Usman Institute of Technology

Department of Computer Science

Course Code: SE319

Course Title: Web Engineering

Fall 2022

Lab 04

Objective:

This experiment introduces the students about the fundamentals concept of C# to begin with Back End Programming Language.

Student Information

Student Name	
Student ID	
Date	

Assessment

Marks Obtained	
Remarks	
Signature	

Usman Institute of Technology
Department of Computer Science
SE319 – Web Engineering

Lab 04

Instructions

- Come to the lab in time. Students who are late more than 15 minutes, will not be allowed to attend the lab.
- Students have to perform the examples and exercises by themselves.
- Raise your hand if you face any difficulty in understanding and solving the examples or exercises.
- Lab work must be submitted on or before the submission date.

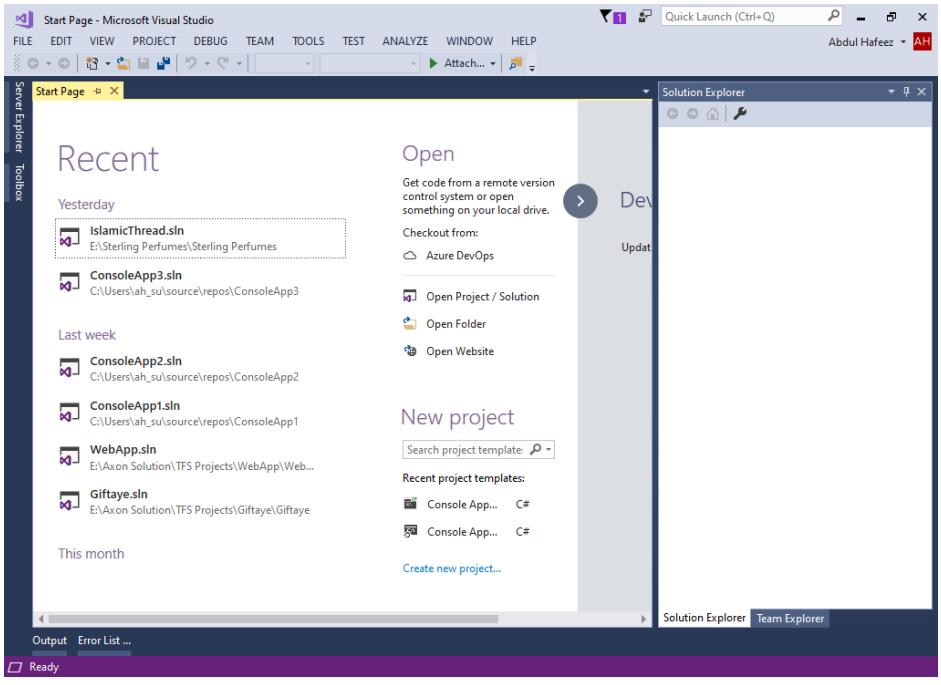
1. Objective

This experiment introduces the students about the fundamentals concept of C# to begin with Back End Programming Language.

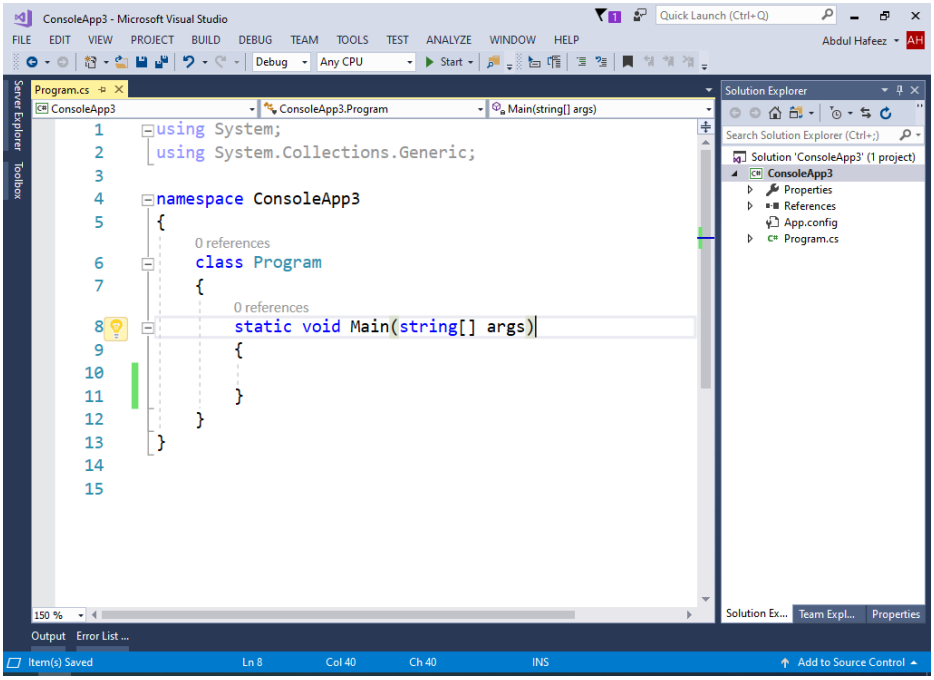
2. Overview

Microsoft has made a tool called Visual Studio, which is a great place to write programs. It comprises the compiler, along with an integrated editor, and debugger. It is provided in a number of versions with different feature sets.

Start Page



New Project



Project Structure

- Solution file, project file, program files
- Modules have a file suffix of .cs
- C# has a "special" procedure named **Main**() i.e. The Main entry point for the Application

In C#, all statements end with a semicolon**;** statements can span multiple lines

Getting Output

- To compile the program, click **Build** on the menu bar, and then click **Build Solution**
- As an alternative, you can press **F6**
- Click **Debug** on the menu bar and then click **Start Without Debugging Shortcut (Ctrl + F5)**
- Steps for viewing a program output
- Compile **source code** into **intermediate language (IL)**
- C# **just in time (JIT)** compiler translates the intermediate code into executable statements

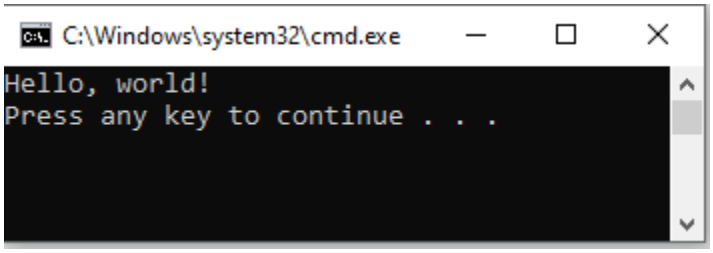


Figure 1: Output of the Hello application in Visual Studio

Modifying Your Project

So in the center, you should currently see some text that looks identical to this

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace HelloWorld
{
    class Program
    {
         static void Main(string[] args)
        {
             // Your Code goes here //
        }
    }
}
```

In a minute, we'll discuss what all of that does, but for now, let's go ahead and make our first change— something that will print out the message "Hello World!". Right in the middle of that code, you'll see three lines that say static void Main(string[] args) then a starting curly brace ('{') and a closing curly brace ('}'). We want to add our new code right between the two curly braces. Here's the line we want to add:

```
Console.WriteLine("Hello World!");
```

So now our program's full code should look like this:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

3. Creating Variables

We'll discuss the different types of C# variables in more detail in a minute. First, let's look at the basics of how to create a variable.

Task 1
Write the following program in IDE, to create variables in C#.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            int INTEGER_VARIABLE = 100;
            Console.WriteLine(INTEGER_VARIABLE);

            string STRING_VARIABLE = "HELLO WORLD";
            Console.WriteLine(STRING_VARIABLE);
        }
    }
}
```

There are many others datatypes which we can use in our program as per need. Below is the detail of different datatypes;

Type Name	Description	Bytes	Data Range	Example
short	stores smaller integers	2	-32,768 to 32,767	short score = 495;
int	stores medium sized integers	4	-2,147,483,648 to 2,147,483,647	int score = 450000;
long	stores very large integers	8	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	long highScore = 4043333;
byte	stores small unsigned (no + or -) integers	1	0 to 255	byte color = 55;
ushort	stores small unsigned integers	2	0 to 65,535	ushort score = 495;
uint	stores medium unsigned integers	4	0 to 4,294,967,295	uint score = 4500000;
ulong	stores large unsigned integers	8	0 to 18,446,744,073,709,551,615	ulong highScore = 4043333;
float	stores smaller real (floating point) numbers	4	±1.5 × 10^−45 to ±3.4 × 10^38, 7 digits of accuracy	float xPosition = 3.2f;
double	stores larger real numbers	8	±5.0 × 10^−324 to ±1.7 × 10^308, 15 to 16 digits of accuracy	double yPosition = 3.3;
decimal	real numbers, smaller range, higher accuracy	16	±1.0 × 10^−28 to ±7.9 × 10^28, 28 to 29 digits of accuracy	decimal zPosition = 3.3;
char	stores a single character or letter	2	any single character	char myFavoriteLetter = 'c';
string	stores a sequence of numbers, letters, or symbols	any	a sequence of any character of any length	string message = "Hello World!"
bool	stores a *true* or *false* value	1	*true* or *false*	bool levelComplete = true;

Task 2
Write the following program in IDE, to take an input from the user.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            //For String
            string input1 = Console.ReadLine();
            Console.WriteLine(input1);

            //For Integer
            int input2 = Convert.ToInt16(Console.ReadLine());
            Console.WriteLine(input2);
        }
    }
}
```

4. Operators in C#

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. C# has rich set of built-in operators and provides the following type of operators:

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Misc Operators

Arithmetic Operators

Following table shows all the arithmetic operators supported by C#. Assume variable **A** holds 10 and variable **B** holds 20 then:

Operator	Description	Example
+	Adds two operands	A + B = 30
-	Subtracts second operand from the first	A - B = -10
*	Multiplies both operands	A * B = 200
/	Divides numerator by de-numerator	B / A = 2
%	Modulus Operator and remainder of after an integer division	B % A = 0
++	Increment operator increases integer value by one	A++ = 11
--	Decrement operator decreases integer value by one	A-- = 9

Relational Operators

Following table shows all the relational operators supported by C#. Assume variable **A** holds 10 and variable **B** holds 20, then:

Operator	Description	Example
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Logical Operators

Following table shows all the logical operators supported by C#. Assume variable **A** holds Boolean value true and variable **B** holds Boolean value false, then:

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero then condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non zero then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

Task 3

```
using System;
namespace AgeProgram
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Please Enter an Employee Age : ");
            int age = Convert.ToInt32(Console.ReadLine());

            if (age >= 18 && age < 58)
                Console.WriteLine("Eligible to work");
        }
    }
}
```

5. Conditional Statement

An **if** statement can be followed by an optional **else** statement, which executes when the Boolean expression is false.

Syntax

The syntax of an **if...else** statement in C# is:

```
if(boolean_expression)
{
  /* statement(s) will execute if the boolean expression is true */
}
else
{
  /* statement(s) will execute if the boolean expression is false */
}
```

If the Boolean expression evaluates to **true**, then the **if block** of code is executed, otherwise **else block** of code is executed.

Task 4

```
using System;

namespace ConditionalStatement
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter Any Number less than 6");
            int a = Convert.ToInt32(Console.ReadLine());
            if (a < 6)
            {
                Console.WriteLine(a);
            }
            else
            {
                Console.WriteLine("plz enter correct number");
            }

        }
    }
}
```

Task 5

```
using System;

namespace ConditionalStatement
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter Any Number between 1 to 5");
            int a = Convert.ToInt32(Console.ReadLine());
            if (a == 1)
            {
                Console.WriteLine("one");
            }
            else if (a == 2)
            {
                Console.WriteLine("two");
            }
            else if (a == 3)
            {
                Console.WriteLine("three");
            }
            else if (a == 4)
            {
                Console.WriteLine("four");
            }
            else if (a == 5)
            {
                Console.WriteLine("five");
            }
            else
            {
                Console.WriteLine("wrong number");
            }
        }
    }
}
```

6. Loops

There may be a situation, when you need to execute a block of code several number of times. In general, the statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or a group of statements multiple times and following is the general from of a loop statement in most of the programming languages:

C# provides following types of loop to handle looping requirements. Click the following links to check their detail.

Loop Type	Description
while loop	It repeats a statement or a group of statements while a given condition is true. It tests the condition before executing the loop body.
for loop	It executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
do...while loop	It is similar to a while statement, except that it tests the condition at the end of the loop body
nested loops	You can use one or more loop inside any another while, for or do..while loop.

FOR LOOP

A **for** loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax

The syntax of a **for** loop in C# is:

```
for (initialization of variable; condition; increment)
{
  statement(s);
}
```

Task 6

Write a Program which will take a number from User and Print its table?

```
using System;
namespace FoorLoop
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter a number to print table : ");
            int number = Convert.ToInt32(Console.ReadLine());

            for (int i = 1; i <= 10; i++)
            {
                Console.WriteLine("{0} x {1} = {2}", number, i, number *
i);
            }
        }
    }
}
```

NESTED FOR LOOP

C# allows to use one loop inside another loop. Following section shows few examples to illustrate the concept.

Syntax

The syntax for a **nested for loop** statement in C# is as follows:

```
for (initialization of variable; condition; increment)
{
  for (initialization of variable; condition; increment)
  {
    statement(s);
  }
  statement(s);
}
```

Task 7

Write a Program which print all the tables in the given range?

```
using System;

namespace TableRange
{
    class Program
    {
        static void Main(string[] args)
        {
            int start = Convert.ToInt32(Console.ReadLine());
            int end = Convert.ToInt32(Console.ReadLine());

            for (int i = start; i <= end; i++)
            {
                for (int j = 1; j <= 12; j++)
                {
                    Console.WriteLine("{0} x {1} = {2}", i, j, (i * j));
                }
                Console.WriteLine();
            }
        }
    }
}
```

While Loop

A **while** loop statement in C# repeatedly executes a target statement as long as a given condition is true.

Syntax

The syntax of a **while** loop in C# is:

```
while(condition)
{
  statement(s);
}
```

Task 8

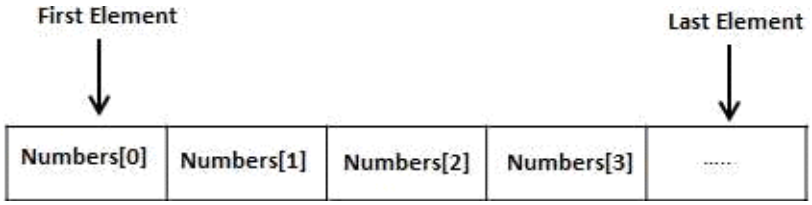
```
using System;
using System.Collections;

namespace TableRange
{
    class Program
    {
        static void Main(string[] args)
        {
            int num = 1;
            while (num < 11)
            {
                Console.WriteLine(num);
                num++;
            }
        }
    }
}
```

7. Working with Array

n 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.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

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



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.

For example,
```
double[] balance;
```

Initializing an Array
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.

For example,
```
double[] balance = new double[10];
```

Assigning Values to an Array
You can assign values to individual array elements, by using the index number, like:

```
double[] balance = new double[10];
balance[0] = 4500.0;
```

You can assign values to the array at the time of declaration, as shown:
```
double[] balance = { 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, as shown:
```
int [] marks = new int[]  { 99,  98, 92, 97, 95};
```

Task 9
Write a Program Which Will take input from user and store it in an array using Loop, also print the sum of the values stored in array.

```
using System;
using System.Collections;

namespace ArrayWorking
{
    class Program
    {
        static void Main(string[] args)
        {
            int size = 5;
            int[] intarray = new int[size];
            for (int i = 0; i < size; i++)
            {
                Console.WriteLine("Enter Value at {0}", i);
                int input = Convert.ToInt32(Console.ReadLine());
```

```
                intarray[i] = input;
            }

            int sum = 0;
            for (int j = 0; j < intarray.Length; j++)
            {
                sum = sum + intarray[j];
            }

            Console.WriteLine("The sum of the values in array is : " +
sum);

        }
    }
}
```

8. Lab Task

1. Write a C# console program that reads input & calculates the area of circle & Triangle?
2. Create a calculator which will ask user to input 2 number and an operation (Add, Subtract, Divide and multiply) that user want to perform.
3. Take two numbers from user and find out all even and odd numbers between then using while loop?
4. Write a program to print a string in reverse order. For your convenience the output is given below (Hint: Use string length property)
   Output:
   Input A string: Welcome
   The string in the reversed string is: emocleW
5. Write a program which will take input from user and search it in an array if input number is present in an array then prints “Found” otherwise “Not Found”?
6. Write a Program to make the pattern using C# on output screen

1 2 3 4 5
2 1 2 3 4
3 2 1 2 3
4 3 2 1 2
5 4 3 2 1