# First Year Computing (C++ Course)
Dr John Joss Chinn

Week 2.

## Structure of a C++ Program, Basic Maths and Loops

### Structure of a C++ Program

The basic structure of a simple C++ program is as follows:-

```
Headers

Main function
```

It does get a little more complicated than this as we go along.

As mentioned briefly last week, the headers contain extra code, which is needed for your C++ program to work. Unlike high-level languages, which put in ALL of the code that your program is ever going to need, in C++ you must find out which headers you need to include. In C and early versions of C++ the headers were simply names of files, which contained the extra code. On the computers in GB/B4 go to `m:\borlandc\include`, which is an old version of the Borland C++ compiler, and you will see a list of header files, all with the file extension `.h`. You can view these by opening them in NOTEPAD (`start→programs→accessories→notepad`). You will see that these files do indeed contain C++ coding. We will look at which headers need to be included for which keywords as we progress through the course. Examples of headers are:

| | |
|---|---|
| `iostream` | needed for input and output |
| `math` | needed for the mathematical keywords |
| `fstream` | needed for input and output to text files |
| etc | |

There are probably several hundred different headers. Don’t worry, we will only use about a dozen in this course!

Every C++ program contains a `main() { }` function. All but the simplest program contains several other functions. For the first few weeks of this course we will look at programs which contain ONLY a main function.

The structure of a C++ computer program then is something that you will learn as we progress through the course.

### Types of Variable

On page 11 of last weeks notes we looked at a little program (`io1.cpp`) which reads in some numbers, does some little sums and then prints out the answers. The numbers were read into variables. The variables were `a, b, c` etc. This is a bit like algebra. In `io1.cpp` only two types of variables were used, integer (`int`) and floating point real (`float`). In a C++

program you have to DECLARE the variables which you are going to use. You also have to do this in FORTRAN, but in BASIC it is up to you, i.e. it is arbitrary but good practice. There are essentially five types of variables:-

| | |
|---|---|
| `int` | integer |
| `float` | floating point real |
| `double` | double precision floating point |
| `char` | character or string |

The variable type `double` is similar to `float`, it just lets you work with greater precision. Some C++ text books advocate that you don't bother with `float` at all, just use `double` for all floating point numbers. Let us look at some examples of characters and strings. The following program (`char1.cpp`) reads in a character from the keyboard and then puts it up on the screen.

```
#include<iostream>
#include<conio.h>
#include<stdio.h>

void main()
{
      char key;

      cout<<"Press a Key"<<endl;
      key = getch();
      cout<<"Ouch!, you pressed the ";
      putchar(key);
      cout<<" key"<<endl;
      getch();
}
```

Firstly we need to declare the variable `key` as a `char`. In other words the variable `key` will accept a character as a variable, i.e. any character. The words `Press a Key` appear on the screen. The line `key = getch();` is the part which grabs the character, which the user presses. The keyword `getch()`, which stands for GET CHARACTER, needs the `conio.h` header for it to work. If you try to compile the program with out it you will get an error. See pages 8 and 9 for how to compile and run source code. The line `putchar(key)` 'PUTS' the CHARACTER on to the screen. This needs the header `stdio.h`. This is actually the old C type input output header. The final line, `getch()`, which we have seen in `hello.cpp` and `io1.cpp`, from last week, just waits for a key press before continuing. Try missing out this last line and see what happens!

This next program (`string1.cpp`) behaves in a similar way to `char1.cpp` but deals with a string of characters, rather than just one character.

```
//#include "stdafx.h"  /*needed in Microsoft C++*/
#include<iostream>
#include<conio.h>
#include<stdio>


//using namespace std;  /*needed in Microsoft C++*/

void main()
{
      char kitty[20];

      cout<<"what would you like to name your cat? "<<endl;
      gets(kitty);
      puts(kitty);
      cout<<" is a nice name for a cat";
     getch();              /*not needed in Microsoft C++*/
}
```

There are a few things going on here. Firstly the top line `#include "stdafx.h"` and the line `using namespace std;` would be needed to run the program (any C++ program) in Microsoft C++. These lines have been "commented out" using the two forward slashes to designate them as just comments, in this Borland C++ program (`//`) . The line `char kitty[20];` declares the variable `kitty` as a string (of characters) which can take up to twenty characters. The line `gets(kitty);` collects the string from the keyboard which the user has typed. The line `puts(kitty);` puts the string up on the screen again. At some point you may wish to manipulate strings of text. This sort of routine is useful if you are to read in peoples names and exam marks from several files, for different courses say.

**Basic Maths**

In the scheme of work, given out last week, it was mentioned that we would cover some mathematical keywords, such as trigonometric and logarithmic keywords:- `y=sin(x), x=asin(y), y=cos(x), x=acos(y), y=tan(x), x=atan(y), y=log(x), y=log10(x),y=exp(x), z=pow(x,y) y=sqrt(x), y=abs(x)` and `y=fabs(x)` (absolute value of a floating point number). The appropriate header files to use are `math.h` and `stdlib.h`. I will defer covering these until we have covered loops. You may then mess about with these maths keywords as part of this weeks tutorial.

**Loops**

In C++ there are `for` loops and there are `while` loops. Take a look at the following two programs (`for1.cpp` and `while1.cpp`), which basically do the same thing.

```
#include<iostream>
#include<math.h>
#include<conio.h>

void main()
{
      int i,a;

      for(i=1;i<=10;i=i+1)
      {
        a=pow(i,3);
        cout<<"\t"<<i<<"\t"<<a<<endl;
      }
      getch();
}
```

In this program we have included the header `math.h` (you may find that just `math`, without the `.h` extension, works in Borland C++). This header is needed for the line `a=pow(i,3);` this means raise `i` to the power 3, i.e. $i^3$ . Because this occurs in a loop, which runs for i = 1 to i = 10 then the power line, and the line `cout<<"\t"<<i<<"\t"<<a<<endl,` operate ten times. In this way two columns of numbers are printed to the screen, `i` and `a`. The way that a `for` loop works is as follows. The parenthesis following the word `for` contain three arguments. The first tells the loop where to start (`i =1`), the second tells the loop to keep going until the statement is no longer true (`i<=10`), i.e. keep going until `i` is greater than 10. The third increments or decrements `i` by one each time through the loop (`i=i+1`). Two points here, you don't have to increment or decrement by 1, it can be any number. The second point is that, in C++, the short hand notation for `i=i+1` is just `i++` .

The second program looks like this

```
#include<iostream>
#include<math.h>
#include<conio.h>

void main()
{
      int i,a;
      i=1;

      while(i<=10)
      {
            a=pow(i,3);
            cout<<"\t"<<i<<"\t"<<a<<endl;
            i++;
      }
      getch();
}
```

In a `while` loop there is only one argument. In this case the argument is `i<=10`. So this means keep going until `i` is greater than 10. The setting of the initial value of `i` is done before entering the loop (`i=1`) and the incrementing is done within the loop (`i++`). In C++ all variables have to be initialised at some point. That is they must be given a value before they are first used, otherwise they just take on ANY value. You will come across this idea again.

## Week 2 Tutorial

There are several things that you need to do for this weeks tutorial, based on the work of the last two weeks:-

1. Type in the program `hello.cpp`, on page 10. Compile it and run it. Incidently, short hand for compile (in Borland C++) is `ALT F9` and shorthand for run is `CTL F9.` Try using a different text message in `hello.cpp.`
2. Type in the program `io1.cpp`, on page 11. Change the values of the variables. Try assigning the variables values within the program rather than typing them in. Make sure that you understand how `io1.cpp` works. This is really the basis for most of the work that we shall be covering.
3. Type in the program `char1.cpp` , on page 13. Run it to see how it works. Try commenting out either of the headers `#include<conio.h>` or `#include<stdio.h>.` See what error messages you get.
4. Look at `string1.cpp`, on page 14. Try duplicating some of the lines of code to take in and print out more strings of text.
5. Look at `for1.cpp` and `while1.cpp`, on page 15. Try some of the other mathematical keywords. You may find the line of code `float pi = 4*atan(1)` useful for defining π. Put it immediately after the `#include` statements. You will need this because all of the trigonometric keywords need their arguments in radians. So if you want to input a value in degree, and find the sine of it say, then you need to convert it to radians before using the keyword `y=sin(x)` .
6. Try decrementing within a loop as well as incrementing. In other words start off with `i` = 10 and then decrement by one each time (`i--`).