Information Processing Techniques

Lab 1

Introduction to C#

February 24, 2021

Programming in C#

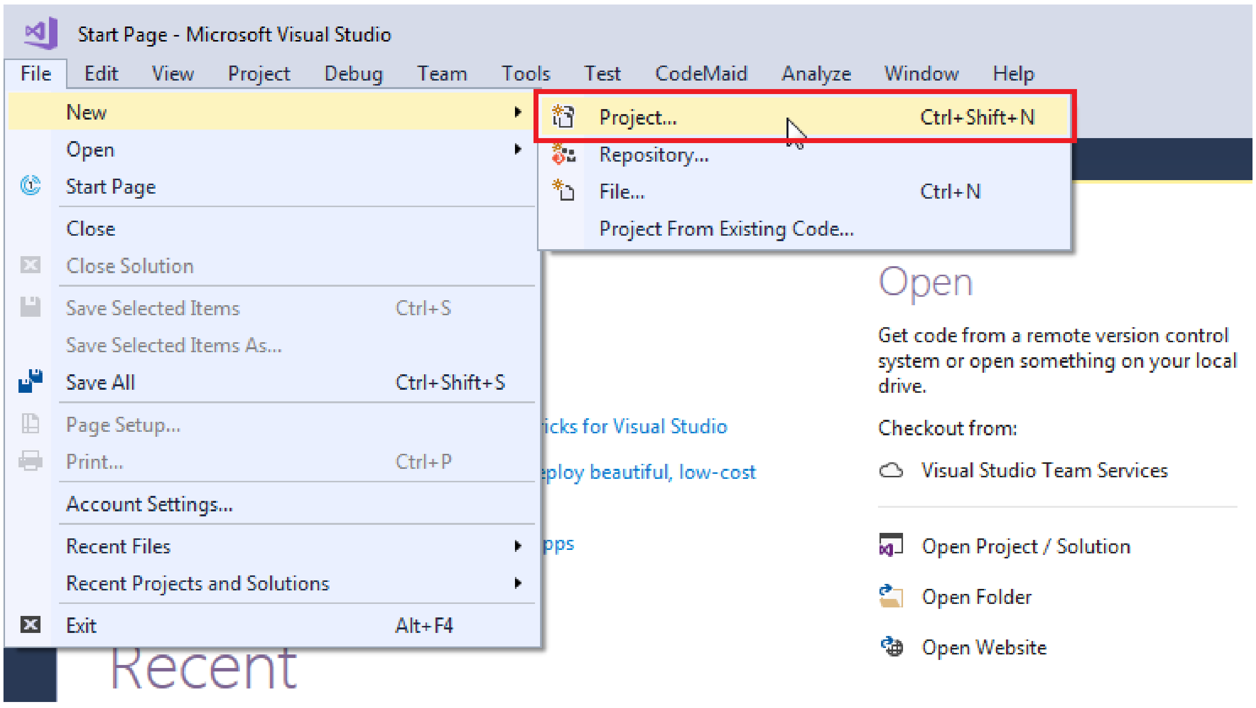
Introduction

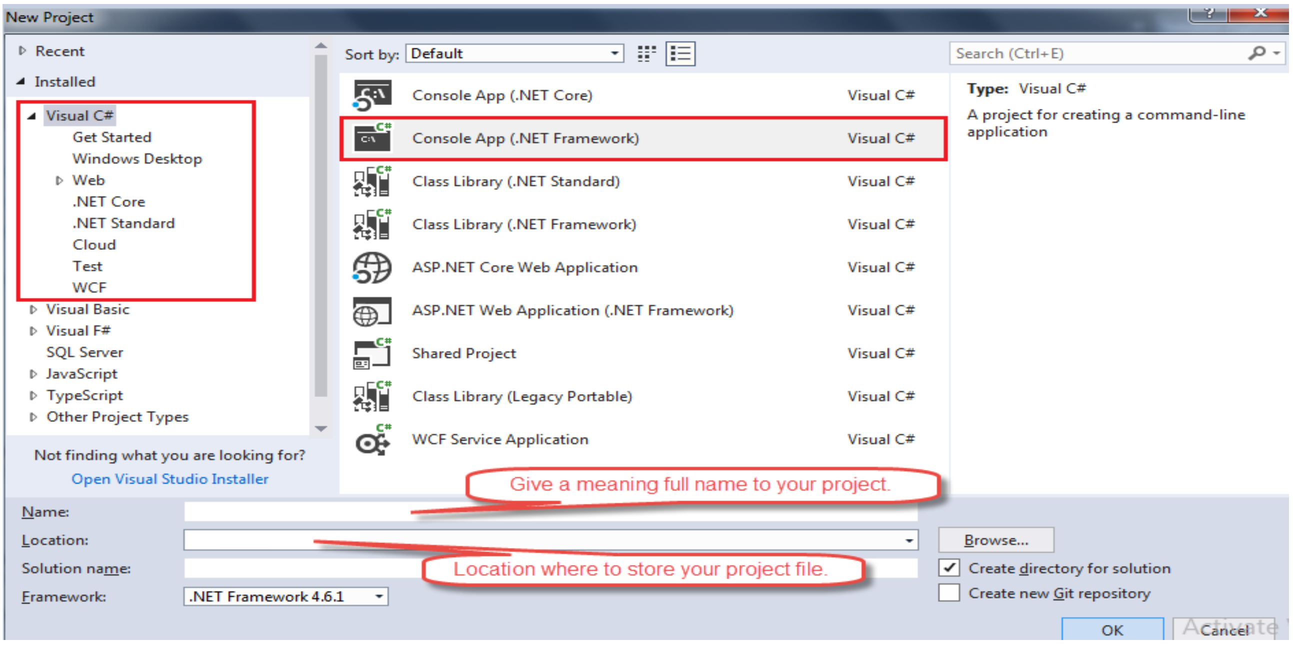
The C# programming language (pronounced "C sharp") was introduced with the .NET Framework. It represents a milestone in the area of programming language development. Even though the genesis of the new C# language is in existing programming languages such as Java and C++, it represents the most innovative, modern, and even—in most areas—the preferred .NET programming language. C# was introduced to combine the key values of Visual Basic and C++. Whereas Visual Basic had ease of use and high developer productivity, C++ represented the ultimate flexibility and control.

Integrated Development Environment (IDE)

An integrated development environment IDE is a tool that helps you write your programs. Microsoft provide visual studio to write C# code. Now let's set up development environment.

Open visual studio and follow the steps to create new project.





Console App (.NET Core)	Visual C#
Console App (.NET Framework)	Visual C#
Class Library (.NET Standard)	Visual C#
Class Library (.NET Framework)	Visual C#
ASP.NET Core Web Application	Visual C#
ASP.NET Web Application (.NET Framework)	Visual C#
Shared Project	Visual C#
Class Library (Legacy Portable)	Visual C#
WCF Service Application	Visual C#

Type: Visual C#

A project for creating a command-line application

Name:

Location:

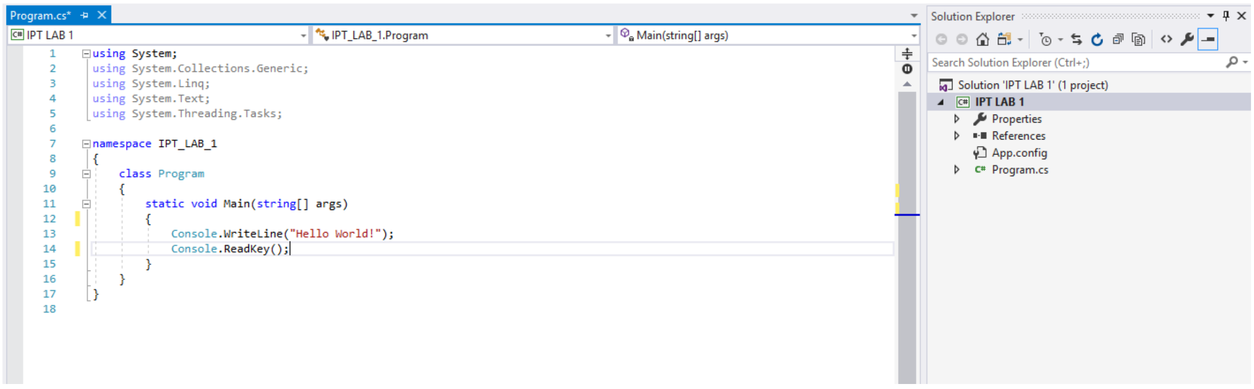
Solution name:

Framework: .NET Framework 4.6.1

[✓] Create directory for solution

[ ] Create new Git repository

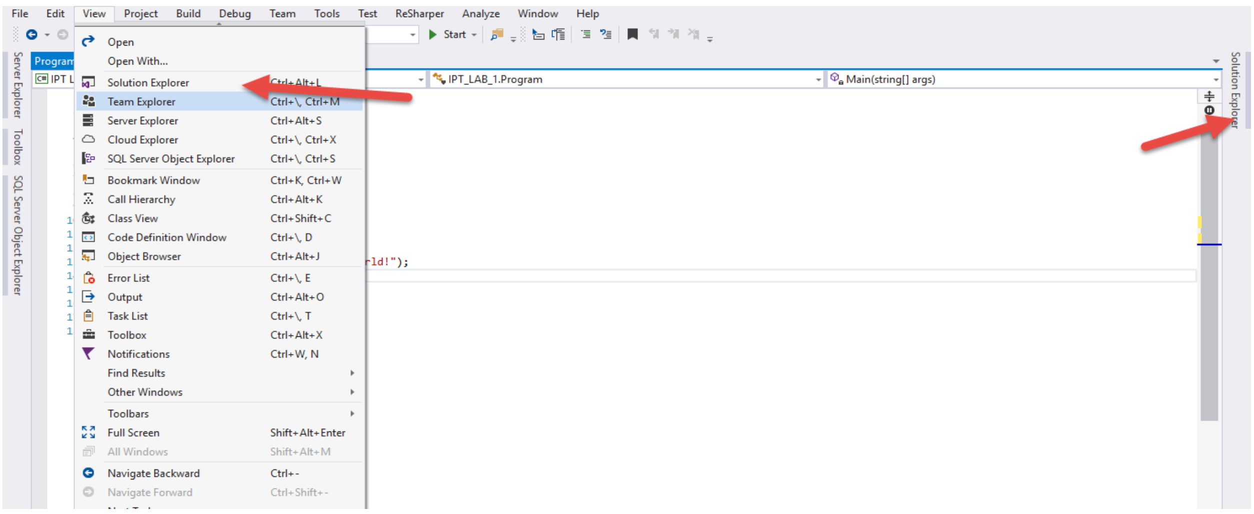
Once you create new project visual studio automatically created **program.cs** file which will be the main for the project.



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

namespace IPT_LAB_1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
            Console.ReadKey();
        }
    }
}
```

You can navigate to project artifacts in Solution explorer, it is usually on the right side but in case you are unable to locate it you can also access it from menu.

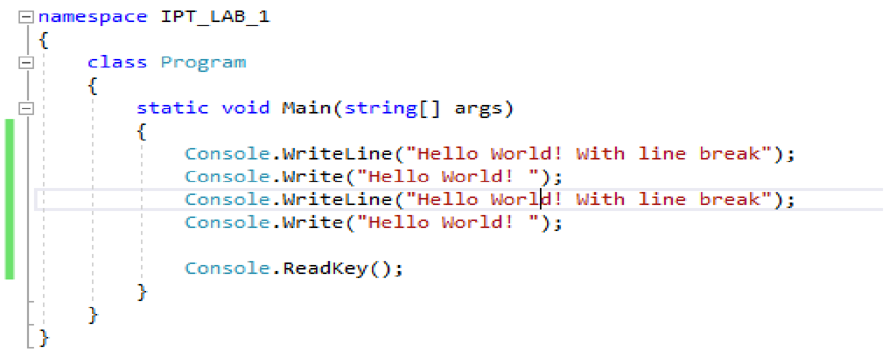


Standard input/output

Output

For standard output we have Console.WriteLine() and Console.Write() the difference between two is Console.WriteLine() will automatically move to next line after printing whereas Console.Write() will stay in the same line.

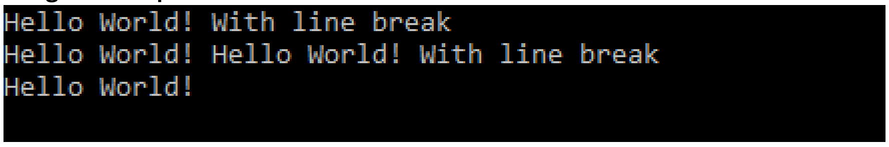
```
Console.WriteLine("Hello World!");
Console.Write("Hello World!");
```



```
namespace IPT_LAB_1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World! With line break");
            Console.Write("Hello World! ");
            Console.WriteLine("Hello World! With line break");
            Console.Write("Hello World! ");

            Console.ReadKey();
        }
    }
}
```

Program Output



```
Hello World! With line break
Hello World! Hello World! With line break
Hello World!
```

Input

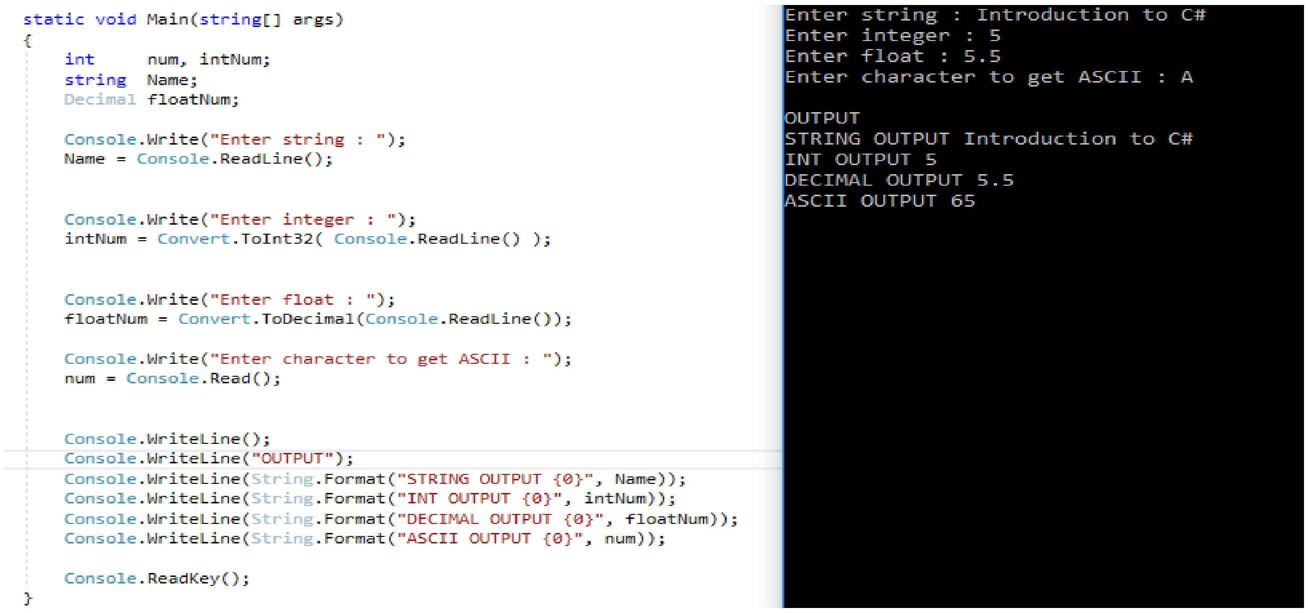
For standard output we have the following three functions.

1. **Console.Read();** Reads the next character from the standard input.
2. **Console.ReadLine();** Reads the next lines of character from the standard input.
3. **Console.ReadKey();** Read only one character usually used then you give used an option like press Y Or N to continue etc.

The basic difference is:

Console.Read give int value but that value will be the ASCII value of that. On the other side, Console.ReadLine gives the string as it is given in the input stream.

NOTE: both Console.Read and Console.ReadLine is used to get character input to take input in other data types we first need to convert using CONVERT class.



```
static void Main(string[] args)
{
    int     num, intNum;
    string  Name;
    Decimal floatNum;

    Console.Write("Enter string : ");
    Name = Console.ReadLine();


    Console.Write("Enter integer : ");
    intNum = Convert.ToInt32( Console.ReadLine() );


    Console.Write("Enter float : ");
    floatNum = Convert.ToDecimal(Console.ReadLine());

    Console.Write("Enter character to get ASCII : ");
    num = Console.Read();


    Console.WriteLine();
    Console.WriteLine("OUTPUT");
    Console.WriteLine(String.Format("STRING OUTPUT {0}", Name));
    Console.WriteLine(String.Format("INT OUTPUT {0}", intNum));
    Console.WriteLine(String.Format("DECIMAL OUTPUT {0}", floatNum));
    Console.WriteLine(String.Format("ASCII OUTPUT {0}", num));

    Console.ReadKey();
}
```

```
Enter string : Introduction to C#
Enter integer : 5
Enter float : 5.5
Enter character to get ASCII : A

OUTPUT
STRING OUTPUT Introduction to C#
INT OUTPUT 5
DECIMAL OUTPUT 5.5
ASCII OUTPUT 65
```

Code

```
int     num, intNum;
string  Name;
Decimal floatNum;

Console.Write("Enter string : ");
Name = Console.ReadLine();


Console.Write("Enter integer : ");
intNum = Convert.ToInt32( Console.ReadLine() );


Console.Write("Enter float : ");
floatNum = Convert.ToDecimal(Console.ReadLine());

Console.Write("Enter character to get ASCII : ");
num = Console.Read();


Console.WriteLine();
Console.WriteLine("OUTPUT");
Console.WriteLine(String.Format("STRING OUTPUT {0}", Name));
Console.WriteLine(String.Format("INT OUTPUT {0}", intNum));
Console.WriteLine(String.Format("DECIMAL OUTPUT {0}", floatNum));
Console.WriteLine(String.Format("ASCII OUTPUT {0}", num));

Console.ReadKey();
```

Data Types

The data type tells the C# compiler what kind of value a variable can hold. C# includes many in-built data types for different kinds of data, e.g. String, number, float, decimal, etc.

Reserved Word	.NET Type	Type	Size (bits)	Range (values)
byte	Byte	Unsigned integer	8	0 to 255
sbyte	SByte	Signed integer	8	-128 to 127
short	Int16	Signed integer	16	-32,768 to 32,767
ushort	UInt16	Unsigned integer	16	0 to 65,535
int	Int32	Signed integer	32	-2,147,483,648 to 2,147,483,647
uint	UInt32	Unsigned integer	32	0 to 4294967295
long	Int64	Signed integer	64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
ulong	UInt64	Unsigned integer	64	0 to 18,446,744,073,709,551,615
float	Single	Single-precision floating point type	32	-3.402823e38 to 3.402823e38
double	Double	Double-precision floating point type	64	-1.79769313486232e308 to 1.79769313486232e308
decimal	Decimal	Precise fractional or integral type that can represent decimal numbers with 29 significant digits	128	(+ or -)1.0 x 10e-28 to 7.9 x 10e28
char	Char	A single Unicode character	16	Unicode symbols used in text
bool	Boolean	Logical Boolean type	8	True or False
object	Object	Base type of all other types		
string	String	A sequence of characters		
DateTime	DateTime	Represents date and time		0:00:00am 1/1/01 to 11:59:59pm 12/31/9999

Enumerations

C# provides the enumerations programming construct, which provides a human-readable form of a series of related constant values.

```
enum Days{
            Monday   = 0,Tuesday  = 1,Wednesday= 2,Thursday = 3,
            Friday   = 4, Saturday = 5,Sunday   = 7
         };
static void Main(string[] args)
{
Days day= Days.Monday;
Console.WriteLine(day);
Console.WriteLine((int)day);
Console.ReadKey();
}
```

Conditional Statement

C# provide following conditional statements

- If statements
- If….else statements
- Switch statements

```
int num;
Console.Write("Enter Number : ");
num = Convert.ToInt32(Console.ReadLine());
if(num%2 == 0)
{
Console.WriteLine("Number is Even");
}
else
{
Console.WriteLine("Number is Odd");
}
```

```
int num;
Console.Write("Enter Number : ");
num = Convert.ToInt32(Console.ReadLine());
switch(num % 2)
{
case 0:
{
Console.WriteLine("Number is Even");
break;
}
case 1:
{
Console.WriteLine("Number is Odd");
break;
}

}
```

Loops

C# provides following types of loop to handle looping requirements.

- For loop
- While loop
- Do while loop
- For each loop

For each Loop is used to iterate elements in a collections.

```
Char[] myArray = { 'H', 'e', 'l', 'l', 'o', 'w', 'o', 'r', 'l', 'd' };
for (int i = 0; i < myArray.Length; i++)
{
  Console.WriteLine(myArray[i]);
}
Console.ReadKey();
```

Methods

A method is a group of statements that together perform a task, The syntax for defining a method in C# is as follows.

```
<Access Specifier> <Return Type> <Method Name>(Parameter List) {
  Method Body
}
```

Class

A class definition starts with the access specifier, followed by keyword class then the class name; and the class body enclosed by a pair of curly braces. Following is the code example of the class.

```
class Rectangle
        {
            public double length;   // Length of a Rectangle
            public double breadth;  // Breadth of a Rectangle

            public Rectangle(double l,double b)
            {
                length  = l;
                breadth = b;
            }

            public double area()
            {
                return length * breadth ;
            }
        }
static void Main(string[] args)
        {
            Rectangle rec = new Rectangle(5, 8);
            Console.WriteLine(String.Format("Area Of Rectangle {0}",rec.area()));
            Console.ReadKey();
        }
```

Access Modifiers

Access modifiers (access specifiers) describes the scope of an Object and its members. C# provides following access modifiers.

- Public
- Private
- Protected
- Internal
- Protected Internal

Public

It is the most common access specifier in C# which allow access from anywhere.

Private

The scope of the accessibility is limited only inside the classes or struct in which they are declared.

Protected

Its scope is limited within the class or struct and the class derived (Inherited) from this class.

Internal

This can access within the program that contain its declarations and also access within the same assembly level but not from another assembly.

Protected Internal

It is same as both protected and internal. It can access anywhere in the same assembly and in the same class also the classes inherited from the same class.

Practice Questions

Question 1: Write a C# program which will take a number as input from user and print all the prime number up to that number.

Question 2: Write a C# program which will accept string input from user and create a new string with first 4 characters will be in lower case and rest of the letter will be upper case.

Question 3: Write a C# program which will take matrix size as input from user then generate a random matrix and find sum of left diagonals of a matrix.

Question 4: Create a class 'Calculator' which would have the following methods:
• Add
• Subtract
• Multiple
• Divide
Now overload these methods to accommodate multiple parameters (i.e. 2 parameters, 3 parameters, and n number of parameters).

Question 5: Create a class 'Amount' which is derived from the base class 'Account'. The base class has a method int balance(), you have to override it in the 'Amount' class.

Question 6: Create a class 'Student' with the following properties:
• Name
• Year of Birth
• Semester
• GPA
Implement the IComparable<> interface so that you can use the Sort method. You should be able to sort the List<Student> in descending order of GPA.

Question 7: You have two character arrays (strings). You have to compare if both the values are equivalent or not using the below method?

```
string string1 = new string(new char[] { 'h', 'e', 'l', 'l', 'o' });
string string2 = new string(new char[] { 'h', 'e', 'l', 'l', 'o' });
```

```
public static bool CompareObjects (object name, object name2) {
// write your comparison code here
}
```