Our First Programs

C# Programming

Programs

- In the practical session last week we created a number of programs
- In this session we are going to look at each program and see how it works/what it does
- We are also going to investigate how programs store and work with data

Hello World

```
using System;

class Hello
{
      static void Main()
       {
             Console.WriteLine("Hello World");
      }
}
```

- This program just prints Hello World

Hello World

```
using System;

class Hello
{
      static void Main()
       {
            Console.WriteLine("Hello World");
      }
}
```

- This is the statement that does the work

Greeter

```
using System;

class Greeter
{
   static void Main()
   {
        Console.Write("Enter your name : ");
        string name = Console.ReadLine();
        Console.WriteLine("Hello " + name );
   }
}
```

- This program reads in something, stores it and then writes it out again

Greeter

```
using System;

class Greeter
{
   static void Main()
   {
        Console.Write("Enter your name : ");
        string name = Console.ReadLine();
        Console.WriteLine("Hello " + name );
   }
}
```

- Write a message to ask the user a question

UNIVERSITY OF Hull

Greeter

```
using System;

class Greeter
{
   static void Main()
   {
        Console.Write("Enter your name : ");
        string name = Console.ReadLine();
        Console.WriteLine("Hello " + name );
   }
}
```

- Read the message and store it in a string variable called name

Greeter

```
using System;

class Greeter
{
   static void Main()
   {
        Console.Write("Enter your name : ");
        string name = Console.ReadLine();
        Console.WriteLine("Hello " + name );
   }
}
```

- This creates a variable – a "box" in memory to hold strings
- The box is called "name"

Greeter

```
using System;

class Greeter
{
   static void Main()
   {
        Console.Write("Enter your name : ");
        string name = Console.ReadLine();
        Console.WriteLine("Hello " + name );
   }
}
```

- This reads a line of text that the user types in

Greeter

```
using System;

class Greeter
{
   static void Main()
   {
        Console.Write("Enter your name : ");
        string name = Console.ReadLine();
        Console.WriteLine("Hello " + name );
   }
}
```

- This is an *assignment*
- It takes the value on the right and assigns it to the variable on the left

Greeter

```
using System;

class Greeter
{
   static void Main()
   {
        Console.Write("Enter your name : ");
        string name = Console.ReadLine();
        Console.WriteLine("Hello " + name );
   }
}
```

- Write the word "Hello" followed by the contents of the variable called name

How Programs Work

- The bit of the program that actually does the work is contents of the Main method
- This is a sequence of *statements* each of which does one particular step
- The program performs each statement in turn and then ends after it has completed the last statement in the program

Sums

```
static void Main()
{
    Console.WriteLine("This program adds two numbers together");
    Console.Write("First Number : ");
    string number1Text = Console.ReadLine();
    int number1 = int.Parse(number1Text);
    Console.Write("Second Number : ");
    string number2Text = Console.ReadLine();
    int number2 = int.Parse(number2Text);
    int result = number1 * number2;
    Console.WriteLine("Sum is : " + result );
}
```

- This program reads in two numbers, adds them together and then prints out the result

Sums

```
    static void Main()
    {
         Console.WriteLine("This program adds two numbers together");
         Console.Write("First Number : ");
         string number1Text = Console.ReadLine();
         int number1 = int.Parse(number1Text);
         Console.Write("Second Number : ");
         string number2Text = Console.ReadLine();
         int number2 = int.Parse(number2Text);
         int result = number1 * number2;
         Console.WriteLine("Sum is : " + result );
    }
```

- Say Hello

Sums

```
    static void Main()
    {
         Console.WriteLine("This program adds two numbers together");
         Console.Write("First Number : ");
         string number1Text = Console.ReadLine();
         int number1 = int.Parse(number1Text);
         Console.Write("Second Number : ");
         string number2Text = Console.ReadLine();
         int number2 = int.Parse(number2Text);
         int result = number1 * number2;
         Console.WriteLine("Sum is : " + result );
    }
```

- Ask for the number

Sums

```
    static void Main()
    {
         Console.WriteLine("This program adds two numbers together");
         Console.Write("First Number : ");
         string number1Text = Console.ReadLine();
         int number1 = int.Parse(number1Text);
         Console.Write("Second Number : ");
         string number2Text = Console.ReadLine();
         int number2 = int.Parse(number2Text);
         int result = number1 * number2;
         Console.WriteLine("Sum is : " + result );
    }
```

- Read in a string of text (we can't read numbers directly)

Sums

```
    static void Main()
    {
         Console.WriteLine("This program adds two numbers together");
         Console.Write("First Number : ");
         string number1Text = Console.ReadLine();
         int number1 = int.Parse(number1Text);
         Console.Write("Second Number : ");
         string number2Text = Console.ReadLine();
         int number2 = int.Parse(number2Text);
         int result = number1 * number2;
         Console.WriteLine("Sum is : " + result );
    }
```

- Use a method called Parse to convert the text into a numeric value

Sums

```
    static void Main()
    {
         Console.WriteLine("This program adds two numbers together");
         Console.Write("First Number : ");
         string number1Text = Console.ReadLine();
         int number1 = int.Parse(number1Text);
         Console.Write("Second Number : ");
         string number2Text = Console.ReadLine();
         int number2 = int.Parse(number2Text);
         int result = number1 * number2;
         Console.WriteLine("Sum is : " + result );
    }
```

- Repeat for the second number

Sums

```
    static void Main()
    {
         Console.WriteLine("This program adds two numbers together");
         Console.Write("First Number : ");
         string number1Text = Console.ReadLine();
         int number1 = int.Parse(number1Text);
         Console.Write("Second Number : ");
         string number2Text = Console.ReadLine();
         int number2 = int.Parse(number2Text);
         int result = number1 * number2;
         Console.WriteLine("Sum is : " + result );
    }
```

- Do the sum (anyone spot the bug?)

UNIVERSITY OF Hull

Sums

```
static void Main()
{
     Console.WriteLine("This program adds two numbers together");
     Console.Write("First Number : ");
     string number1Text = Console.ReadLine();
     int number1 = int.Parse(number1Text);
     Console.Write("Second Number : ");
     string number2Text = Console.ReadLine();
     int number2 = int.Parse(number2Text);
     int result = number1 * number2;
     Console.WriteLine("Sum is : " + result );
}
```

- Write out the answer

Finding the Bug

- The bug does not cause the program to crash
  - It will always produce an answer when given two numbers
- However, the answer is wrong, because the user is expecting to see an addition, and they are being given a multiplication

Sums

```
    static void Main()
    {
         Console.WriteLine("This program adds two numbers together");
         Console.Write("First Number : ");
         string number1Text = Console.ReadLine();
         int number1 = int.Parse(number1Text);
         Console.Write("Second Number : ");
         string number2Text = Console.ReadLine();
         int number2 = int.Parse(number2Text);
         int result = number1 + number2;
         Console.WriteLine("Sum is : " + result );
    }
```

- To fix the program we just have to swap the * (multiply) for a + (add)

Test Data

- The scary thing is that the program will work fine for some inputs:
  - `0+0` is the same as `0*0`
  - `1+1` is the same as `1*1`
  - `2+2` is the same as `2*2`
- When we create a program we need to design some proper tests to prove that it works

Adding 4 numbers

```
int v1, v2, v3, v4;

v1 = int.Parse(Console.ReadLine());
v2 = int.Parse(Console.ReadLine());
v3 = int.Parse(Console.ReadLine());
v4 = int.Parse(Console.ReadLine());

int sum = v1 + v2 + v3 + v4;
```

- This program calculates the sum of four numbers
- It works by “scaling up” the earlier solution

Adding 4 numbers

```
int v1, v2, v3, v4;

v1 = int.Parse(Console.ReadLine());
v2 = int.Parse(Console.ReadLine());
v3 = int.Parse(Console.ReadLine());
v4 = int.Parse(Console.ReadLine());

int sum = v1 + v2 + v3 + v4;
```

- A program can create lots of variables at once by using a list

Adding 4 numbers

```
int v1, v2, v3, v4;

v1 = int.Parse(Console.ReadLine());
v2 = int.Parse(Console.ReadLine());
v3 = int.Parse(Console.ReadLine());
v4 = int.Parse(Console.ReadLine());

int sum = v1 + v2 + v3 + v4;
```

- A program can feed the output of one method call into another

Is this a good solution?

- We can see how it works
- We could add 1,000 numbers this way
  - But it would be very tedious
- Is there a better way that avoids the need for v1, v2, v3 and the rest?

A better solution

```
int sum = 0;

sum = sum + int.Parse(Console.ReadLine());
sum = sum + int.Parse(Console.ReadLine());
```

- This makes the program much simpler
- We don’t need all the v1, v2, v3 stuff
- We just increase the sum by each number as it is reads in

A perfect solution?

- This is not a perfect solution
- To add 1,000 numbers we would still need 1,000 statements
  - A better way would be to use a loop, which we will see next week
- We have learnt that sometimes simple solutions don’t scale very well

More Numbers

```
int v1, v2, v3, v4;

v1 = int.Parse(Console.ReadLine());
v2 = int.Parse(Console.ReadLine());
v3 = int.Parse(Console.ReadLine());
v4 = int.Parse(Console.ReadLine());

int sum = v1 + v2 + v3 + v4;
```

- To fix the program we just have to swap the * (multiply) for a + (add)

Averages

```
int average = sum / 4;
```

- We can use our program to work out the average of the values
- However, this doesn’t end well when we use integers

What is an integer?

- An integer is used for counting
- We could use it to count how many sheep are in a field
  - We never need to store fractions of sheep, and so we don’t need any fractions
- We can save computer memory and keep programs simpler by just storing the number part and leaving off the fraction

Problems with integers

```
int average = sum / 4;
```

- We get problems when we try to work out sums that need fractional parts
  - For example the average weight of a sheep
- Integers are no good for us
- We can use floating point values instead

Converting to floats

```
float average = sum / 4;
```

- C# provides a type called float that we can use in just the same way as integer
- But it stores a fractional part as well
- This means that we can get more precise results – if we need them

Things to Remember

- Programs run one statement at a time
- They can store data in named boxes called “variables”
- The Parse method will convert a string of digits into a numeric value
- Integers do not hold fractional parts, but floats do