Introduction

This course is mainly designed to solve practical problem using the suitable data structure and algorithm to give you hands-on experience on topics covered in the lectures. The programming language that has been chosen for this module is C#. The reason for this is that C# is the most widely used general purpose programming language for Windows and GUI programming. It is also relatively learner-friendly for an introductory programming language. Students who learn C# will have little difficulty picking up Java, C or C++.

Development Environment

In this module, we shall be using Visual Studio for developing programs for solving the practical problems. For developing and practicing at home, you can download the express edition of Visual Studio Community (free) from the Visual Studio web site: https://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx. This is a free, fully-featured, and extensible Integrated Development Environment (IDE) for creating applications for Windows, Android, and iOS, as well as web applications and cloud services. This could be useful for other modules in which you are required to develop codes in other languages and/or different platforms. If you do not want all other features then install Microsoft Visual Studio Express (free) https://www.microsoft.com/en-gb/download/details.aspx?id=34673 for developing solution using C#.

Creating Your First C# Program

First, you have to create a new project by selecting an in-built template from the Visual Studio (VS). The VS will create an empty project for you afterwards you can add C# files to the project. In order to do this, start VS and from the Start Page window pick “File->New->Project (see Figure 1).

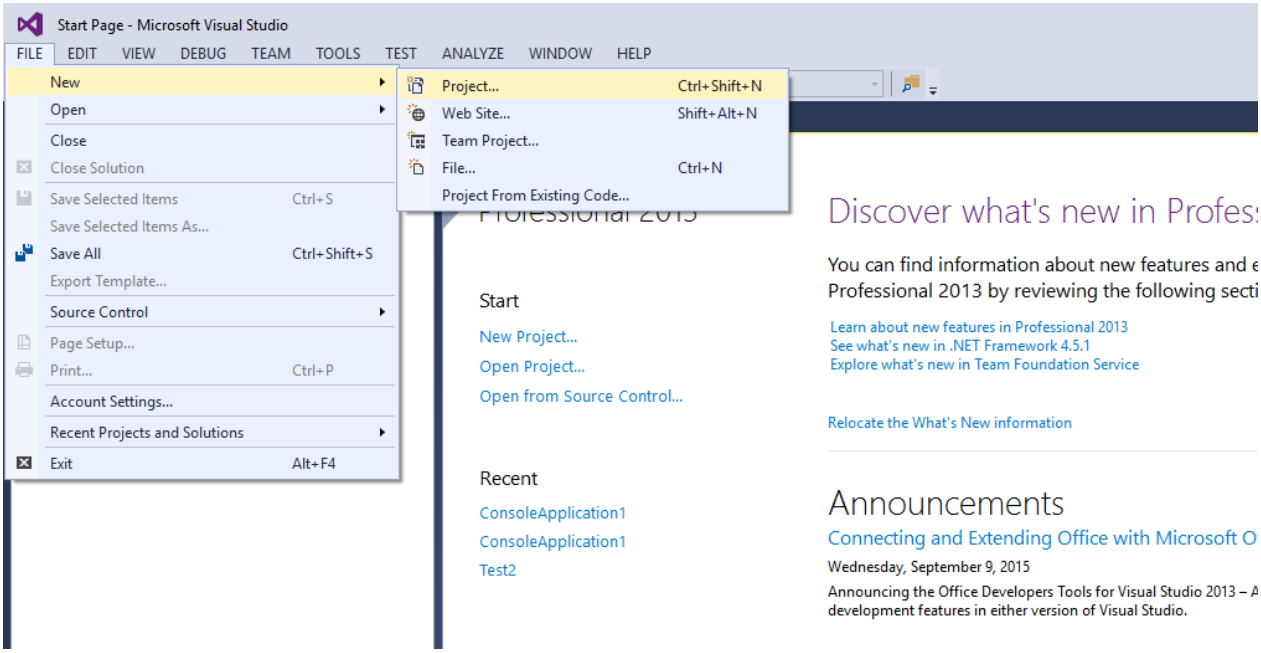

Figure 1 The Visual Studio Start-up Screen

In the New Project, you should select “Console Application” as shown in Figure 2. If you wish you could change the name and location of the project ab the bottom of the template. In this one, we name it as a “ConsoleApplication2”. Then press the OK button.

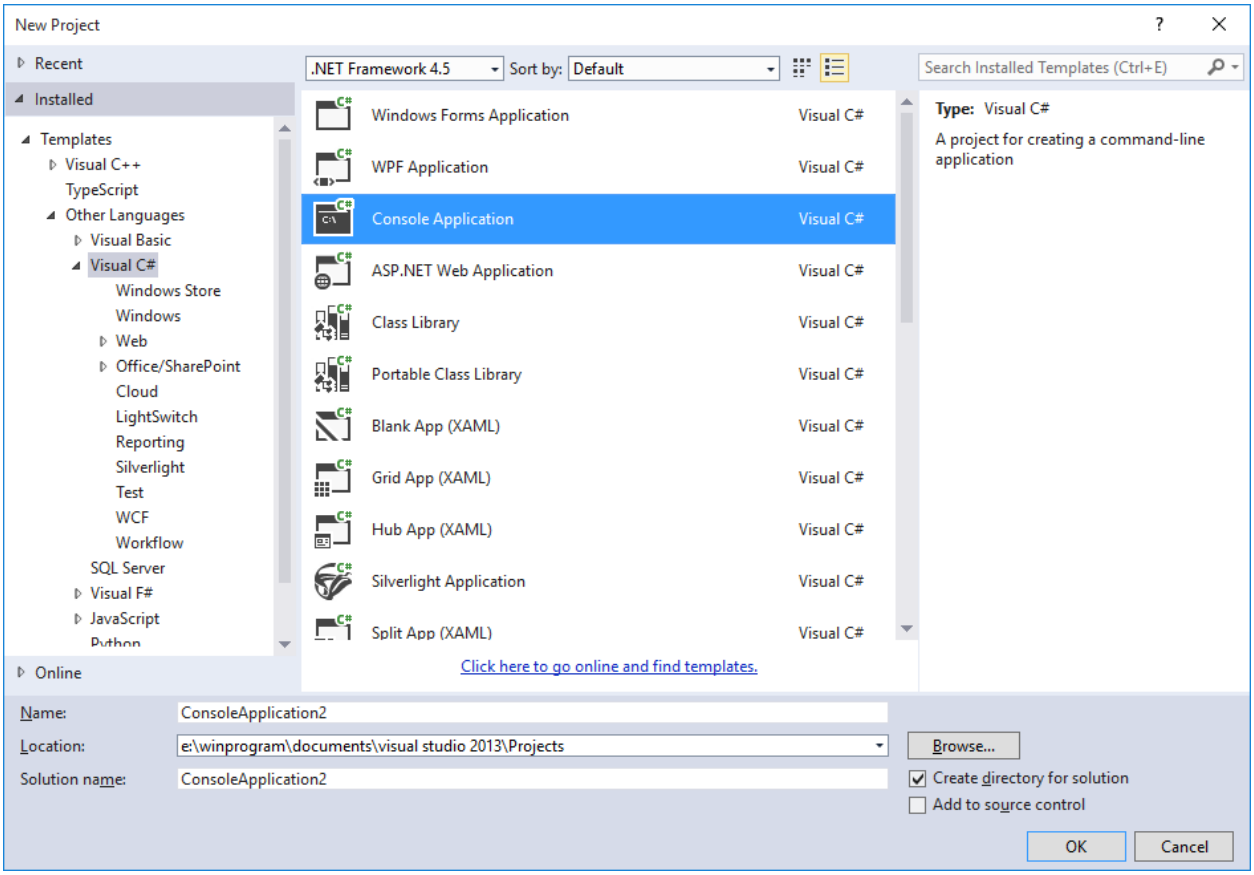

Figure 2 The Visual Studio New Project Screen

After this, you can will see the Visual Studio main window, with your “ConsoleApplication2” solution left/right pane. In the figure below (Figure 3), it is in the left pane.

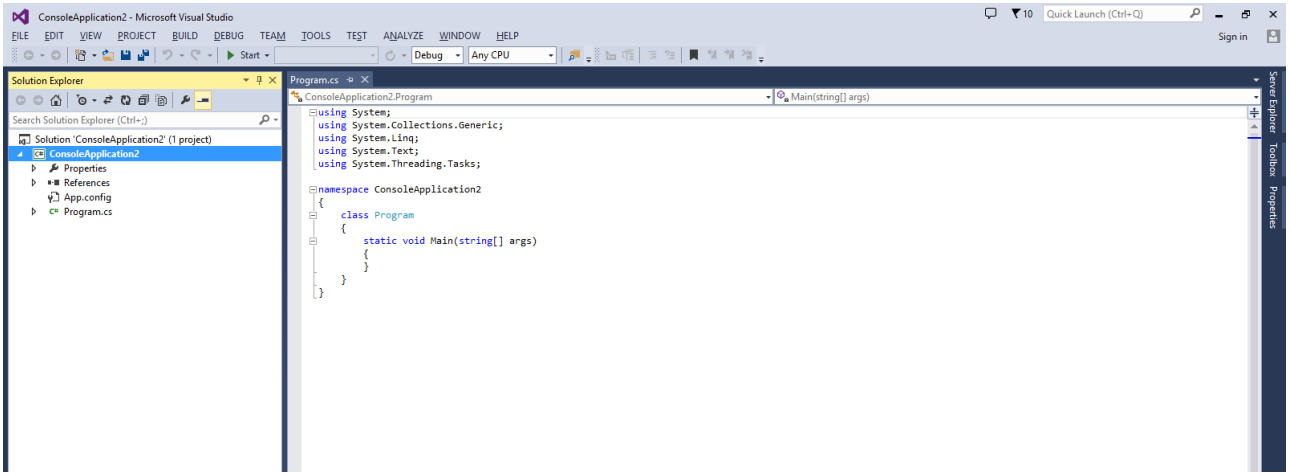

Figure 3 The default empty project of the Visual Studio.

Adding Code to a Project

You can now add source code to your project. This is done by adding files with ".cs" extension to the project. The source files will contain the class definitions that make up your application. Your application could have many files within it. To begin with we will just create a single source file. This has been done for you, automatically (usually called Program.cs). This is the file we will be working on for now.

Later on, you can easily add a new file by right-clicking on the Project folder in the Solution Explorer and selecting Add->New Item. See below Figure 4.

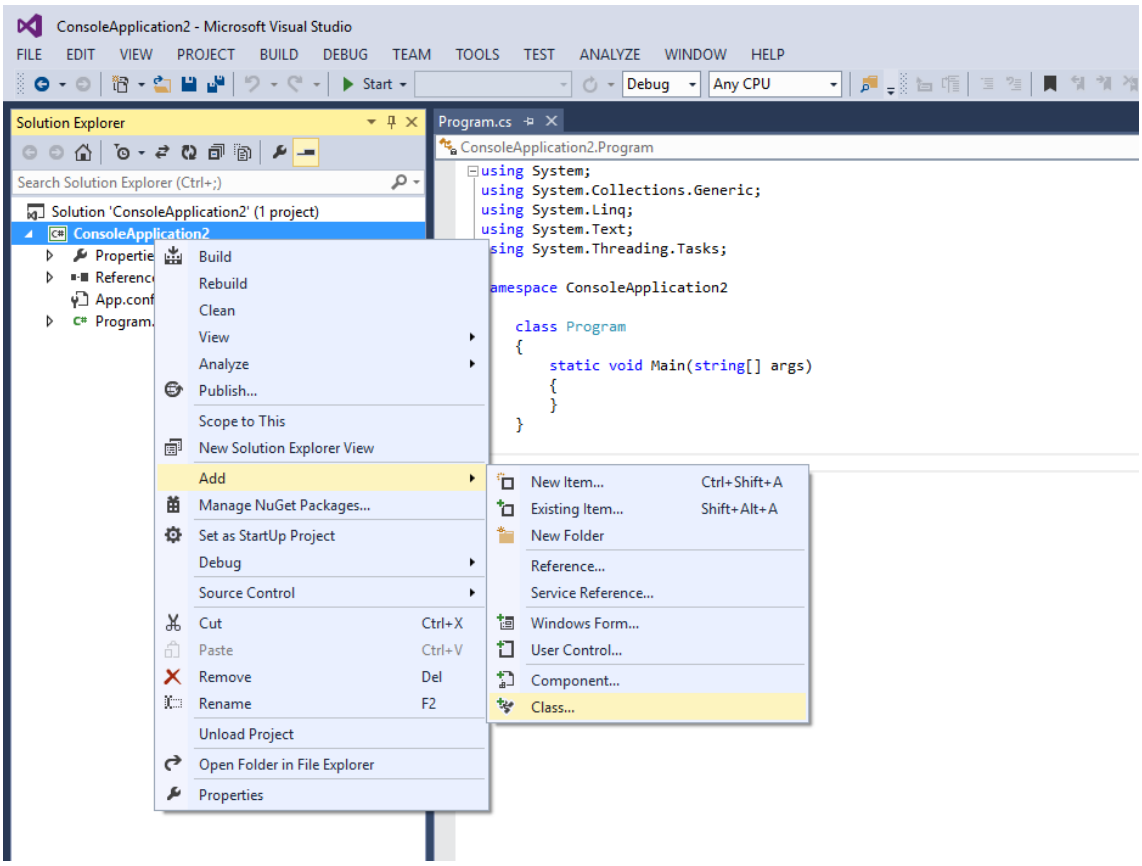

Figure 4 Adding new source file.

In the Add New Item window, you are required to select the Class template in most cases. Later in the course we might use some of the other templates. Once you have selected a template, you are required to change its name (using the text box at the end of the dialog window). Note the ".cs" file extension. See below Figure 5. Then click Add button to create this new "MyClass.cs" source file.

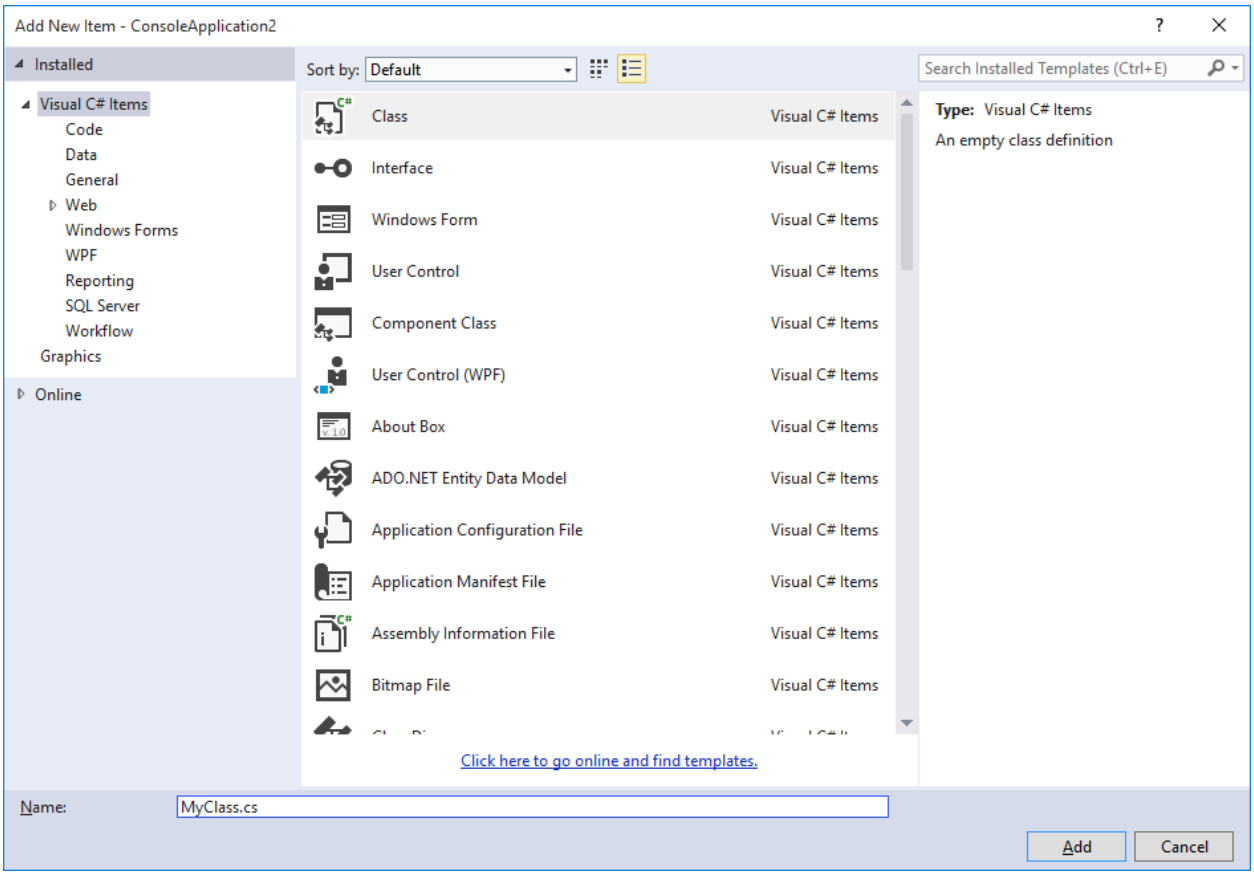

Figure 5 Adding a new .cs file

Now, you can see the new file named “MyClass.cs” is added and that an editor window has been opened in the main pane for you to edit the code (see Figure 6). You can always select the file you wish to edit by double-clicking on its name in the solution explorer.

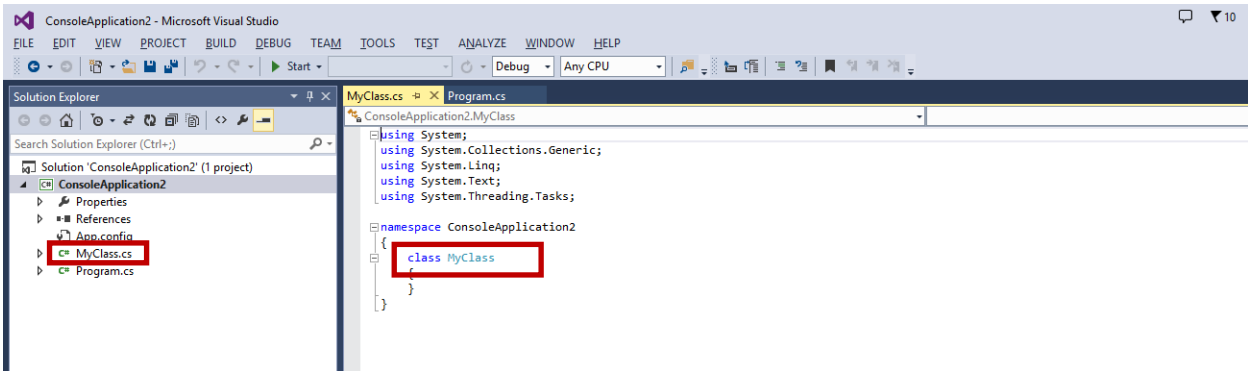

Figure 6 Added “MyClass.cs” file to the project “ConsoleApplication2”.

You can now enter the source code for your first C# program. Now open the file “Program.cs” which is already created for you and add the line `Console.WriteLine("This is my first C# Program");` to the “Main” method as shown below (Figure 7).

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

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("This is my first C# Program");
        }
    }
}
```

Figure 7 Adding a line of code to the Program.cs file

Once the code is added, save the file (from the file menu or using shortcut key CTRL+S) and then build your code by selecting “Build Solution” from the Build menu or pressing button F7.

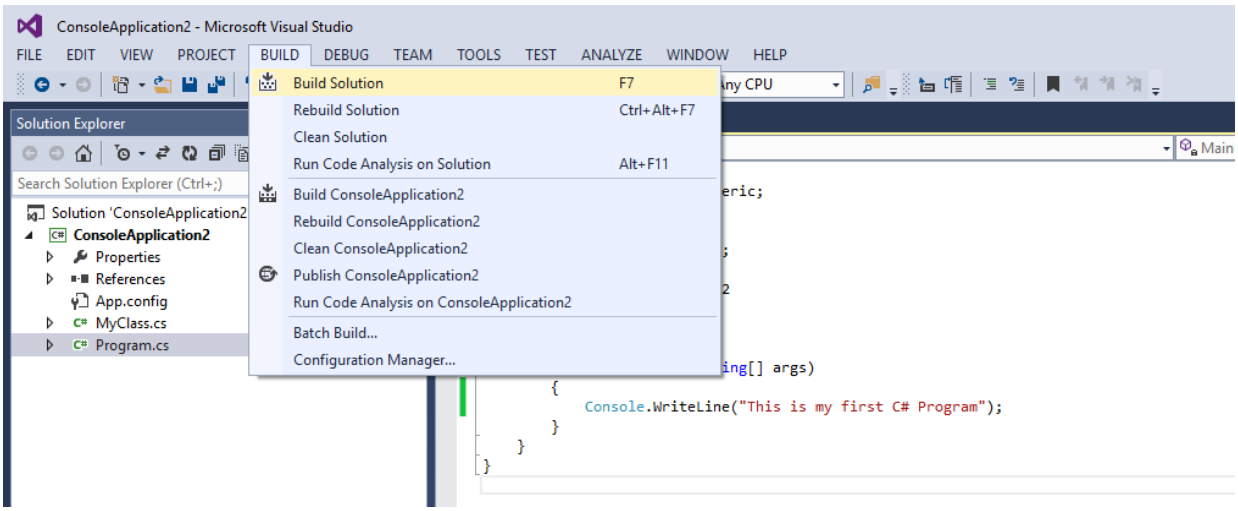

Figure 8 Building the solution.

Then look for the message in the output window of the Visual Studio whether the build is successful or not. If it is successful, it will show “succeeded” message as shown in the Figure 9.

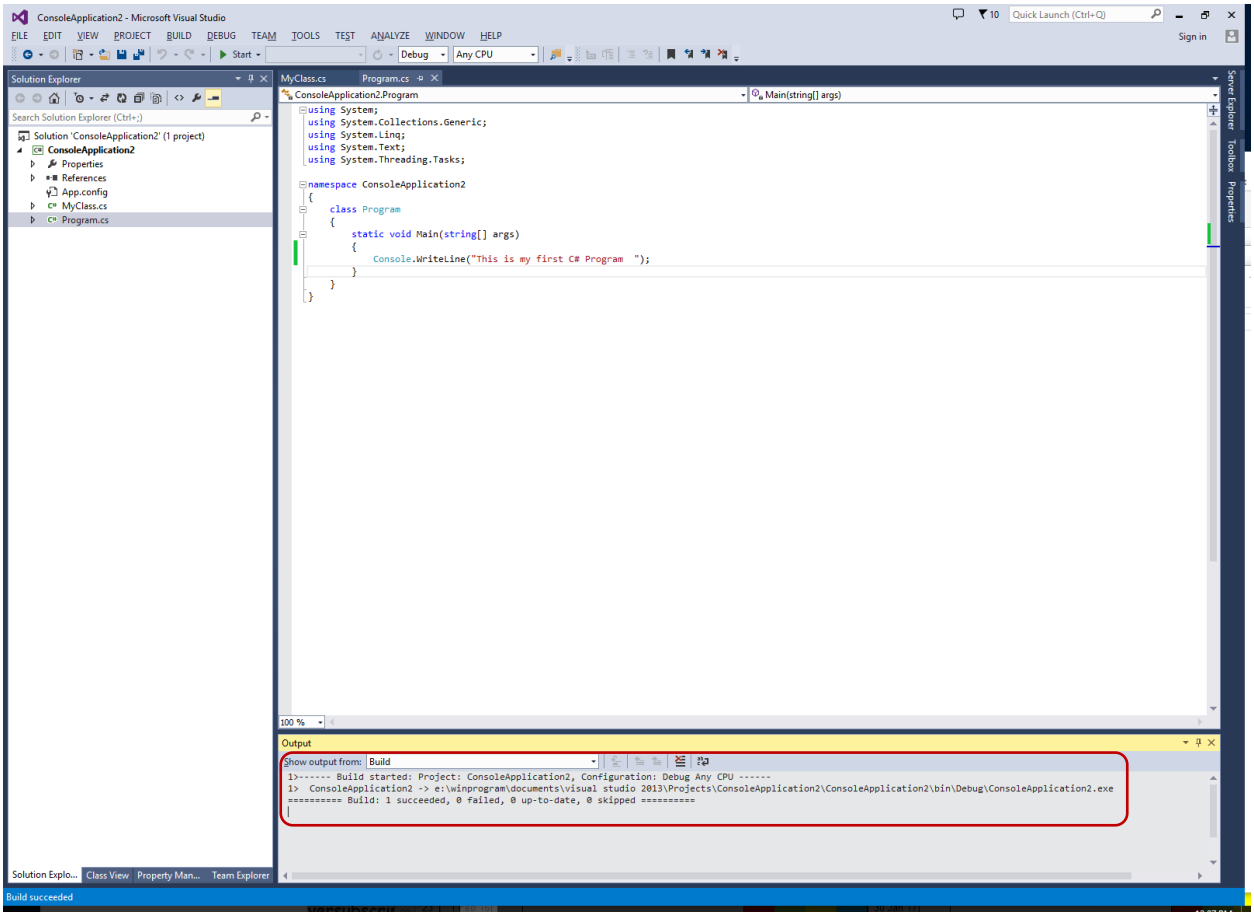

Figure 9 Build output.

Now, you can execute the program by going to the Debug menu and selecting “Start Without Debugging” or using shortcut key CTRL+F5 as shown below.

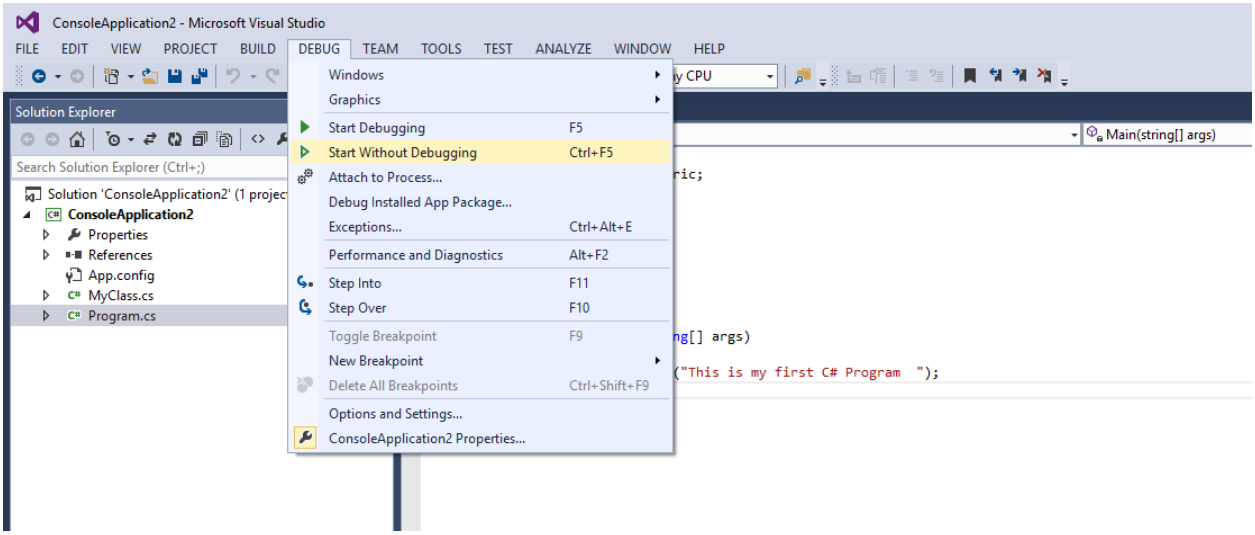

Figure 10 Executing the program.

You should get a terminal window appearing with message “This is my first C# Program” as shown below.

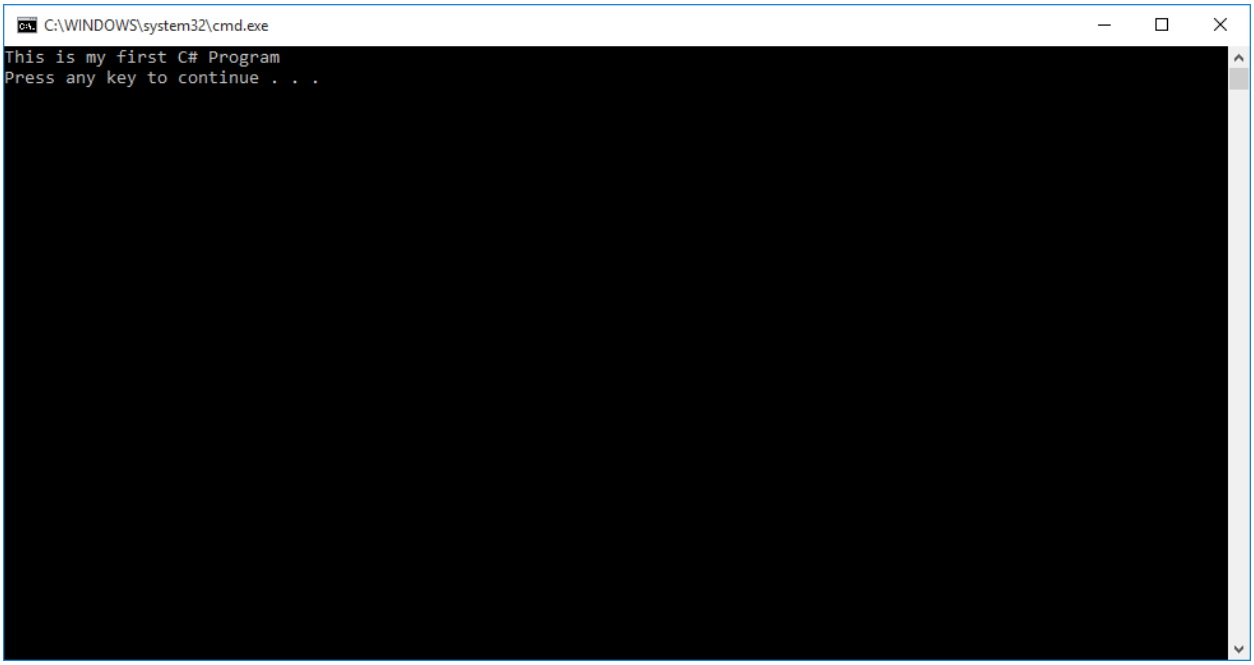

Figure 11 Console output displaying “This is my first C# Program” message.

How the program works?

The first 5 lines of the program are used to access the C# library codes. In this case: `System`, `System.Collections.Generic`, `System.Linq`, `System.Text` and `System.Threading.Tasks`. These libraries are the core functionalities of the C# and are added to your code via keyword `using`. Your own C# code requires a these in-built libraries.

Your class/methods/codes will be placed in the next block starting with the keyword `namespace`. This is automatically created for you while generating an empty new project. In this case the name is given as “ConsoleApplication2”. If you wish you can change this name as well. A namespace is a mechanism for creating scope within an application and can be used to hierarchically organise classes and is similar to the use of packages in Java. Similar to Java, your program consists of class definitions and at least one class must have a Main method in order to run. The Main method is the starting point for the code execution. Finally, the line `Console.WriteLine("This is my first C# Program");` used to print the string (within the double quotes) out to the standard out stream. The method “WriteLine” to display string on the console. Similar to Java, you could append strings using the ‘+’ symbol. For example

```
Console.WriteLine("My name is " + "Ardhendu" + " " + "Behera");
```

Would write the following message on the terminal output

```
My name is Ardhendu Behera
```

Your program can read from the command line using `Console.ReadLine()`. In order to use it you also need lines of code for reading values from the command line. The entered value will be read after each time the "Enter" (return) key is pressed. The value read will be in string format. You need to convert it to the respective format before manipulating it. Try to build and run the following code.

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

namespace InputOutput
{
    class Program
    {
        static void Main(string[] args)
        {

            string s1, s2;
            Console.WriteLine("Enter value 1:");
            s1 = Console.ReadLine();
            Console.WriteLine("Enter value 2:");
            s2 = Console.ReadLine();
            //Enter values are stored as string NOT integer
            Console.WriteLine("The entered values are: " + s1 + " and " + s2);
            //Convert them to integer and print the sum
            int i1, i2;
            i1 = Convert.ToInt32(s1);
            i2 = Convert.ToInt32(s2);
            Console.WriteLine("The sum is: " + (i1+i2));
        }
    }
}
```

Exercises

EX1:
Modify the last line of the above code by removing the parentheses around "i1+i2". Compile and run the code. What output do you see? Explain the reason.

EX2:
Create a program in a new project that allows a user to enter their forename, then their surname and then print out the result on a single line. The user should be prompted to enter each piece of data separately. For example, one would see the bellow message in the terminal when you run your code.

```
Please enter your name:
Ardhendu
Please enter your surname:
Behera
Your name is Ardhendu Behera
```

EX3:
Extend the above program in EX2 for printing the full name of more than one person. The program should be in a new project and able to read and write the name of multiple person. For example your code should display the following output.

```
Please enter the number of people:
5
Person 1, please enter your name:
Ardhendu
Person 1, please enter your surname:
Behera
The name of person 1 is Ardhendu Behera
```

Your code should display the name of person 1 to 5 and then exit.

EX4:
Create a program in a new project that calculates the area and perimeter of a circle. Your program should prompt the user for the radius of the circle, and should use a constant for π assigned the value of 3.142. Your program should able to display the perimeter and area of the circle.

EX5:
Extend the above program (EX4) by defining π in a separate class file (".cs") rather than the program itself. In order to do this, you need to add a new class file (call it "Pi.cs") to your project and move the definition from the "Program.cs" to the new class. Access it from your "Program.cs".