PROGRAMMING IN C#

Module 3: Statements and Operators
Module 4: Programming Constructs
Module 5: Array

Lab Guide for Lab2

Session Objectives

At the end of this session, you will able to understand:

- *Statements & Operators*
- *Programming Construct*
- *Array*

Part 1 – Getting started (30 minutes)

Exercise 1: foreach statement

Step 1: Add a console based project ‘ForEach’ to the solution

Step 2: Right click on project ForEach -> set as Startup project

Step 3: Rename the class file ‘Program.cs’ to ‘ForEach.cs’

Step 4: Replace the code in ‘ForEach.cs’ with the given code

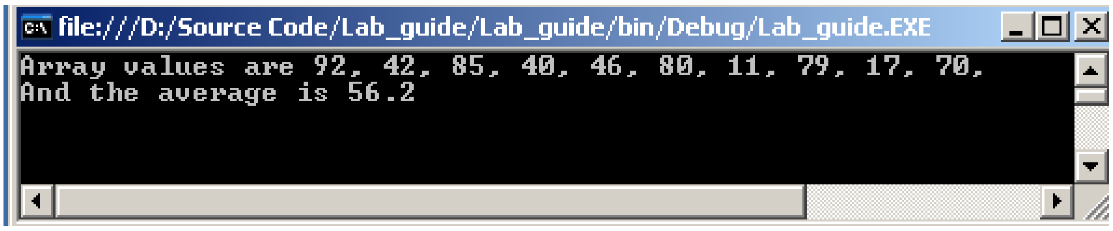
```
using System;
class ForEach
{
        static void Main(string[] args)
        {
            DateTime now = DateTime.Now;
            Random rand = new Random((int)now.Millisecond);
            int[] Arr = new int[10];
            for (int x = 0; x < Arr.Length; ++x)
            {
                Arr[x] = rand.Next() % 100;
            }
            int Total = 0;
            Console.Write("Array values are ");
            foreach (int val in Arr)
            {
                Total += val;
                Console.Write(val + ", ");
            }
            Console.WriteLine("\nAnd the average is {0,0:F1}",
                                (double)Total / (double)Arr.Length);
            Console.ReadLine();
        }
}
```

Step 5: Select menu File -> Save to save the file

Step 6: Select Build -> Build 'ForEach' option to build the project

Step 7: Select Debug -> Start without Debugging to execute the program

The output of program as following


file:///D:/Source Code/Lab_guide/Lab_guide/bin/Debug/Lab_guide.EXE

```
Array values are 92, 42, 85, 40, 46, 80, 11, 79, 17, 70,
And the average is 56.2
```

Exercise 2: Array of Object

Step 1: Add a console based project 'ObjectArray' to the solution

Step 2: Right click on project ObjectArray -> set as Startup project

Step 3: Rename the class file 'Program.cs' to 'ObjectArray.cs'

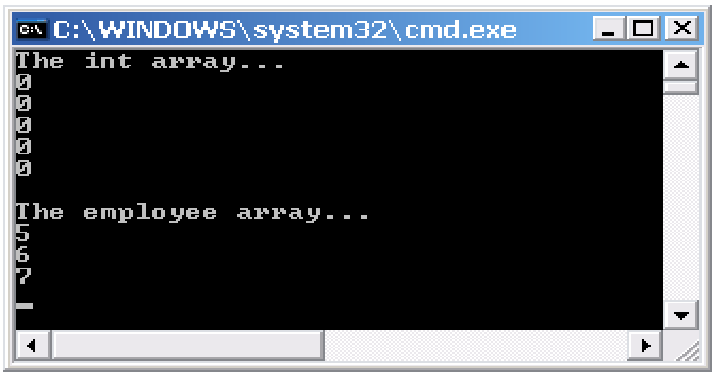
```
using System;
class Employee
{
        private int empID;

        // constructor
        public Employee(int empID)
        {
            this.empID = empID;
        }
        public override string ToString()
        {
            return empID.ToString();
        }
}
class ObjectArray
{
        public void Run()
        {
            int[] intArray;
            Employee[] empArray;
            intArray = new int[5];
            empArray = new Employee[3];

            // populate the array
            for (int i = 0; i < empArray.Length; i++)
            {
                empArray[i] = new Employee(i + 5);
            }

            Console.WriteLine("The int array...");
            for (int i = 0; i < intArray.Length; i++)
            {
                Console.WriteLine(intArray[i].ToString());
            }

            Console.WriteLine("\nThe employee array...");
            for (int i = 0; i < empArray.Length; i++)
            {
                Console.WriteLine(empArray[i].ToString());
            }
        }
        static void Main(string[] args)
        {
            ObjectArray arr = new ObjectArray();
            arr.Run();
            Console.ReadLine();
        }
}
```

Step 5: Select menu File -> Save to save the file

Step 6: Select Build -> Build 'ObjectArray' option to build the project

Step 7: Select Debug -> Start without Debugging to execute the program

The output of program as following



```
The int array...
0
0
0
0
0

The employee array...
5
6
7
_
```

Exercise 2: `Array Class`

Step 1: Add a console based project 'ArrayClass' to the solution

Step 2: Right click on project ArrayClass -> set as Startup project

Step 3: Rename the class file 'Program.cs' to 'ArrayClass.cs'

Step 4: Replace the code in 'ArrayClass.cs' with the given code

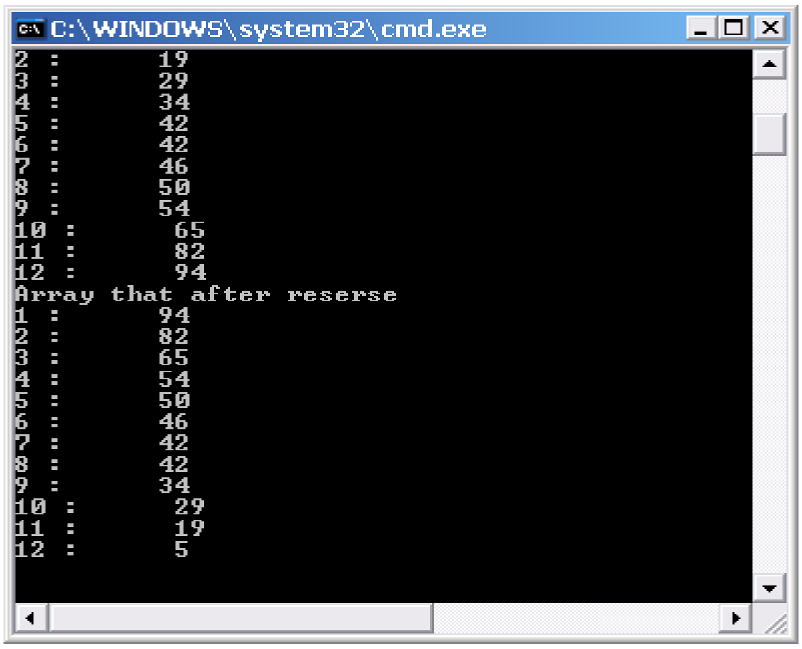
```
using System;
class ArrayClass
    {
        static void Main(string[] args)
        {
            int[] Arr = new int[12] { 29, 82, 42, 46, 54, 65, 50,
42, 5, 94, 19, 34 };
            Console.WriteLine("The first occurrence of 42 is at
index "
                              + Array.IndexOf(Arr, 42));
            Console.WriteLine("The last occurrence of 42 is at index
"
                              + Array.LastIndexOf(Arr, 42));
            int x = 0;
            while ((x = Array.IndexOf(Arr, 42, x)) >= 0)
            {
                Console.WriteLine("42 found at index " + x);
                ++x;
            }
            x = Arr.Length - 1;
            while ((x = Array.LastIndexOf(Arr, 42, x)) >= 0)
            {
                Console.WriteLine("42 found at index " + x);
                --x;
            }

            Console.WriteLine("Array that before sorted");
            for (int i = 0; i < Arr.Length; i++)
            {
                Console.WriteLine("{0} :      {1}", i + 1, Arr[i]);
            }
            Array.Sort(Arr);
            Console.WriteLine("Array that after sorted");
            for (int i = 0; i < Arr.Length; i++)
            {
                Console.WriteLine("{0} :      {1}", i + 1, Arr[i]);
            }
            Array.Reverse(Arr);
            Console.WriteLine("Array that after reserse");
            for (int i = 0; i < Arr.Length; i++)
            {
                Console.WriteLine("{0} :      {1}", i + 1, Arr[i]);
            }
            Console.ReadLine();
        }
}
```

Step 5: Select menu File -> Save to save the file

Step 6: Select Build -> Build ‘ArrayClass’ option to build the project

Step 7: Select Debug -> Start without Debugging to execute the program

The output of program as following



```
2 :       19
3 :       29
4 :       34
5 :       42
6 :       42
7 :       46
8 :       50
9 :       54
10 :       65
11 :       82
12 :       94
Array that after reserse
1 :       94
2 :       82
3 :       65
4 :       54
5 :       50
6 :       46
7 :       42
8 :       42
9 :       34
10 :       29
11 :       19
12 :       5
```

Part 2 – Workshops (30 minutes)

- Quickly look at workshops of Module 3, Module 4 and Module 5.
- Try to compile, run and observe the output of sample code provided for related workshops. Discuss with your class-mate and your instructor if needed.

Part 3 – Lab Assignment (60 minutes)

Do the assignment for Module 3, Module 4 and Module 7 carefully. Discuss with your class-mates and your instructor if needed.

Part 4 – Do it yourself

SIN Validator

Background

A Canadian SIN has nine digits. The right-most digit is a check digit that enables validation. For the whole number to be a valid SIN, a weighted sum of the other digits plus this check digit must be divisible by 10.

To obtain the weighted sum, take the digit to the left of the check digit and then every second digit leftwards. Add each of these digits to itself. Then, add each digit of each sum to form the weighted sum of the even positioned digits. Add each of the remaining SIN digits (except the check digit) to form the sum of the odd positioned digits. Add the two sums and subtract the next highest number ending in zero from their total. If this number is the check digit, the whole number is a valid SIN; otherwise, the whole number is not a valid SIN.

Consider the following example:

```
 SIN  193 456 787
                |
 check digit is 7
 add first set of alternates to themselves
   9  4  6  8
   9  4  6  8
  ___________
  18  8 12 16
 add the digits of each sum 1+8+8+1+2+1+6 = 27
 add the other alternates   1+3+5+7       = 16
 total                                    = 43
 Next highest integer multiple of 10      = 50
 Difference                               =  7
 Matches the check digit, therefore this SIN is
valid
```

Specifications
Design a program that validates a Canadian Social Insurance Number (SIN). Your program keeps accepting a whole number and determining if that whole number is a valid SIN. Your program terminates when the user enters 0 as the whole number. The output from your program looks something like:

```
 SIN Validator
 =============
 SIN (0 to quit): 193456787
 This is a valid SIN.
 SIN (0 to quit): 193456788
 This is not a valid SIN.
 SIN (0 to quit): 0
 Have a Nice Day!
```