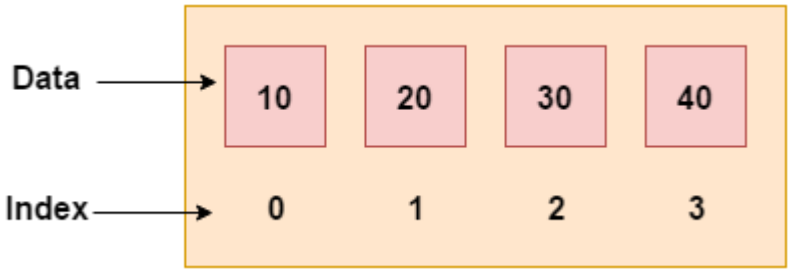
Arrays

Array in C# is a group of similar types of elements that have contiguous memory location. In C#, Array is an ***object*** of base type **System.Array**. In C#, Array index starts from 0. We can store only fixed set of elements in C# Array.



An Array has the following properties:

- An Array can be Single-Dimensional, Multidimensional or Jagged.
- 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 derived from the abstract base type Array.

Single Dimensional Array

To create single dimensional Array, you need to use square brackets [] after the type.

```
double[] doubleArray = new double[5];

char[] charArray = new char[5];

bool[] boolArray = new bool[2];

string[] stringArray = new string[10];
```

```
using System;
public class ArrayExample

{
   public static void Main(string[] args)
   {
      int[] arr = new int[5];//creating Array
      arr[0] = 10;//initializing Array
      arr[2] = 20;
      arr[4] = 30;
        //traversing Array
      for (int i = 0; i < arr.Length; i++)
      {
         Console.WriteLine(arr[i]);
      }
   }
}
```

Output:

```
10
0
20
0
30
```

```
using System;
namespace ArrayApplication23
{
    0 references
    class Program
        {
            0 references
            static void Main(string[] args) {
            try {
                int i;
                int[] arr = new int[5];
                for (i = 0; i < 5; i++) {
                    Console.Write("\nEnter your number:\t");
                    arr[i] = Convert.ToInt32(Console.ReadLine());
                }
                Console.WriteLine("\n\n");
                for (i = 0; i < 5; i++) {
                    Console.WriteLine("you entered {0}", arr[i]);
                }
                Console.ReadLine();
            }
            catch(Exception e) {
                Console.WriteLine(e);
            }

            }
        }
    }
```

Output:

```
file:///c:/users/admin/documents/visual studio 2015/Project...

Enter your number:      2

Enter your number:      3

Enter your number:      4

Enter your number:      5

Enter your number:      6


you entered 2
you entered 3
you entered 4
you entered 5
you entered 6
```

Multidimensional Arrays

The multidimensional Array is also known as rectangular Arrays in C#. It can be two dimensional or three dimensional. The data is stored in tabular form (row * column) which is also known as matrix.

To create multidimensional Array, we need to use comma inside the square brackets. For example:

1. int[,] arr=new int[3,3];//declaration of 2D Array
2. int[,,] arr=new int[3,3,3];//declaration of 3D Array

```
using System;
public class MultiArrayExample
{
   public static void Main(string[] args)
   {
      int[,] arr=new int[3,3];//declaration of 2D Array
      arr[0,1]=10;//initialization
      arr[1,2]=20;
      arr[2,0]=30;
      //traversal
      for(int i=0;i<3;i++)
      {
         for(int j=0;j<3;j++)
         {
            Console.Write(arr[i,j]+" ");
         }
         Console.WriteLine();//new line at each row
      }
   }
}
```

Output:

```
0  10  0
0  0  20
30  0  0
```

Declaration and initialization at same time

1 int[,] arr = new int[3,3]= { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } }; We can omit the Array size.

2 int[,] arr = new int[,] { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

We can omit the new operator also.

3 int[,] arr = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

```
using System;
namespace ArrayApplication23
{
   class Program
   {
      static void Main(string[] args)
      {
         try
         {
            int i, j;
            string[,] Books = new string[3, 3];
            for (i = 0; i < 3; i++)
            {
               for (j = 0; j < 3; j++)
               {
                  Console.Write("\nEnter Book Name for {0}. Row and {1}.
                  column:\t", i + 1, j + 1);
                   Books[i, j] = Console.ReadLine();
               }
            }
            Console.WriteLine("\n\n========================");
            Console.WriteLine("All the element of Books array is:\n\n");
            for (i = 0; i < 3; i++)
            {
               for (j = 0; j < 3; j++)
               {
                  Console.Write("{0}\t", Books[i, j]);
               }
               Console.Write("\n");
```

```
            }
            Console.WriteLine("\n\n========================");
        }
        catch (Exception)
        {
            throw;
        }
        Console.ReadLine();
    }
}}
```

Output:

```
file:///c:/users/admin/documents/visual studio 2015/Projects/ArrayApplication23/ArrayApplication23/bin/Debug/ArrayApplication23.EXE
Enter Book Name for 1. Row and 3. column:       Java

Enter Book Name for 2. Row and 1. column:       C#

Enter Book Name for 2. Row and 2. column:       SQL

Enter Book Name for 2. Row and 3. column:       HTML

Enter Book Name for 3. Row and 1. column:       CSS

Enter Book Name for 3. Row and 2. column:       JS

Enter Book Name for 3. Row and 3. column:       .Net


========================
All the element of Books array is:


C       C++     Java
C#      SQL     HTML
CSS     JS      .Net


========================
```

Jagged Arrays

Jagged Array is also known as "Array of Arrays" because its elements are Arrays. The element size of jagged Array can be different.

Declaration of Jagged Array

Jagged Array has two elements.

int[][] arr = new int[2][];

Initialization of Jagged Array

arr[0] = new int[4];

arr[1] = new int[6];

Initialization and filling elements in Jagged Array

arr[0] = new int[4] { 11, 21, 56, 78 };

arr[1] = new int[6] { 42, 61, 37, 41, 59, 63 };

Here, size of elements in jagged Array is optional. So, you can write above code as given below:

arr[0] = new int[] { 11, 21, 56, 78 };

arr[1] = new int[] { 42, 61, 37, 41, 59, 63 };

```
public class JaggedArrayTest
{
   public static void Main()
   {
      int[][] arr = new int[2][];// Declare the Array

      arr[0] = new int[] { 11, 21, 56, 78 };// Initialize the Array
      arr[1] = new int[] { 42, 61, 37, 41, 59, 63 };

      // Traverse Array elements
      for (int i = 0; i < arr.Length; i++)
      {
         for (int j = 0; j < arr[i].Length; j++)
         {
            System.Console.Write(arr[i][j]+" ");
         }
         System.Console.WriteLine();
      }
   }
}
```

Output:

```
11 21 56 78
42 61 37 41 59 63
```

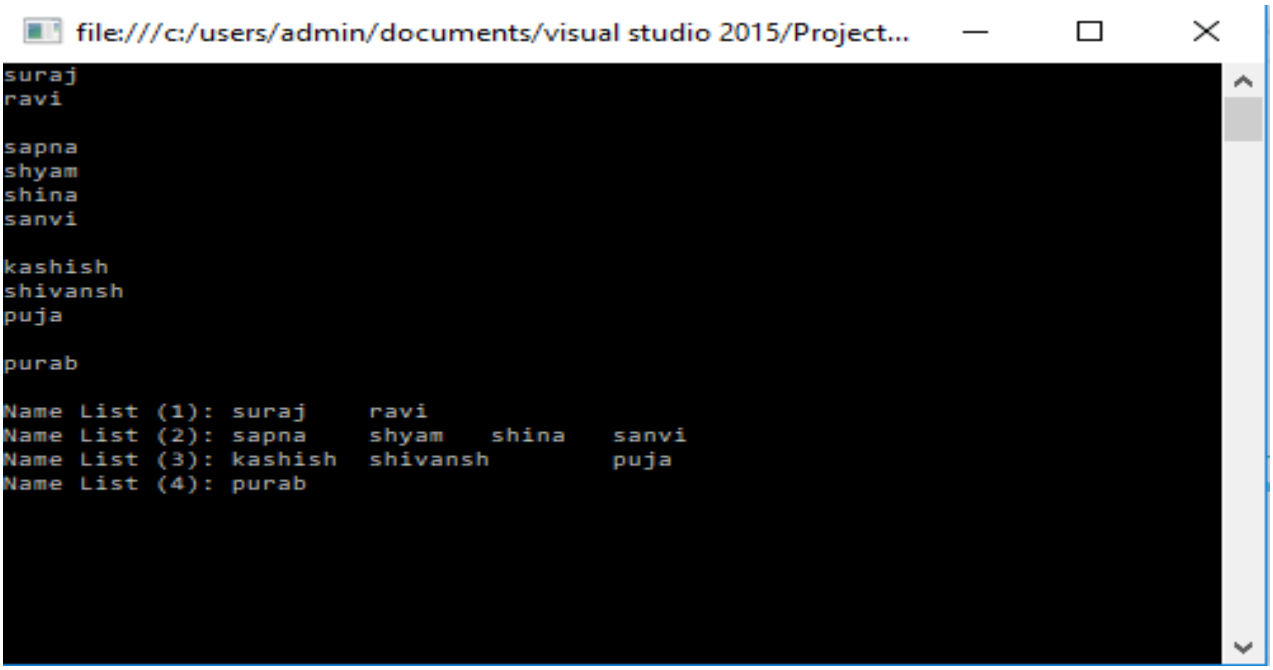
Initialization of Jagged Array upon Declaration

```
int[][] arr = new int[3][]{
        new int[] { 11, 21, 56, 78 },
        new int[] { 2, 5, 6, 7, 98, 5 },
        new int[] { 2, 5 }
        };
```

```
using System;
namespace ArrayApplication23
{
    class Program
    {
        static void Main(string[] args)
        {
            string name;
            string[][] Members = new string[4][]
            {
                new string[2],
                new string[4],
                new string[3],
                new string[1],
                        //new string[]{"Rocky", "Sam", "Alex"},
                        // new string[]{"Peter", "Sonia", "Prety", "Ronnie","Dino"},
                       // new string[]{"Yomi", "John", "Sandra", "Alex" , "Shaun"},
                        // new string[]{"Teena", "Mathew", "Arnold", "Stallone",
                       };
            for (int i = 0; i < Members.Length; i++)
            {
                for (int j = 0;j<Members[i].Length; j++)
                {
                  Members[i][j]  = Console.ReadLine();
                }
                System.Console.WriteLine();
            }
        //accessing the Array
            for (int i = 0; i < Members.Length; i++)
            {
                System.Console.Write("Name List ({0}): ", i + 1);
```

```
            for (int j = 0; j < Members[i].Length; j++)
            {
                System.Console.Write(Members[i][j] + "\t");
            }
            System.Console.WriteLine();
        }
        Console.ReadKey();
    }
  }
}
```

Output:



References:

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/Arrays/

https://www.javatpoint.com/c-sharp-Array

https://www.tutorialspoint.com/csharp/csharp_Arrays.htm

http://www.c-sharpcorner.com/article/working-with-Arrays-in-C-Sharp/

By

Indu Sharma