C# Data Types

In C# data is assigned to variables. These are given useful names so they can be manipulated in code. The variables are divided into data types; each specialised for a particular purpose. A number behaves differently to a word. Numbers themselves can behave in different ways such as whole or decimal numbers.

Whole number data types

This is quite a list but int and long are the most commonly used.

Type	Range	Size
sbyte	-128 to 127	Signed 8-bit integer
byte	0 to 255	Unsigned 8-bit integer
char	U+0000 to U+ffff	Unicode 16-bit character
short	-32,768 to 32,767	Signed 16-bit integer
ushort	0 to 65,535	Unsigned 16-bit integer
int	-2,147,483,648 to 2,147,483,647	Signed 32-bit integer
uint	0 to 4,294,967,295	Unsigned 32-bit integer
long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed 64-bit integer
ulong	0 to 18,446,744,073,709,551,615	Unsigned 64-bit integer

Variables are declared as follows:

```
int myNum;
```

They can be assigned a value with =

```
myNum = 21;
```

It is good practiced to declare a variable and assign it an initial value in 1 line.

```
int myNum =0;
```

If the variable is assigned a value that is not within range the code will throw an error.

```
int myNum = 20000000000;
// noooooooooooooo
```

The unsigned numbers do not accept a negative number. They do not crash but the results might be unexpected.

```
uint myNum = 2;
myNum = myNum - 3;
//what is myNum now ?
```

Naturally being whole numbers they are not overly happy about accepting decimals

```
int myNum = 2.222;
//this code will not run
```

Floating point data types

Type	Approximate range	Precision
float	±1.5e−45 to ±3.4e38	7 digits
double	±5.0e−324 to ±1.7e308	15-16 digits

```
float myFloat = 2.222;
//this is not good
```

The correct way to declare floats and doubles is as follows:

```
float myFloat = 2.222f;
double myDouble = 2.22222222222222222222d;
```

Both will accept whole numbers but this should only be done if necessary as floating point numbers are big but not precise.

```
float myFloat = 0.0f;
```

Maths

Having got some numbers to play with we can try out some maths.

```
int a, b, c;
a = 1;
b = 2;
c = a + b;
Console.WriteLine(c);
```

A more helpful output would be

```
Console.WriteLine("a added to b gives c, which is " + c);
```

Any output within " " is written as words not values:

```
Console.WriteLine("c");
//is not the same as
Console.WriteLine(c);
```

The basic maths operators in C# are:

- +
- -
- /
- *

The following are also useful

- % modulus or remainder
- ++ add one such as x++
- -- subtract 1

+= and -= do work with C#.

Maths in C# follows the BODMAS rules, ( ) can be used with formulae.

More exciting operations can be found from `System.Math` such as square root.

```
double c = System.Math.Sqrt(4);
```

Trigonometry functions such as Sin work with radians not degrees.

When converting degrees to radians a circle (360^{o}) is 2π
degrees = radians * 180 / π

Non-numeric data types

Boolean data types can only be true or false. This is a surprisingly useful concept to assess.

```
bool good = true;
bool bad = false;
```

Words are seen as strings of characters. The char data type consists of 1 and only 1 letter.

```
char letter = 'a';
//this is not going to be good
char badLetter = 'aa';
```

A string of characters can hold a word or lots of words.

```
String word = "hello";
```

Note the `''` to hold a char but `" "` to hold a string.

There are a number of useful built in functions to work on words, such as finding one word inside a set of other words.

Strings can be strung together for output in `Console.WriteLine`

```
String word = "hello";
String otherWord = "goodbye";
Console.WriteLine("I say " +  word + "and you say " + otherWord  );
```

The same concept works to put one string into another

```
String beatlesLyric = "I say " +  word + "and you say " + otherWord
;
```