# Y11 -> Y12 Transition

# Overview

- Installing Visual Studio
- Creating your first Visual Studio programs
- Newspaper article – Origins of programming languages
- Course book & possible reading list

- C# .Net framework

# Installing visual studio

- You want to ensure you install the C# part of the visual studio for definite!

# Programming

```
//If statements
string name;
Console.WriteLine("What is your name?");

name = Console.ReadLine();
if (name == "Bob")
{
    Console.WriteLine("Hi bob");
}
else
{
    Console.WriteLine("I'm not talking to you");
}
```

Read through and complete the exercises on the data types worksheet

- **Under age problem**
- Difficulty: 🛠
- Write a program that asks for your age.  If you are over 18 it outputs the message, “Over 18”, otherwise it outputs, “Under age”.
- **Water temperature problem**
- Difficulty: 🛠
- Write a program that reads in the temperature of water in a container in Centigrade and displays a message stating whether the water is frozen (zero or below), boiling (100 or greater) or neither.

- **Vocational grade problem**
- Difficulty: 🛠
- Write a program that allows you to enter a test mark out of 100. The program outputs “FAIL” for a score less than 40, “PASS” for a score of 40 or more, “MERIT” for a score of 60 or more and “DISTINCTION” for a score of 80 or more.

- **Extended visual dice problem**
- Difficulty: 🛠
- For a six sided dice, write a program that asks for a number and outputs that number as a graphical dice. E.g.
-
- **ooooooooooo**
- **o o**
- **o # o**
- **o # o**
- **o # o**
- **o o**
- **ooooooooooo**
-

# Computing
# Y11/12 Transition work

There are many different high-level languages in the world today.

Create a newspaper article on one of these languages.

Your articles should cover the following areas:

- The platform for which your language was originally designed
- The origins of the language
- The main use of the language
- Some example code created in this language (annotated to show the reader how the code operates)
- Details of any special features the language has.
- Some indication of the popularity of the language
- How portable the software that is created using this language is
- The type of user interface that is used to control programs written in this language
- Any drawback with this language.

<u>Extension:</u>

1. Compare some of the more popular high-level languages that are used in school and colleges today. Your comparison should cover the following:
   - Ease of use
   - Versatility
   - Cost
2. What prompts the development of high level programming languages? Research into the origins of languages such as Java and Delphi and find out how they developed.

# Reading list

- If any of you would like to do any further reading over the holidays for your own personal enjoyment then I have included a reading list on the next two slides
- It is worth checking online, libraries and  online second hand book shops (i.e https://www.abebooks.co.uk/) for cheap versions

- The course book is **AQA AS and A Level Computer Science** P M Heathcoate (2016). I highly recommend you get yourself a copy – it will be invaluable over the next 2 years -  I still have mine from my A levels!
- https://www.amazon.co.uk/AQA-AS-Level-Computer-Science/dp/1910523070

# Reading List

- *Computational Fairy Tales* by Jeremy Kubica. ISBN: 978-1477550298 - a romp through the principles of computational thinking, illustrating high-level computer science concepts, the motivation behind them, and their application via the medium of a fairy tale. Aimed at secondary school students. "Bonkers, but very enjoyable."
- *Computer Science: An Overview* by J. Glenn Brookshear. ISBN: 978-0321544285 - overview of what computer science is all about: each topic is presented with its historical perspective, current state, and future potential, as well as ethical issues.
- *Code: The Hidden Language of Computer Hardware and Software* by Charles Petzold. ISBN: 978-0735611313 - "What do flashlights, the British invasion, black cats, and seesaws have to do with computers? ...see how ingenuity and our very human compulsion to communicate have driven the technological innovations of the past two centuries."
- *Out of Their Minds* by D Shasha and Cathy Lazere. ISBN: 978-3540979920 - the lives and discoveries of fifteen unsung computer scientists whose programs have helped people from factory owners to cartoonists.
- *The Pattern on the Stone: The Simple Ideas That Make Computers Work* by Daniel Hillis. ISBN: 978-0465025961 - explains the basic concepts of the computer in everyday language.

# Reading List

- *The Information: A History, a Theory, a Flood* by James Gleick. ISBN: 978-0007225736 - a chronicle that shows how information has become "the modern era's defining quality - the blood, the fuel, the vital principle of our world."
- *The Pleasures of Counting* by Tom Kôrner. ISBN: 978-0521568234 - puts Maths into the context of how it is used to solve real-world problems.
- *The Code Book* by Simon Singh. ISBN: 978-1857028898 - not strictly about Computer Science, but an interesting introduction to code-breaking and cryptography, fields that have a strong connection to Computer Science.
- *The New Turing Omnibus* by A Kee Dewdney. ISBN: 978-0805071665 - mini articles on Computer Science topics.
- *Algorithmic Puzzles* by Anany Levitin and Maria Levitin. ISBN: 978-0199740444 - "...The emphasis lies in training the reader to think algorithmically and develop new puzzle-solving skills: the majority of puzzles are problems where we are asked to find the shortest distance or the fewest moves to get from A to B, or construct a proof that a puzzle has no solution ..."

**ALL - know that variables store data and there are different types of data.**
**MOST - choose a correct data type for a given task and assign data to a variable.**
**SOME - explore casting and some of the potential problems when dealing with data types.**

## 1. Data Types

All programming languages have a variety of data types. Fortunately C# has virtually every data type you can imagine. That means if you can learn all about them now you can easily pick up any other language later on in life.

Whenever some data is stored by the computer it needs to know what type of data it is so that it can allocate a section of memory that is the correct size. Think of it like packing items into boxes. You need to know the thing you are going to put in a box before you know what size of box you are going to get!

For us humans it is much more difficult to remember the number 3.4537164738712376 than it is the number 2. This is the same for a computer as well. A computer needs more memory space to store a number with a lot of digits than it does for a simple number.

There are lots of different data types you can choose from in C#. For those of you that are interested you can see a full list here.

For this A-level course however we are going to keep it simple. The ones you need to know are in the table below:

| Data Type | Number of Bytes | What it Can Store |
|---|---|---|
| int | 4 | Stands for **integer**. It can store any **whole number** (a number without a decimal point). |
| double | 8 | Any **floating point number** (a number with a decimal point). |
| bool | 1/2 | Stands for **boolean**. It can store either **true** or **false**. |

| | | |
|---|---|---|
| string | 16 | Any combination of characters. Normally a word or sentence. |
| char | 2 | Any **unicode character**. |

1. If you wanted to store your name what data type would you use?
2. If you needed to store the value of Pi which data type would you use?
3. If you needed to store somebody's age what data type would you use?.
4. You are making a ticket system and want to store whether or not somebody is a student. Which data type would you use?

### 2. Variables

Once we know what we want to store and worked out our data type then we need to give it a name! If we do this is called a **variable** and we can start to store things in it.

Let's say we are going to store the number of cats that a cat lover has. We could use any abstract name for our variable like **x** or **giraffe** but it makes sense to make our names meaningful so lets just use **cats**.

We can now **declare** our variable and **assign** a value to it. Like this:

```
int cats;



cats = 8;

Console.WriteLine("The number of cats you own is: " + cats);
```

1. Try out the example code above.
2. You are going to store the number of pupils in your class. Decide on a data type, think of an appropriate name, declare and assign the variable then output it to the screen.
3. A cricketer wants to store his batting average of 43.58. Decide on a data type, think of an appropriate name, declare and assign the variable then output it to the screen.
4. Make a variable that will store somebody's name. Assign it a value then display it to the screen.
5. You need to store whether or not a person is registered for your website. Decide on a data type, think of an appropriate name, declare and assign the variable then output it to the screen.
6. Make variables called firstname and surname. Assign them values then display the person's full name to the screen using only one command.

7. Make 2 integer variables called number1 and number2. Assign them whatever value you like. Display the sum (add them up) of the two numbers on the screen.
8. Make 4 double variables and give them a value. Output the product of those values (multiply them together).
9. Create 2 integer variables called number1 and number2. Assign them the values 10 and 3 then display the outcome of number1 / number2. What happens? Fix it so that the outcome is correct.
10. Write a program that takes a Fahrenheit temperature and converts it to a Celcius temperature. The output should look like this: "The Fahrenheit temperature x is equal to the Celsius temperature x".

### 3. Casting

Sometimes you might need to convert a value from one data type to another. This is called **casting**. There are a few ways to do this but one simple way is to use Convert.

To convert to a string you would use Convert.ToString(). To convert to a double you would use Convert.ToDouble(). To convert to an integer you would use Convert.ToInt32(). Here is an example:

```
string x;

int y;


x = "37";

y = Convert.ToInt32(x);


Console.WriteLine(y);
```

To see an easy-to-use guide to typecasting click here.

1. Try the method above
2. Change the value of x to be "37.5" instead of "37" then run the code again. What happens? Why?
3. Alter the code so that you are converting strings to ints, ints, to doubles, doubles to strings and so on. Chances are you will encounter some problems. If you don't then try all manner of values until you get some problems.

**Challenges**

1. Find out why the double data type is not suitable for handling monetary values e.g. for an accountant. What data type should be used instead?
2. Follow the link near the top of the page to the other C# data types. Think of a suitable piece of data for each type.

## Under age problem

Difficulty: 🛠

Write a program that asks for your age. If you are over 18 it outputs the message, “Over 18”, otherwise it outputs, “Under age”.

## Water temperature problem

Difficulty: 🛠

Write a program that reads in the temperature of water in a container in Centigrade and displays a message stating whether the water is frozen (zero or below), boiling (100 or greater) or neither.

## Vocational grade problem

Difficulty: 🛠

Write a program that allows you to enter a test mark out of 100.
The program outputs “FAIL” for a score less than 40, “PASS” for a score of 40 or more, “MERIT” for a score of 60 or more and “DISTINCTION” for a score of 80 or more.

## Extended visual dice problem

Difficulty: 🛠

For a six sided dice, write a program that asks for a number and outputs that number as a graphical dice. E.g.