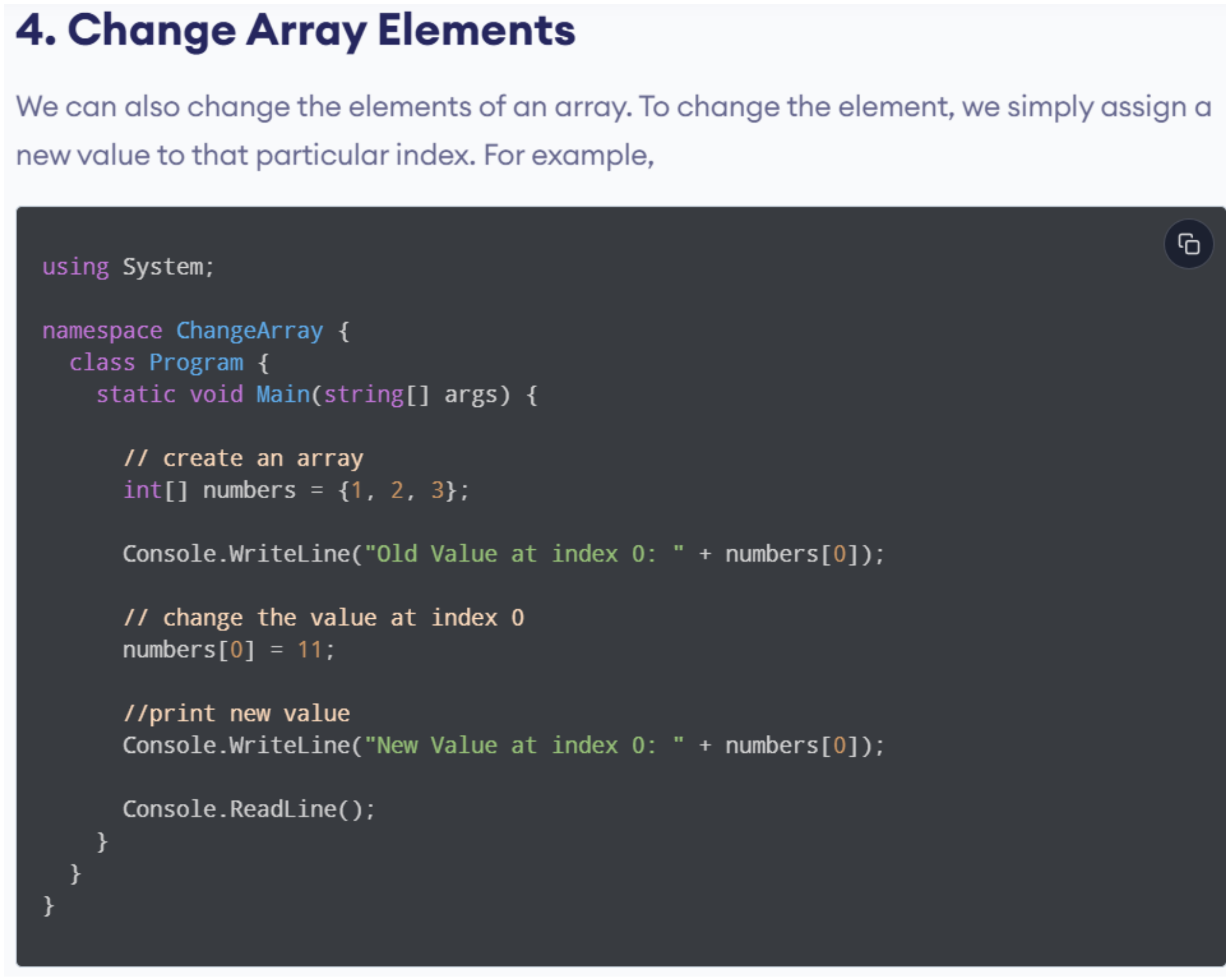
We can access the elements in the array using the index of the array. For example,

```
// access element at index 2
array[2];

// access element at index 4
array[4];
```

```
using System;

namespace AccessArray {
  class Program  {
    static void Main(string[] args) {

      // create an array
      int[] numbers = {1, 2, 3};

      //access first element
      Console.WriteLine("Element in first index : " + numbers[0]);

      //access second element
      Console.WriteLine("Element in second index : " + numbers[1]);

      //access third element
      Console.WriteLine("Element in third index : " + numbers[2]);

      Console.ReadLine();

    }
  }
}
```

Output

```
Element in first index : 1
Element in second index : 2
Element in third index : 3
```



4. Change Array Elements

We can also change the elements of an array. To change the element, we simply assign a new value to that particular index. For example,

```
using System;

namespace ChangeArray {
  class Program {
    static void Main(string[] args) {

      // create an array
      int[] numbers = {1, 2, 3};

      Console.WriteLine("Old Value at index 0: " + numbers[0]);

      // change the value at index 0
      numbers[0] = 11;

      //print new value
      Console.WriteLine("New Value at index 0: " + numbers[0]);

      Console.ReadLine();
    }
  }
}
```

Output

```
Old Value at index 0: 1
New Value at index 0: 11
```

In the above example, the initial value at index 0 is 1. Notice the line,

```
//change the value at index 0
numbers[0] = 11;
```

Here, we are assigning a new value of **11** to the index 0. Now, the value at index 0 is changed from **1** to **11**.

5. Iterating C# Array using Loops

In C#, we can use loops to iterate through each element of an array. For example,

Example: Using for loop

```
using System;

namespace AccessArrayFor {
  class Program {
    static void Main(string[] args) {

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

      for(int i=0; i < numbers.Length; i++) {
        Console.WriteLine("Element in index " + i + ": " + numbers[i]);
      }

      Console.ReadLine();
    }
  }
}
```

Output

```
Element in index 0: 1
Element in index 1: 2
Element in index 2: 3
```

In the above example, we have used a for loop to iterate through the elements of the array, `numbers`. Notice the line,

```
numbers.Length
```

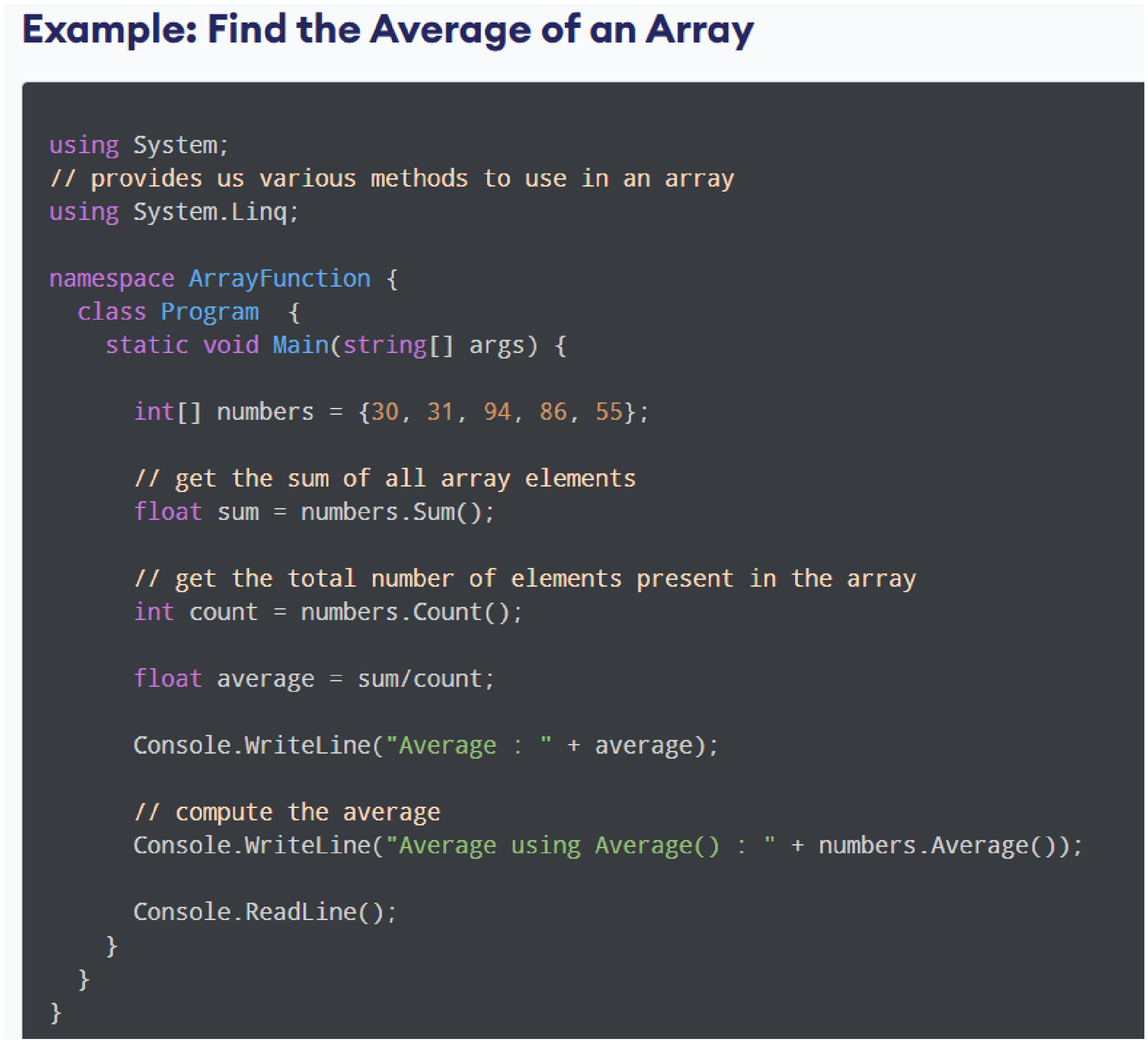
Here, the `Length` property of the array gives the size of the array.

6. C# Array Operations using System.Linq

In C#, we have the `System.Linq` namespace that provides different methods to perform various operations in an array. For example,

Example: Find Minimum and Maximum Element

```
using System;

 // provides us various methods to use in an array
using System.Linq;

namespace ArrayMinMax {
  class Program  {
    static void Main(string[] args) {

      int[] numbers = {51, 1, 3, 4, 98};

      // get the minimum element
      Console.WriteLine("Smallest  Element: " + numbers.Min());

      // Max() returns the largest number in array
      Console.WriteLine("Largest Element: " + numbers.Max());

      Console.ReadLine();
    }
  }
}
```

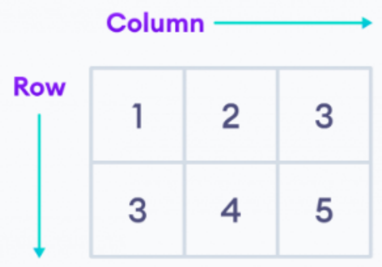


Example: Find the Average of an Array

```
using System;
// provides us various methods to use in an array
using System.Linq;

namespace ArrayFunction {
  class Program  {
    static void Main(string[] args) {

      int[] numbers = {30, 31, 94, 86, 55};

      // get the sum of all array elements
      float sum = numbers.Sum();

      // get the total number of elements present in the array
      int count = numbers.Count();

      float average = sum/count;

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

      // compute the average
      Console.WriteLine("Average using Average() : " + numbers.Average());

      Console.ReadLine();
    }
  }
}
```

Two-dimensional array in C#

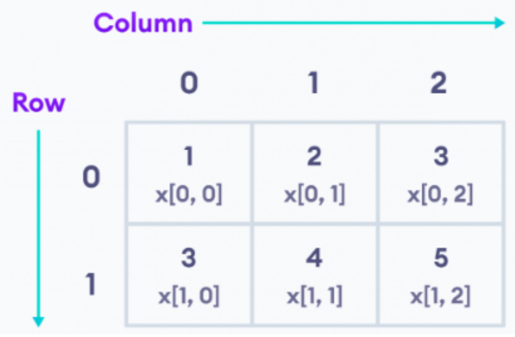
A two-dimensional array consists of single-dimensional arrays as its elements. It can be represented as a table with a specific number of rows and columns.



C# Two-dimensional array

Here, rows **{1, 2, 3}** and **{3, 4, 5}** are elements of a 2D array.

```
// a 2D array
int[ , ] x = { { 1, 2 ,3}, { 3, 4, 5 } };

// access first element from first row
x[0, 0];  // returns 1

// access third element from second row
x[1, 2];  // returns 5

// access third element from first row
x[0, 2];  // returns 3
```



Example: C# 2D Array

```csharp
using System;

namespace MultiDArray {
  class Program {
    static void Main(string[] args) {

        //initializing 2D array
       int[ , ] numbers = {{2, 3}, {4, 5}};

        // access first element from the first row
       Console.WriteLine("Element at index [0, 0] : "+numbers[0, 0]);

        // access first element from second row
       Console.WriteLine("Element at index [1, 0] : "+numbers[1, 0]);
    }
  }
}
```

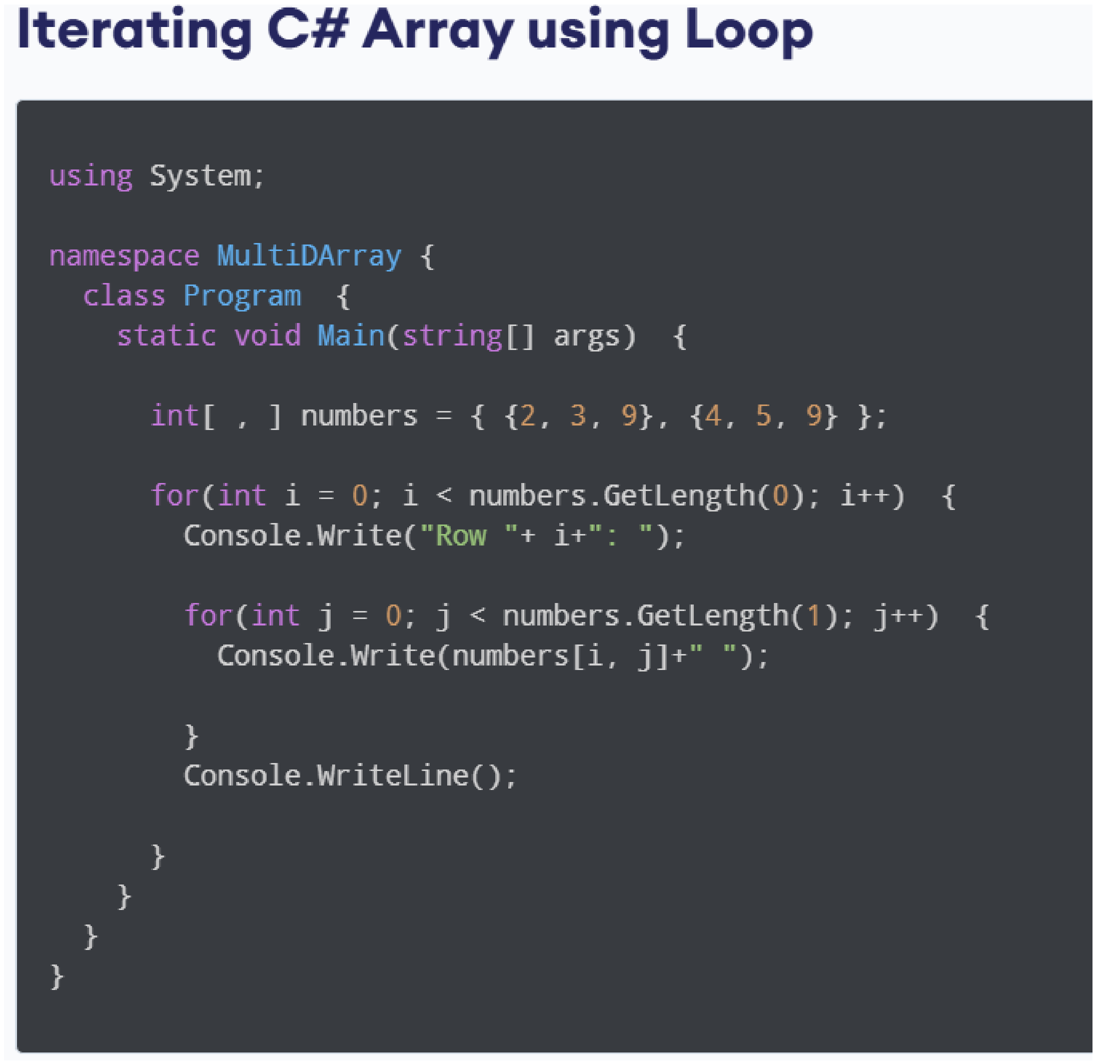
Output

```
Element at index [0, 0] : 2
Element at index [1, 0] : 4
```

Change Array Elements

We can also change the elements of a two-dimensional array. To change the element, we simply assign a new value to that particular index. For example,

```
using System;

namespace MultiDArray {
  class Program {
    static void Main(string[] args) {

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

     // old element
    Console.WriteLine("Old element at index [0, 0] : "+numbers[0, 0]);

      // assigning new value
    numbers[0, 0] = 222;

      // new element
    Console.WriteLine("New element at index [0, 0] : "+numbers[0, 0]);
    }
  }
}
```



Iterating C# Array using Loop

```
using System;

namespace MultiDArray {
  class Program  {
    static void Main(string[] args)  {

      int[ , ] numbers = { {2, 3, 9}, {4, 5, 9} };

      for(int i = 0; i < numbers.GetLength(0); i++)  {
        Console.Write("Row "+ i+": ");

        for(int j = 0; j < numbers.GetLength(1); j++)  {
          Console.Write(numbers[i, j]+" ");

        }
        Console.WriteLine();

      }
    }
  }
}
```