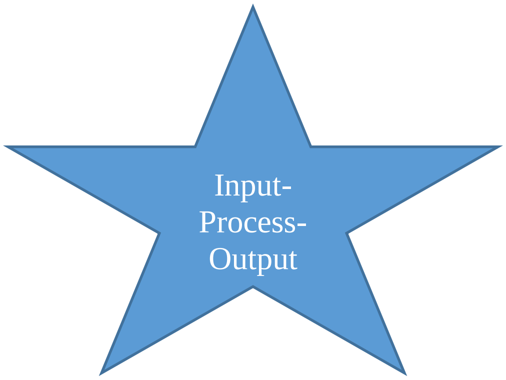

Notes on reading input from the user

Reading input text from the user

The first programs that we wrote just printed a message. It did not receive any input from the user. The next program we write is going to read some user input.

When you want a C# program to read a message from the keyboard you have to set up a place where the message can be stored. This message will be a string of text, and so we use a **`string`** data type to store it in, as in the program below:

```
using System;

class Program
{
      static void Main()
      {
            // declare variables
            string name;

            // input data
            Console.Write("Enter your name : ");
            name = Console.ReadLine();

            // output results
            Console.WriteLine("Hello " + name );
      }
}
```

Experiment with **`Write()`** and **`WriteLine()`**. See if you can describe the difference between the two.

Reading input numbers from the user

We are now going to write a program which can read numbers from the user and perform calculations. C# regards strings of text, for example **‟Hello world″**, as quite different from numeric values.

If you want your program to read a number from the user you have to read a string of text and then convert that string into a value. C# has something called **‟Parse″** which takes a string and converts it into an integer value.

It is something that the **int** class can do for us. In the next program we are going to use this to enable us to write a program which reads in two numbers and then prints out their sum.

```
using System;

class Program
{
      static void Main()
      {
            string number1Text,number2Text;
            int number1,number2,result;

            //prompt user - I like to use Write
            // since it puts input and response from user on the same
            // line
            Console.Write("First Number : ");
            //read in the input and store in
            //the variable number1Text
            number1Text = Console.ReadLine();

            // convert to numeric format
             int number1 = int.Parse(number1Text);

            Console.Write("Second Number : ");
            number2Text = Console.ReadLine();
            number2 = int.Parse(number2Text);

            result = number1 + number1;
            Console.WriteLine("Sum is : " + result );
      }
}
```

We will cover what each line of the program does in more detail later in the course. Note that I have added comment lines so that you can see what each part of the program does.

Note when could shorten this program by coding the input instructions as follows

```
Console.Write("First Number : ");
// read and convert on the same line.
number1 = int.Parse(Console.ReadLine());
```

Some programmers love to make their programs as small as possible, I think the readability/understandability of the program is more important – for now just be aware that there is often more than one way to achieve the same result – but the most important thing for you is that **you understand all the steps**

Debugging and Testing

If the program that you have typed in seems NOT to work OK, then you must have not copied it exactly.

You might have a syntax error – haven't followed the rules correctly, e.g, missing a ';'

Or

You might have a logical error - coded the incorrect steps- e.g multiplied when you should have added

In the case of the above program it sometimes gets the answers wrong.

Before you go any further; perform the following:

1. Make sure that your program is *exactly* the same as the one above.
2. Compile and run it.
3. Devise some test data. Some of the test data should cause the program to fail. Other values should cause the program to appear to work.
4. Fix the bug in the program and test it with the data you have devised.

Whenever we write a program from now on you will be required to create test data for the program. In fact, it is a great idea to the tests *before* you create the program itself! – why is this ?