

MICROSOFT C#

CONSOLE APPLICATION

AN INTRODUCTION TO PROGRAMMING

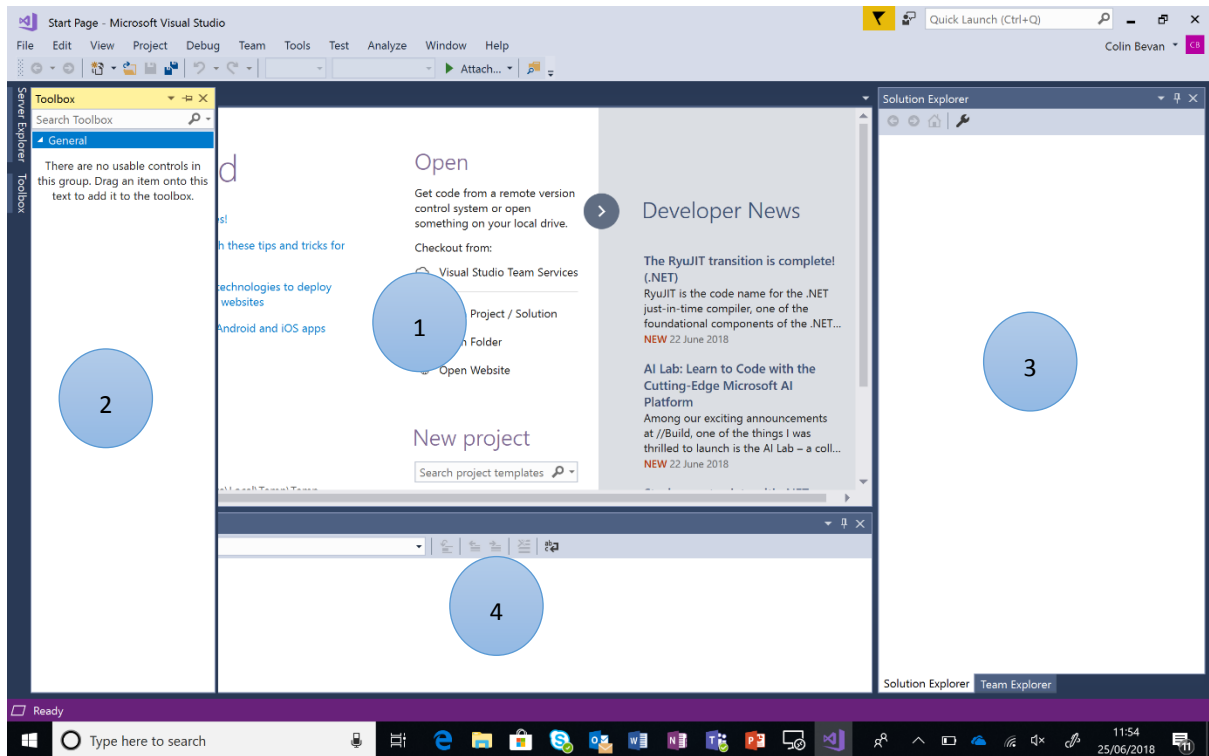
Computer Programming is defined as the process of creating computer software using programming languages. Just like we speak and understand English or Spanish or French, computers can understand programs written in certain languages. These are called programming languages. In the beginning there were just a few programming languages and they were really easy to learn and comprehend. But as computers and software became more and more sophisticated, programming languages evolved fast, gathering more complex concepts along the way. As a result most modern programming languages and their concepts are pretty challenging to grasp by a beginner. This fact has started discouraging people from learning or attempting computer programming.

Visual Studio and C# is a professional development tool but before we progress onto using some of the professional development tools we need to grasp the basic of computer programming. These notes will focus on building console applications within the C# development environment.

We are using Visual Studio Community Edition 2017 in college. The Microsoft Community product range is free which means you can download this at home to continue with your work. C# is one of the development tools within the Visual Studio package.

THE C# DEVELOPMENT ENVIRONMENT

Let us start with a quick introduction to the Visual Studio Environment. Find Visual Studio Community 2017 and launch the Visual Studio 2017 application. When you first launch Visual Studio, you will see a window that looks like the following figure.



Area 1 – In this area you manage your different projects. On the left hand side of this area is the New and Open Project links plus Recent which lists your most recent projects.

Area 2 – The toolbox is not used in console applications so you can un-pin this window.

Area 3 – The solution explorer window show all the different components used in the project.

Area 4 – The output area is used when running the application.

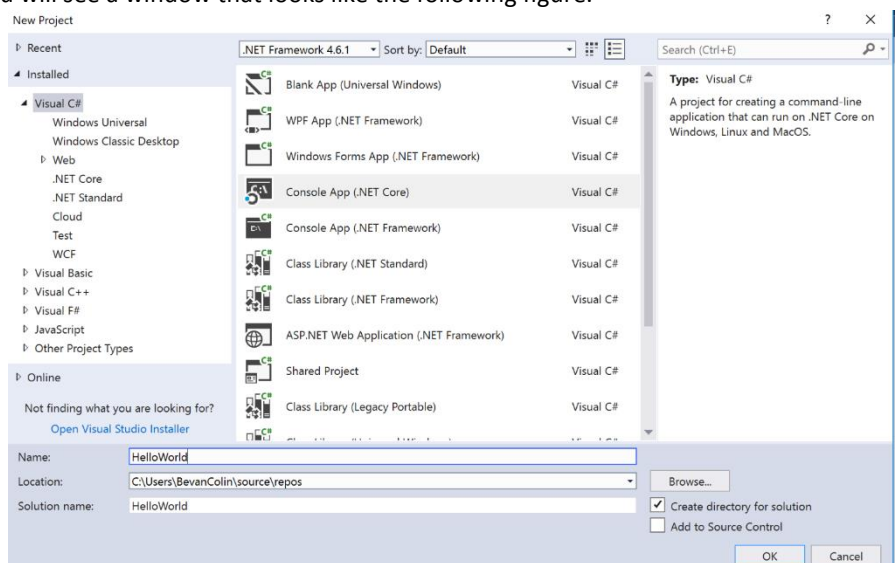
Click on File, New and Project. You will see a window that looks like the following figure.

Select Console Application.

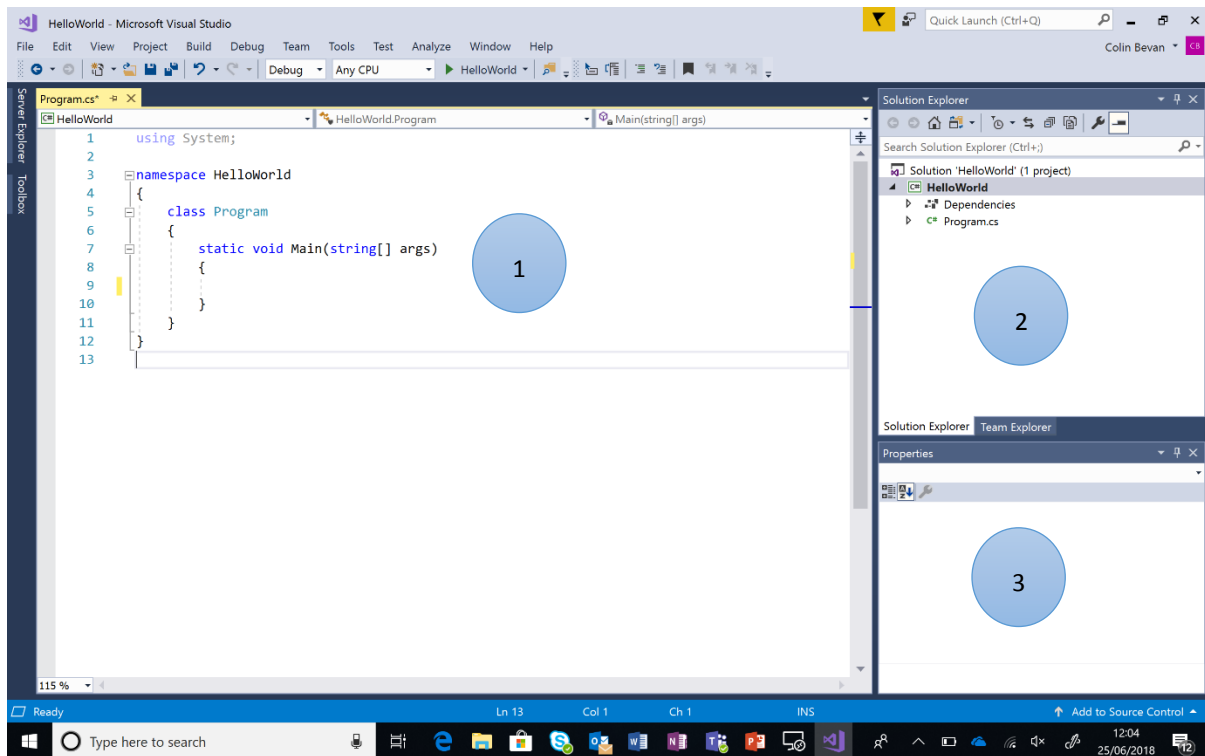
Change the default Name from ConsoleApplication1 to HelloWorld

Change the location of the application to your USB or network drive.

Click the OK button.



You will see a windows that looks like the following figure.



Area 1 – The editor is where we will write or program code.

Area 2 – The Solution Explorer shows all the components of the application.

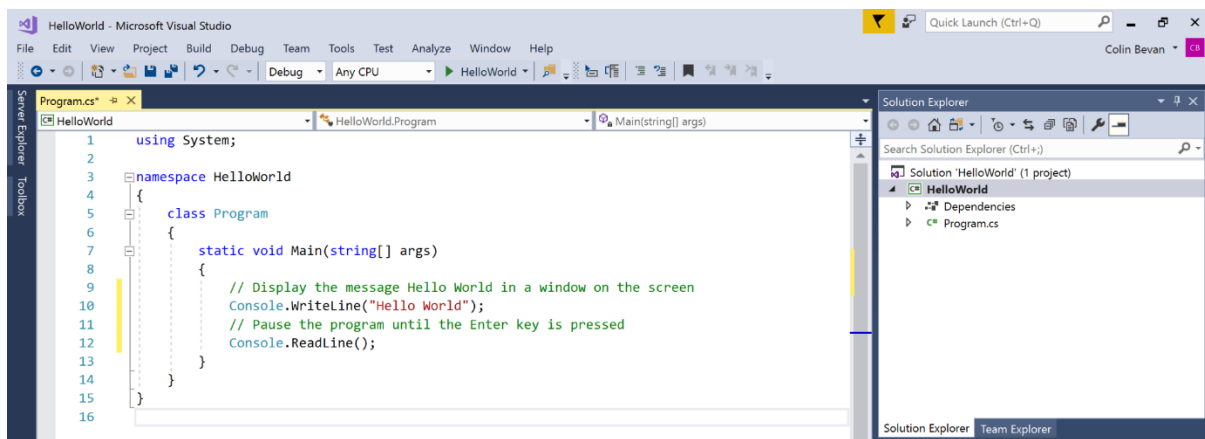
Area 3 – The Properties Window show the properties associated with objects used in the application.

YOUR FIRST PROGRAM

Now that you are familiar with the C# Environment, we will go ahead and start programming in it. Like we just noted above, the editor is the place where we write our programs.

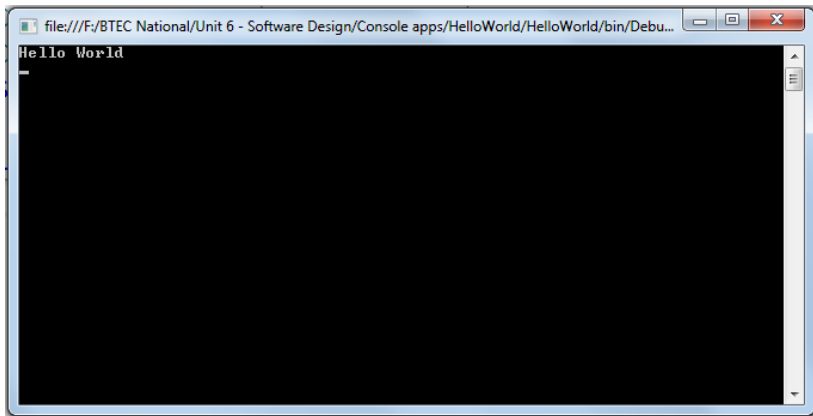
Everybody starts programming by writing a program called Hello World which displays the words “Hello World” on the screen.

So let’s go ahead and type the following lines in the editor between the { and } brackets under static void Main.



You have to write the commands exactly as shown above. If you spell something wrong, enter the wrong command, leave a letter out a blue line will appear below the command. This lets us know that the syntax is wrong. C# is case sensitive, so you need to use capital letters correctly. Statements are terminated with a ;

Now that we have typed our new program, let's go ahead and run it to see what happens. We can run our program by clicking on the *START* button on the toolbar. If everything goes well, our program should run with the result as shown below.



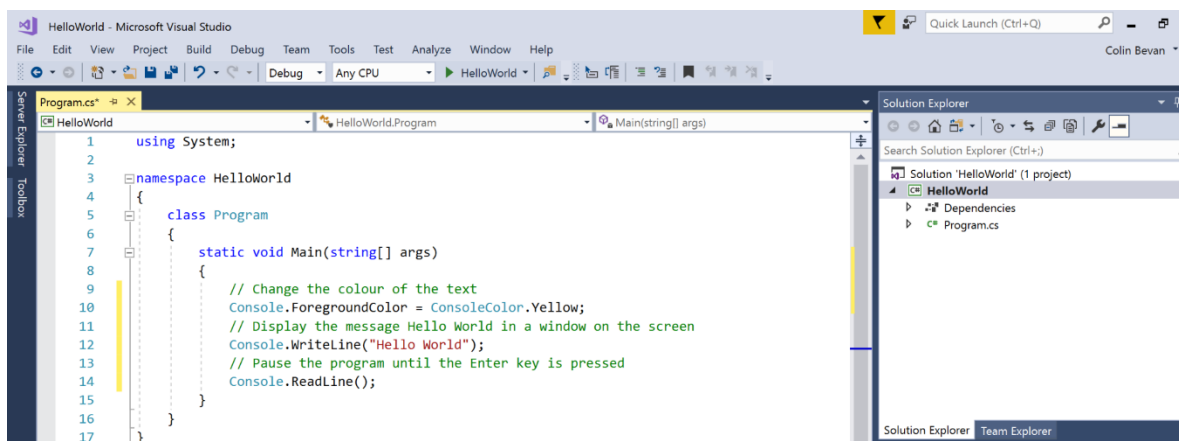
Congratulations! You have just written and run the first C# program. A very small and simple program, but nevertheless a big step towards becoming a real computer programmer!

SAVING YOUR PROGRAM

If you want to close C# and come back later to work on the program you just typed, you can save the program. It is in fact a good practice to save programs from time to time, so that you don't lose information in the event of an accidental shutdown or a power failure. You can save the current program by either clicking on the "save" icon on the toolbar or by selecting FILE and SAVE ALL in the drop down menu.

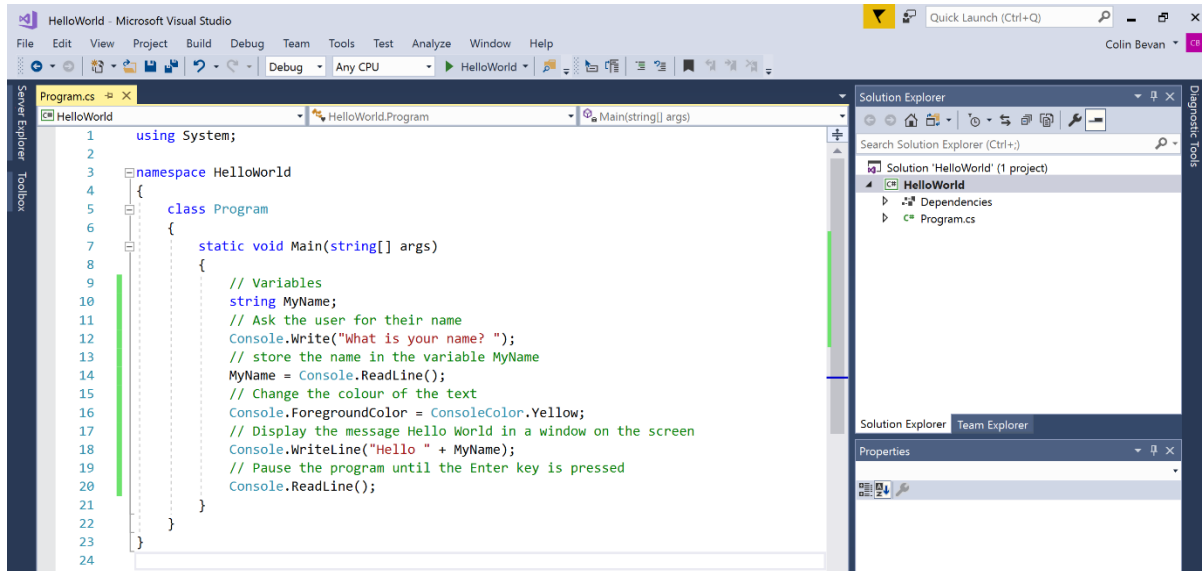
ADDING MORE FEATURES

Now that you have understood our first program, let's go ahead and make it fancier by adding some colours.



INTRODUCING VARIABLES

Wouldn't it be nice if our program can actually say "Hello" with the users name instead of saying the generic "Hello World?" In order to do that we must first ask the user for his/her name and then store it somewhere and then print out "Hello" with the user's name. Let's see how we can do that:

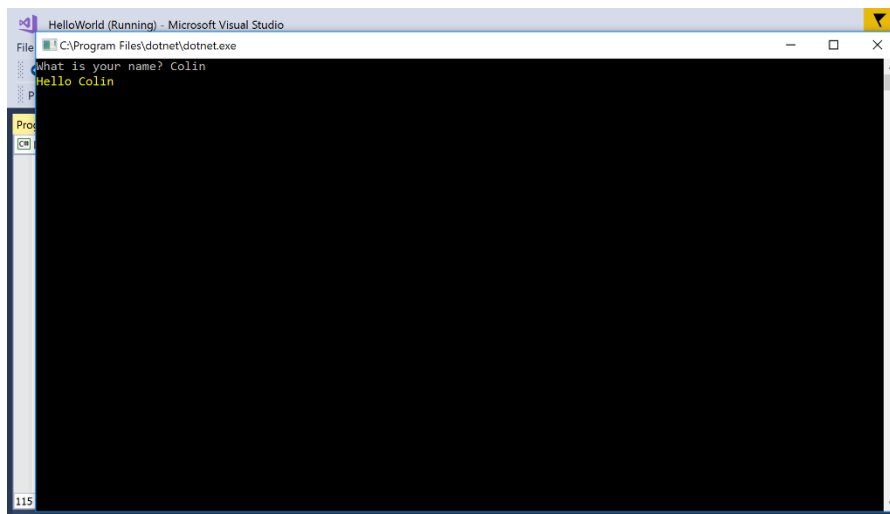


```

1 using System;
2
3 namespace HelloWorld
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             // Variables
10            string MyName;
11            // Ask the user for their name
12            Console.WriteLine("What is your name? ");
13            // store the name in the variable MyName
14            MyName = Console.ReadLine();
15            // Change the colour of the text
16            Console.ForegroundColor = ConsoleColor.Yellow;
17            // Display the message Hello World in a window on the screen
18            Console.WriteLine("Hello " + MyName);
19            // Pause the program until the Enter key is pressed
20            Console.ReadLine();
21        }
22    }
23 }
24

```

When you type and execute this program, you'll see an output like the following:



```

C:\Program Files\dotnet\dotnet.exe
What is your name? Colin
Hello Colin
P

```

The statement `string MyName;` sets up a variable of string type. A string data type can hold any characters; number, letters, hyphen etc.

The `MyName = Console.ReadLine();` command allows the user to enter a name and this is stored in the variable `MyName`.

Placing a `//` in front of a statement changes into a comment; it turns green. Comments are used to improve the readability of programs.

The command `Console.WriteLine("Hello " + MyName)` uses the `+` symbol to combine two elements.

RULES FOR VARIABLE NAMES

Variables have names associated with them and that's how you identify them. There are certain simple rules and some really good guidelines for naming these variables. They are:

1. The name should start with a letter and should not collide with any of the keywords like **if**, **for**, **then**, etc.
2. A name can contain any combination of letters, digits and underscores.
3. It is useful to name variables meaningfully – since variables can be as long as you want, use variable names to describe their intent.

C# contains reserved words, that have special meaning for the compiler. These reserved words are called "keywords". Keywords cannot be used as a name (identifier) of a variable, class, interface, etc.

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.

Example: Data types

```
class Program
{
    static void Main(string[] args)
    {
        string stringVar = "Hello World!!";
        int intVar = 100;
        float floatVar = 10.2f;
        char charVar = 'A';
        bool boolVar = true;
    }
}
```

Each data types includes specific range of values. For example, a variable of int data type can have any value between -2,147,483,648 to 2,147,483,647. The same way, bool data type can have only two value - true or false.

There are no issues moving data from one variable to another with the same data type. If the variables have different data types then you need to convert the data when it is moved from one variable to another. For example:

```
int Money;
Money = Console.ReadLine();
```

This will display an error message because the ReadLine return a string variable and Money is an integer variable. You need to convert it to integer. The correct statement is:

```
Money = int.Parse(Console.ReadLine());
```

ACTIVITY 1

1. Write a console application that asks the user for their name and address (street, area and town) and displays the information as shown below:

Colin Bevan,
1234, Long Street, Cadoxton, Neath,
SA11 7RQ

2. Write a console application which asks the user to enter two names and their ages of the two people. The application displays their names, ages and total age.