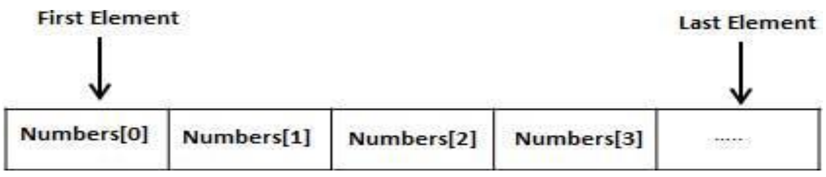
Arrays in C#

An array stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type stored at contiguous memory locations. In C#, an array **index** starts at **zero** (i.e. arrays are zero based). That means the **first item** of an array starts at the **0thposition**. The position of the **last item** on an array will **total number of items - 1**. So if an array has 10 items, the last 10th item is at 9th position.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



Array Types

Arrays can be divided into the following four categories.

- Single-dimensional arrays
- Multidimensional arrays or rectangular arrays
- Jagged arrays
- Mixed arrays.

1. Single Dimension Arrays (1-D Arrays)

Single-dimensional arrays are the simplest form of arrays. These types of arrays are used to store number of items of a predefined type. All items in a single dimension array are stored contiguously ***starting from 0 to the size of the array -1***.

Declaring Arrays

To declare an array in C#, you can use the following syntax:

```
datatype[] arrayName;
```

where,

- d*atatype* is used to specify the type of elements in the array.
- [ ] specifies the rank of the array. The rank specifies the size of the array.
- arrayName specifies the name of the array.

Examples:

```
int[] x;  // can store int values
string[] s; // can store string values
double[] d; // can store double values
```

<u>**Note**</u> : Only Declaration of an array doesn't allocate memory to the array. For that array must be **initialized**.

<u>Initializing an Array</u>

Declaring an array does not initialize the array in the memory. When the array variable is initialized, you can assign values to the array. ***Array is a reference type***, so you need to use the **new** keyword to create an instance of the array.

Syntax :

```
type [ ] < ArrayName > = new < datatype > [size];
```

Here:

- **type** specifies the type of data being allocated.
- **size** specifies the number of elements in the array, and

- **Name_Array** is the name of array variable.
- **new** will allocate memory to an array according to its size.

Examples:

To Show Different ways for the Array Declaration and Initialization For example:

Example 1: Below statement declares & initializes **int** type array that can store five **int** values.

```
// defining array with size 5.
// But not assigns values
int[] intArray1 = new int[5];
```

Note:

When you create an array, C# compiler implicitly initializes each array element to a default value depending on the array type. For example, for an **int array** all elements are initialized to 0.

Example 2: defining array and assigning values at the same time

```
int[] Array2 = new int[5]{1, 2, 3, 4, 5};
```

Example 3: the value of the array is directly initialized without taking its size. So, array size will automatically be the number of values which is directly taken.

```
int[] Array3 = {1, 2, 3, 4, 5};
```

Assigning Values to an Array

You can assign values to individual array elements after array declaration, by using the index number, like:

```
double[] A = new A[10];
```

```
A[0] = 4500.0;
```

You can assign values to the array at the time of declaration as shown:

```
double[] A = { 2340.0, 4523.69, 3421.0};
```

You can also create and initialize an array as shown:

```
int [] marks = new int[5] { 99, 98, 92, 97, 95};
```

You may also omit the size of the array (i.e. a dynamic length array) as shown:

```
int [] marks = new int[] { 99, 98, 92, 97, 95};
```

You can copy an array variable into another target array variable. In such case, both the target and source point to the same memory location:

```
int [] marks = new int[] { 99, 98, 92, 97, 95};
int[] score = marks;
```

Accessing Array Elements

At the time of initialization, we can assign the value. But, we can also assign the value of array using its index randomly after the declaration and initialization. We can access an array value through indexing, placed index of the element within square brackets with the array name.

Example:

```
//declares & initializes int type array
  int[] A = new int[5];
 // assign the value 10 in array on its index 0
   A[0] = 10;
// assign the value 30 in array on its index 2
   A[2] = 30;
// assign the value 20 in array on its index 1
   A[1] = 20;
// assign the value 50 in array on its index 4
   A[4] = 50;
// assign the value 40 in array on its index 3
   A[3] = 40;
// Accesing array elements using index
```

```
Console.WriteLine(A[0]) ;  //Print the 1st element (i.e. 10)
Console.WriteLine(A[2]) ;  // Print the 2nd element (i.e. 30)
```

<u>Implementation:</u> Accessing Array elements using different loops

The following C# program illustrates creating an array of integers, puts some values in the array, and prints each value to standard output.

```
using System;
namespace AccDefLp {

class ADL {

   // Main Method
   public static void Main()
   {

      // declares an Array of integers.
      int[] A;

      // allocating memory for 5 integers.
      A = new int[5];

      // initialize the elements of array-A
      A [1] = 20;
      A [2] = 30;
      A [3] = 40;
      A [4] = 50;

      // accessing the elements using for loop ----------- (1)
      Console.Write("For loop :");
      for (int i = 0; i < A.Length; i++)  // where A.Length returns the size (length) of array
         Console.Write(" " + A [i]);

      Console.WriteLine("");
      Console.Write("For-each loop :");  // using for-each loop --------- (2)
      foreach(int i in A)
         Console.Write(" " + i);

      Console.WriteLine("");
      Console.Write("while loop :");  // using while loop ----------- (3)
      int j = 0;
      while (j < A.Length)
       {
         Console.Write(" " + A [j]);
         j++;
        }

      Console.WriteLine("");
```

```
        Console.Write("Do-while loop :");  // using do-while loop ------- (4)
        int k = 0;
        do
        {
           Console.Write(" " + A [k]);
           k++;
        } while (k < A.Length);
    }
}
}
```

Output:

```
For loop: 10 20 30 40 50
For-each loop: 10 20 30 40 50
while loop: 10 20 30 40 50
Do-while loop: 10 20 30 40 50
```

Other Examples:

Example1: Write a C# program to creating an array of the string as week days, store day values in the weekdays, and prints each value.

```
using System;
namespace Weekdays {
 Wclass WD {
   public static void Main()
   {  // declares an 1D Array of string.
       string[] weekDays;
       // allocating memory for days.
       weekDays = new string[] {"Sun", "Mon", "Tue", "Wed",
                               "Thu", "Fri", "Sat"};
       // Displaying Elements of array
       foreach(string day in weekDays)
           Console.Write(day + " ");
   }
}
}
```

Output :

```
Sun Mon Tue Wed Thu Fri Sat
```

Example2: The following example demonstrates the above-mentioned concepts **declaration**, **assignment**, and **accessing** arrays:

```
using System;
namespace ArrayApplication
{
        class MyArray
        {
                static void Main(string[] args)
                {
                        int [] n = new int[10]; /* n is an array of 10 integers */
                        int i,j;

                        /* initialize elements of array n */
                        for ( i = 0; i < 10; i++ )
                        {
                                n[ i ] = i + 100;
                        }
                        /* output each array element's value */
                        for (j = 0; j < 10; j++ )
                        {
                           Console.WriteLine("Element[{0}] = {1}", j, n[j]);
                        }
                        Console.ReadKey();
                }
        }
}
```

When the above code is compiled and executed, it produces the following result:

```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

<u>Example3:</u> Write a C# program to read an array with 10 integer numbers and find the summation of the total numbers of the array?

```
 Using System;
 // SumArray.cs
// Computing the sum of the elements in an array.
  class SumArray
   {
     // main entry point for application
     static void Main( string[] args )
      {
      int[] a; // declare reference to an array
      a = new int[ 10 ];
               for (int i = 0; i < a.Length; i++)
                    a[i] = Int32.Parse(Console.ReadLine());
        int total = 0;
        for ( int i = 0; i < a.Length; i++ )
        total += a[ i ];
 Console.WriteLine( "Total of array elements: " + total );
 } // end Main
 } // end class SumArray
```

<u>Example4 :</u> Write a C# program to read an array with 10 integer numbers and sorting this array ?

```
using System;
//  BubbleSorters
// Sorting an array's values into ascending order.
class SortArray
  {
      static void Main( string[] args )
```

```
     {
     int temp1;
      int[] x; // declare reference to an array
      x = new int[ 10 ]; // dynamically allocate array and set
             Console.WriteLine("input Array  x :");
              for (int i = 0; i < x.Length; i++)
                    x[i] = Int32.Parse(Console.ReadLine());
 // bubble sort
           for ( int pass = 0; pass < x.Length ; pass++ )
       for ( int i = 0; i < x.Length-1 ; i++ )
              if ( x [ i] >  x[i+1] )
                 {
                     temp1= x[i];
                     x[i ] = x[i+1];
                     x[i+1] = temp1;
                 }
               Console.WriteLine("Sort  Array is  :");
                    for ( int i = 0; i < x.Length-1 ; i++ )
                      Console.WriteLine ( x[i]);
         Console.ReadKey();
    } // end Main
 } // end class SortArray
```

<u>Example5 </u>/Write C# program to read an array with 10 integer numbers and search the value of an element in this array?

```
using System;
// Fig. 34: LinerSearch.cs
// search
class SortArray
  {
      static void Main( string[] args )
      {
      int[] x; // declare reference to an array
```

```
    x = new int[ 10 ]; // dynamically allocate array and set
    int value1 , i ;
      Console.WriteLine("input Array :");
       for ( i = 0; i < x.Length; i++)
             x[i] = Int32.Parse(Console.ReadLine());
      Console.WriteLine("value :");
         value1  = Int32.Parse (Console.ReadLine());
// search value ( linear search)
      for ( i = 0; i < x.Length  ; i++ )
         {
             if   (x [ i ] == value1 )
                      break;
         }
         if  (i  == x.Length )
                  Console.WriteLine( " not found");
         else
           Console.WriteLine("value  " + value1 + " is found in location "+i);
         Console.ReadKey();
   }
 }
```

<u>Example6</u> / Write a C# program to create an array of 10 integer numbers and find maximum and minimum number in the array and print array?

Sol:

```
{
      static void Main(string[] args)
      {
         int [ ] c;
         c = new int[10];
        int max = c[0] ;                    or int max=-9999;
         int min = c[0];
       Console.WriteLine("input Array :");
         for (int i = 0; i < c.Length; i++)
                         c[i] = Int32.Parse(Console.ReadLine());
       for (int i = 0; i < c.Length; i++)
         {
            if (c[i] >= max)
                max = c[i];
         }
      for (int i = 0; i < c.Length; i++)
         {
            if (c[i] <= min)
                min= c[i];
         }
   Console.WriteLine("max value=" + max);
   Console.WriteLine("min value=" + min);
    for (int i = 0; i < c.Length; i++)
        Console.WriteLine("c ["+i+"]  = "+c[i]);
       Console.ReadKey();
```

<u>Example</u>7 / Write a C# program to create an array (A) of 5 integer numbers and an array (B) of 5 integer numbers and the summation of A and B in an array (C) an and print arrays A,B,C?

```
Sol:
          int[ ] A = new int[5];
          int[ ] B = new int [5];
          int[ ] C = new int [5];
        Console.WriteLine(“input Array A :”);
          for (int i = 0; i < A.Length; i++)
             A[i] = Int32.Parse(Console.ReadLine());
        Console.WriteLine(“input Array  B:”);
          for (int i = 0; i < B.Length; i++)
             B[i] = Int32.Parse(Console.ReadLine());
          for (int i = 0; i < C.Length; i++)
             C[i] = A[i] + B[i];
          for (int i = 0; i < C.Length; i++)
          Console.WriteLine("A[" + i + "]  = " +A[i]+ "\t B[" + i + "]  ="
                                 +B[i]+ "\t C[" + i + "] = " + C[i]);
          Console.ReadKey();
```

Home Works

1. Write a C# program to create an array of 10 integer numbers then:
   a- Find the maximum number with its location?
   b- Find the minimum number with its location?
2. Write a C# program to create an array of 10 integer numbers then Increment each number in odd position by 5?
3. Write a C# program to create an array of 10 integer numbers then Count the even and odd numbers in an array?
4. Write a C# program to create an array of 10 integer numbers then Find the frequency of a number entered by the user?

Multidimensional Arrays

The multi-dimensional array contains more than one row to store the values. C# supports multidimensional arrays. The simplest form of the multidimensional array is the two-dimensional array. It is also known as a **Rectangular Array** in **C#** because it's each row length is same. It can be a **2D-array** or **3D-array** or more. To storing and accessing the values of the array, one required the **nested loop**.

You can declare a 2-dimensional array of strings as:

```
string [,] names;
```

And a 3-dimensional array of int variables as:

```
int [ , , ] m;
```

2. Two-Dimensional (2-D) Arrays

The simplest form of the multidimensional array is the 2-dimensional array. A 2-dimensional array is a list of one-dimensional arrays. A 2-dimensional array can be thought of as a **table**, which has **x number of rows** and **y number of columns**. Following is a 2-dimensional array, which contains 3 rows and 4 columns:

	Column0	**Column1**	**Column2**	**Column3**
Row 0	**a[0,0]**	**a[0,1]**	**a[0,2]**	**a[0,3]**
Row 1	**a[1,0]**	**a[1,1]**	**a[1,2]**	**a[1,3]**
Row 2	**a[2,0]**	**a[2,1]**	**a[2,2]**	**a[2,3]**

Thus, every element in the array **a** is identified by an element name of the form **a[i , j]**, where **a** is the **name** of the array, and **i** and j are the subscripts that uniquely identify each element in array **a**.

Initializing Two-Dimensional Arrays

Multidimensional arrays may be **initialized** by specifying **bracketed** values for each **row**. The following array is with **3 rows** and **each row has 4 columns**.

```
int [,] a = int [3,4] = {
        {0, 1, 2, 3} , /* initializers for row indexed by 0 */
        {4, 5, 6, 7} , /* initializers for row indexed by 1 */
        {8, 9, 10, 11} /* initializers for row indexed by 2 */
                  };
```

Accessing Two-Dimensional Array Elements

An element in 2-dimensional array is accessed by using the subscripts. That is, row index and column index of the array. For example, to take **4th** element from the **3rd row** of the array **a**.

```
int val = a[2,3];
```

<u>Example1</u>: Write a C# of a 2-D array of size 5 x 2. Then print it?

```
using System;
namespace ArrayApplication
{
  class MyArray
          {
          static void Main(string[] args)
            { /* an array with 5 rows and 2 columns*/
              int[,] a = new int[5, 2] {{0,0}, {1,2}, {2,4}, {3,6}, {4,8} };
              int i, j;
             /* output each array element's value */
                   for (i = 0; i < 5; i++)
                   {
                     for (j = 0; j < 2; j++)
                            {
                               Console.WriteLine("a[{0},{1}] = {2}", i, j, a[i,j]);
                            }
                   }
             Console.ReadKey();
            }
           }
}
```

<u>Output:</u>

```
a[0,0]: 0
a[0,1]: 0
a[1,0]: 1
a[1,1]: 2
a[2,0]: 2
a[2,1]: 4
a[3,0]: 3
a[3,1]: 6
a[4,0]: 4
a[4,1]: 8
```

<u>Example2</u>: Write a C# program to create two multidimensional arrays of same size. Accept value from user and store them in first array. Then copy all the elements of first array are second array and print output.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace example2
{
    class Program
    {
        static void Main(string[] args)
        {
            int[,] arr1 = new int[3, 3];
            int[,] arr2 = new int[3, 3];
            int i, j;

            // Storing user input in arr1
            for (i = 0; i < 3; i++)
            {
                for (j = 0; j < 3; j++)
                {
                  Console.Write("Enter array value:\t");
                  arr1[i,j]=Convert.ToInt32(Console.ReadLine());
                }
            }
            //copying value of arr1 to arr2
            for (i = 0; i < 3; i++)
            {
                for (j = 0; j < 3; j++)
                {
                    arr2[i, j] = arr1[i, j];
                }
            }
```

```
           Console.WriteLine("\n\nElements of second array
                                    are:\n\n");
```

```
            //Printing the arr2 value
            for (i = 0; i < 3; i++)
            {
                Console.WriteLine();
                for (j = 0; j < 3; j++)
                {
                  Console.Write("\t{0}",arr2[i, j]);
                }
            }
            Console.ReadLine();
        }
    }
}
```

Length of Multidimensional Arrays

Each dimension of a multidimensional array has its own **length**, which can be accessed during the execution of the program. We can get the number of the **rows** and **columns** of a two-dimensional array by using:

```
ArrayName.GetLength(0) // to get the no. of rows
ArrayNam.GetLength(1)  // to get the no. of columns per row
```

Example3: write a C# program to declare and initialize a 2D array called **matrix** of size 2 x 4, then print it?

```
// Declare and initialize a matrix of size 2 x 4
int[,] matrix =
 {
   {1, 2, 3, 4}, // row 0 values
   {5, 6, 7, 8}, // row 1 value
 };
// Print the matrix on the console
for (int row = 0; row < matrix.GetLength(0); row++)
  {
    for (int col = 0; col < matrix.GetLength(1); col++)
      {
          Console.Write(matrix[row, col]);
       }
    Console.WriteLine();
   }
```

Output

```
1 2 3 4
5 6 7 8
```

Example4: read a two-dimensional array from the console. First, we read the values (lengths) of the two-dimensions and then by using two nested loops we assign the value of each element, and in the end we print out the values of the array.

```
Console.Write("Enter the number of the rows: ");
int rows = int.Parse(Console.ReadLine());
Console.Write("Enter the number of the columns: ");
int cols = int.Parse(Console.ReadLine());
int[,] matrix = new int[rows, cols];
Console.WriteLine("Enter the cells of the matrix:");
for (int row = 0; row < rows; row++)
        {
            for (int col = 0; col < cols; col++)
              { Console.Write("matrix[{0},{1}] = ",row, col);
                matrix[row, col] = int.Parse(Console.ReadLine());
              }
        }
        for (int row = 0; row < matrix.GetLength(0); row++)
        {
           for (int col = 0; col < matrix.GetLength(1); col++)
              {
                Console.Write(" " + matrix[row, col]);
              }
           Console.WriteLine(); }
```

The **output** of above program is:

```
Enter the number of the rows: 3
Enter the number of the columns: 2
Enter the cells of the matrix:
matrix[0,0] = 2
matrix[0,1] = 3
matrix[1,0] = 5
matrix[1,1] = 10
matrix[2,0] = 8
matrix[2,1] = 9
2 3
5 10
8 9
```

Example5: Write a C# program to add the diagonal of a 2-Dimensional array?

```
using System;
namespace Add_Diagonal
{
   public class Program
   {
      public static void Main(string[] args)
      {
         int[,] num= {
            {22,50,11, 2, 49},
            {92,63,12,64,37},
            {75,23,64,12,99},
            {21,25,71,69,39},
            {19,39,58,28,83}};

            //Getting Row and Column Length
            int rowlength = num.GetLength(0);
            int columnlength = num.GetLength(1);
            int total=0;
            Console.Write("Diagonals are : ");
            for(int row=0; row<rowlength; row++)
            {
               for(int column=0; column<columnlength; column++)
               {
                  if(row==column)
                  {
                     Console.Write(num[row,column] + " ");
                     total += num[row,column];
                  }
               }
            }
            Console.WriteLine(": Sum : " + total);
            Console.ReadLine();
      }
   }
 }
```

Output:

```
Diagonals are : 22 63 64 69 83 : Sum : 301
```

Example6: Write a C# program to create an array of two dimension [2,3] of integer numbers and print its elements ?

sol

```
int[,] A = new int[2, 3];
for (int i = 0; i < 2; i++)
  {
     for (int j = 0; j < 3; j++)
     {
        A[i, j] = Int32.Parse(Console.ReadLine());
     }
  }
  for (int i = 0; i < 2; i++)
  {
     for (int j = 0; j < 3; j++)
     {
 Console.WriteLine("A[" + i + ","+j+"]  = " + A[i, j]);
     }
  }
           Console.ReadKey();
```

Home Works

1) Write a C# program to create an array of two dimension [4,3] of integer numbers and find:

1- Print the average of its elements.

2- The average of odd and even numbers.

2) Write a C# program to create an array of two dimension [5,3] of real numbers and find The maximum and minimum numbers and their locations.

3) Write a C# program to create an array of two dimension [3,4] of real numbers and find :

a. the sum of each row.
b. The sum of each column.

4) Write a C# program to create an array of two dimension [4,3] of integer numbers and Exchange row 2 with row 4.

5) Write a C# program to create an array of two dimension [4,3] of integer numbers and Exchange col 1 with col 2.

6) Write a C# program to create an array of two dimension [4,3] of integer numbers and find :

   a) Reverse column 3.
   b) The count of numbers divided by 5.

7) Write a C# program to create an array of two dimension [5,5] of integer numbers and find:

   a) The sum of the elements on main diagonal.
   b) The sum of the elements on secondary diagonal.

8) Write a C# program to create an array of two dimension [5,5] of integer numbers and find:

   a) The sum of elements on triangle upper main diagonal.
   b) The sum of elements on triangle lower main diagonal.

9) Write a C# program to create an array of two dimension [5,5] of integer numbers and Exchange the elements of two diagonal.

10) Write a C# program to create an array of two dimension [5,5] of integer numbers and Convert the two dimensional array into one dimensional array.

Passing arrays as arguments in C#

An **array** is a collection of similar type variables which are referred to by a common name. In C#, arrays are the **reference types** so it can be passed as **arguments** to the **method**. A method can modify the value of the elements of the array. Both single dimensional and multidimensional arrays can be passed as an argument to the methods.

1. Passing 1-D Arrays as arguments to methods

One can pass the 1-D arrays to a method. There are various options like:

- First, you declare and initialize the array separately then pass it to the method.
- Second, you can declare, initialize and pass the array to the method in a single line of code.

Example 1: Declaring and initializing array first and then pass it to the method as an argument.

```
// taking an integer array
// declaring and initializing the array
int[] arr = {1, 2, 3, 4};

// passing the array as an argument to the method
// Result is the method name
Result(arr);
```

Example 2: Declaring, initializing and passing the array to the method in a single line of code.

```
Result(new int[] {1, 2, 3, 4});
```

Example3: In the program below, we are passing the 1-D array **arr** to the method **Result**. The method is static and it will print the array elements which are passed to it.

```
// C# program for passing the 1-D
// array to method as argument
using System;

class GFG {

   // declaring a method
   static void Result(int[] arr) {

      // displaying the array elements
      for(int i = 0; i < arr.Length; i++)
      {
         Console.WriteLine("Array Element: "+arr[i]);
      }

   }

   // Main method
   public static void Main() {

      // declaring an array
      // and intializing it
      int[] arr = {1, 2, 3, 4, 5};

      // callling the method
      Result(arr);
   }
}
```

Output:

```
Array Element: 1
Array Element: 2
Array Element: 3
Array Element: 4
Array Element: 5
```

<u>Example4:</u> Write C# program to create an array (A) of 10 integer numbers and an array(B) of 10 integer numbers and find the summation of array A and array B in an array C and print array A, B and C; using methods?

```
// find the summation of array A and array B  in an array C and print array A, B and C
class Program1
{
    static void readarray(int[] x)
    {
        for (int i = 0; i < x.Length; i++)
        {
            Console.Write("array[" + i + "]=  ");
            x[i] = Int32.Parse(Console.ReadLine());
        }
    }
    static void addarray(int[] a, int[] b, int[] c)
    {
        for (int i = 0; i < c.Length; i++)
            c[i] = a[i] + b[i];
    }
    static void printarray(int[] a, int [] b , int [] c )
    {
        for (int i = 0; i < c.Length; i++)
            Console.WriteLine(a[i] + "\t" + b[i] +"\t" + c[i]);
    }
    static void Main(string[] args)
    {
        int[] a = new int[5];
        int[] b = new int[5];
        int[] c = new int[5];
        Console.WriteLine("input array A :");
        readarray(a);
        Console.WriteLine("input array B :");
        readarray(b);
        addarray(a, b, c);
        Console.WriteLine("print array A , B ,C \n");
        Console.WriteLine("A\tB\tC \n --------------------------------------");
        printarray(a,b,c);
        Console.ReadKey();
    }
}
```

<u>Example5:</u> Write a C# program to create array named **table** of 9 real numbers, and search the value **elt** in array and print index of numbers?

```
// using instance of the class (Non –static)
    // search  real number
class Program2
    {
        static void Main(string[] args)
        {
            double[] table = { 12.5, -5.2, 700, 80.3, -6, 1000.23, 4, 78.2, -20.2 };
            double elt = 80.3;
            int i;
            for (i = 0; i < 8; i++)
                if (elt == table[i]) break;
            locaValue(i, elt);
            Console.ReadKey();
        }
        static void locaValue(int ord, double val)
        {
            if (ord == 8)
                Console.WriteLine("Value : " + val + "Not Found.");
            else
                Console.WriteLine("Value : " + val + " Order :" + ord);
        }
    }
```

<u>Example6:</u> Write a C# program using methods to create array named table of n integer numbers, and find the following using methods:

1- the average of even numbers .
2- the count the positive even numbers and count positive odd numbers .
3- the maximum number with its location
4- increment each number in odd position by 10 and each number in even position by 5.
5- Print array .

```
class Program4
    {
        //read  array
        static void ReadArr(int[] list, ref int n)
        {
            for (int i = 0; i < n; i++)
            {
                Console.Write(" input number " + i + " = ");
                list[i] = Int32.Parse(Console.ReadLine());
            }
        }
        // the average of  even numbers .
        static double AvEven(int[] list)
        {
            int sum = 0, count = 0;
            double av;
            for (int i = 0; i < list.Length; i++)
            {
                if (list[i] % 2 == 0)
                {
                    sum += list[i];
                    count++;
                }
            }
            av = (double)sum / count;
            return av;
        }
        //   count the positive even numbers  and count positive odd numbers
        static void CountValue(int[] list, ref int n, out int counto, out int counte)
        {
            counto = 0;
            counte = 0;
            for (int i = 0; i < n; i++)
            {
                if ((list[i] % 2 == 0) && (list[i] > 0))
                    counte++;
                if ((list[i] % 2 != 0) && (list[i] < 0))
                    counto++;
            }
        }
        //the maximum number with its location
        static int MVi(int[] a, ref int n, out int maxIndex)
        {
            int maxVal = a[0];
            maxIndex = 0;
            for (int i = 1; i < n; i++)
                if (a[i] > maxVal)
                {
                    maxVal = a[i];
                    maxIndex = i;
                }
            return maxVal;
        }
       //increment each number in odd position by 10 and each number in even position by 5
        static void IncrementArr(int[] list)
        {
            for (int i = 0; i < list.Length; i++)
            {
                if (i % 2 == 0)
                    list[i] += 5;
                else
                    list[i] += 10;
            }
        }
        //Print  array
        static void PrintArr(int[] list)
        {
            for (int i = 0; i < list.Length; i++)
                Console.WriteLine(" list [  " + i + " ] = " + list[i]);
        }
        static void Main(string[] args)
        {
            int n;
            Console.WriteLine(" inpur size of array : ");
            n = Int32.Parse(Console.ReadLine());
            int[] list = new int[n];
            int index, countodd, counteven;
            double av;
            // Call Read array-Method
            ReadArr(list, ref n);
            // Call the average of  even numbers-Method
            av = AvEven(list);
            Console.WriteLine("the average of  even numbers  is " + av);
          // Call of count the positive even numbers  and count positive odd numbers-Method
            CountValue(list, ref n, out countodd, out counteven);
            Console.WriteLine("\n count the positive even numbers  is " + counteven);
            Console.WriteLine("\n count positive odd numbers is " + countodd);
            //Call Method of the maximum number with its location
        Console.WriteLine("\nThe maximum value in myArray is " + MVi(list, ref n, out index));
        Console.WriteLine("\nThe first occurrence of this value is at element " + (index + 1));
         // Call Method of increment each number in odd position by 10 and each number in
    even  position by 5
            IncrementArr(list);
            // Call Print  array Method
      Console.WriteLine("\nprint the array after increment each number in odd position by 10 and
    each number in even position by 5 \n");
            PrintArr(list);
            Console.ReadKey();
        }
      }
```

H.W : Write a C# program to create array **table** of 10 string values , and search the value **elt** in array and print index of string?

2. Passing Multi-Dimensional Arrays as arguments to Methods

You can also pass the **multidimensional** arrays to a method. There are various options like passing one-dimensional arrays.

Example1: Declaring and initializing array first and then pass it to the method as an argument.

```
// declaring and initializing the 2-D array
int[,] arr = { {1, 2, 3, 4}, {5, 6, 7, 8} };
 // passing the array as an argument to the method
// Result is the method name
Result(arr);
```

Example2: Declaring, initializing and passing the 2-D array to the method in a single line of code.

```
Result(new int[,] { {1, 2, 3, 4}, {5, 6, 7, 8} });
```

Example3:

Write a C# program for passing a 2-D array ***arr*** to the method transpose which gave the transpose of the matrix. **GetLength()** method is used for the count the total number of elements in a particular dimension.

```
// C# program for finding the transpose
// of matrix(2-D array) by using array
// as function arguments
using System;

class GFG {

   // temp is used as temporary variable
   static int temp = 0;

     // passing 2-D array 'arr' as argument
   // to find the transpose of matrix
```

```
static void transpose(int[, ] arr)
{
   for (int i = 0; i < arr.GetLength(0); i++) {
      for (int j = i; j < arr.GetLength(1); j++) {
         temp = arr[i, j];
         arr[i, j] = arr[j, i];
         arr[j, i] = temp;
      }
   }
}

// to display the transposed matrix
static void displayresult(int[, ] arr)
{

   Console.WriteLine("Matrix After Transpose: ");

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

// Main Method
static public void Main()
{
   // declaration of an 2-d array
   int[, ] arr;

   // initialzing 2-D array
   // matrix of 4 rows and 4 colums
   arr = new int[4, 4]{ { 1, 2, 3, 4},
                     { 5, 6, 7, 8},
                     {9, 10, 11, 12},
                     {13, 14, 15, 16} };

   Console.WriteLine("Matrix Before Transpose: ");

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

   Console.WriteLine();

   // calling transpose method
   transpose(arr);
```

```
        // calling displayresult method
        // to display the result
        displayresult(arr);
    }
}
```

Output:

```
Matrix Before Transpose:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Matrix After Transpose:
1 5 9 13
2 6 10 14
3 7 11 15
4 8 12 16
```

Example4 : Write a C# program using method to create an array of two dimension [4,3] of integer numbers and find:

1- Print the average of its elements.

2- The average of odd and even numbers.

3- The maximum and minimum numbers and their locations.

4- the sum of each row.

5- The sum of each column.

6- Exchange row 1 with row 3.

7- Reverse column 2.

8- The count of numbers divided by 5.

9- Exchange col 1 with col 2.

```
class Program7
 {
    static void readArr(int[,] a)
    {
     for (int i = 0; i < 4; i++)
     for (int j = 0; j < 3; j++)
      {
        Console.Write("a[" + i + " , "+j+ " ]= ");
        a[i, j] = Int32.Parse(Console.ReadLine());
      }
    }
  static void printArr(int[,] a)
   {
    for (int i = 0; i < 4; i++)
     {
       for (int j = 0; j < 3; j++)
       Console.Write(a[i, j] + "  ");
     Console.WriteLine();
     }
    }  //----------------------------------End of Print-Method-----------------
  static double average1(int[,] a)
   {
     int sum = 0;
     for (int i = 0; i < 4; i++)
     for (int j = 0; j < 3; j++)
     sum += a[i, j];
     return (double) sum/12;    // return sum/12.0;
    }//----------------------------------------End of average method-------------
```

```
double aveOddandEven(int[,] a, out double avodd)
 {
   int sumo = 0 , sume =0 , codd=0 , ceven=0;
   for (int i = 0; i < 4; i++)
   for (int j = 0; j < 3; j++)
     if (a[i,j] %2 == 0)
       {
          sume+=a[i,j];
          ceven++ ;
        }
     else
     {
          sumo+=a[i,j];
          codd++ ;
     }
  avodd= (double) sumo/codd;
  return (double)sume / ceven;
 } //------------------------------END of aveOddandEven Method-----------------------
 int maxNumber(int[,] a, out int indexi , out int indexj)
   {
      int max1= a[0,0];
      indexi=0;
       indexj=0;
       for (int i = 0; i < 4; i++)
       for (int j = 0; j < 3; j++)
          if (a[i, j] > max1)
           {
            max1= a[i,j];

            indexi=i;

            indexj=j;

         }

    return max1;

  }  //--------------------------------End of Max.Number Method------------------

static int minNumber(int[,] a, out int indexi, out int indexj)

  {

    int min1 = a[0, 0];

    indexi = 0;

    indexj = 0;

    for (int i = 0; i < 4; i++)

    for (int j = 0; j < 3; j++)

        if (a[i, j] < min1)

           {

             min1 = a[i, j];

             indexi = i;

             indexj = j;

           }

   return min1;

}  //-------------------------- End of Min. Number Method ------------------------

static int sumRow(int [,] a,  ref int i)

 {

   int sum=0;

   for (int j = 0; j < 3; j++)

        sum += a[i, j];

  return sum ;

 } // -------------------------- End of SumRow Method --------------------------------
```

```
static int sumColumn(int[,] a, ref int j)
{
  int sum = 0;
  for (int i = 0; i < 4; i++)
  sum += a[i, j];
 return sum;
} // -------------------------- End of SumColumn Method -----------------------------
void exchangeRow(int[,] a)
{
   for (int j = 0; j < 3; j++)
     {
         int t;
         t = a[1, j];
         a[1, j] = a[3, j];
         a[3, j] = t;
      }
} // -------------------------- End of ExchangeRow Method-----------------------------
static void reverseCol(int[,] a)
{
   for (int i = 0; i < 2; i++)
     {
        int t;
        t = a[i,2];
        a[i, 2] = a[3-i, 2];
        a[3-i, 2] = t;
     }
} // -------------------------- End of Revers Column Method -----------------------------
```

```
static int countDiv5(int[,] a)
 {
    int count = 0;
  for (int i = 0; i < 4; i++)
  for (int j = 0; j < 3; j++)
      if (a[i, j] % 5 == 0)
         count++;
 return count;
 } // -------------------------- End of CountDiv 5 Method -----------------------------
 static void exchangecol(int[,] a)
 {
  for (int i = 0; i < 4; i++)
   {
    int t;
    t = a[i, 1];
    a[i, 1] = a[i, 2];
    a[i, 2] = t;
   }
  } // -------------------------- End of ExchangeCol Method --------------------------------
  static void Main(string[] args)
  {
    Program7 p = new Program7();
    int[,] a = new int[4, 3];
    int indexi,indexj , maxnum;
    double avodd;
    // read array a
```

```
     Console.WriteLine("read array a \n ----------------------------\n");
     readArr(a);
     // print array a
     Console.WriteLine("print array a \n ----------------------------\n");
     printArr(a);
     // print the average of its element
    Console.WriteLine("\n the average of its elements =  " + average1(a));
    // find the average of odd and even numbers
    Console.WriteLine("\n the average of even number =  " + p.aveOddandEven(a,out
avodd));
    Console.WriteLine("\n the average of even number =  " + avodd);
    // find the maximum number and its location
    maxnum=p.maxNumber( a, out  indexi ,  out indexj);
   Console.WriteLine("\n the maximum number =  " + maxnum + " the index i  = " + indexi
+ "   the index j  = " + indexj);
   //find the minimum number and its  location
   Console.Write("\n the minimum number =  "+minNumber(a, out  indexi, out indexj));
   Console.WriteLine(" \n  the index i  = " + indexi + "   the index j  = " + indexj);
   // find the sum of each row
   for (int i = 0; i < 4; i++)
      Console.WriteLine("\nthe summation of  row "+  i + " =  "+ sumRow(a, ref i));
   // find the sum of each column
   for (int j = 0; j < 3; j++)
      Console.WriteLine("\n the summation of  column " + j + " =  " + sumColumn(a, ref j));
    // exchange row 1 with row 3
    p.exchangeRow( a);
   Console.WriteLine("\n print array a after exchange row 1 with row 3  \n ------------\n");
   printArr(a);
   //      Reverse column 2
   reverseCol(a);
    Console.WriteLine("\n print array a after Reverse column 2 \n ----------------------\n");
    printArr(a);
   // the count of numbers divided by 5
   int count1 = countDiv5(a);
   Console.WriteLine("\n the count of numbers divided by 5 = " +count1);
  // Exchange column 1 with column 2
  exchangecol( a);
  Console.WriteLine("\n print array a after Exchange column 1 with column 2  \n --------\n");
  printArr(a);
  Console.ReadKey();
 }
} // ------------------------- End of the program -------------------------------
```

Home Works

1) Write a C# program **using Method** to create an array of two dimension [5,5] of integer numbers and find:

a) The sum of elements on triangle upper main diagonal.
b) The sum of elements on triangle lower main diagonal.

2) Write a C# program using Method to create an array of two dimension [5,5] of integer numbers and Exchange the elements of two diagonal.

3) Write a C# program **using Method** to create an array of two dimension [5,5] of integer numbers and Convert the two dimensional array into one dimensional array.

4) Write a C# program **using Method** to create an array of two dimension [5,5] of integer numbers and find:

a) The sum of elements on triangle upper main diagonal.
b) The sum of elements on triangle lower main diagonal.