Week 4 & 5

C++ Programming Language

C++ is case sensitive, and all C++ keywords must be written in lower case.
Every C++ statement ends in a semicolon (;).

Data types:

The following types of data can be stored and handled by C++.

Type	*Range*		*Bytes*	*Represents*
	From	*To*		
char / short	-128	127	1	Characters
unsigned char	0	255	1	Characters
int	-32,768	32,767	2	Whole numbers
unsigned int	0	65,535	2	Whole numbers
long	-2,147,438,648	2,147,438,648	4	Whole numbers
unsigned long	0	4,294,967,295	4	Whole numbers
float	(-) 3.4 x 10-38	(-)3.4 x 1038	4	Fractional numbers
double	(-)1.7 x 10-308	(-)1.7 x 10308	8	Fractional numbers
long double	(-)3.4 x 10-4932	(-)3.4 x 104932	10	Fractional numbers

There is also a data type called void, which means no data type at all!
You may have noticed that there is no 'string' type in C++. This means we have to handle strings in a rather indirect fashion, as will be explained later.

The Escape Sequence Characters:

A character constant in a C++ program may also be an escape sequence if preceded by '\'.
Although these may appear to contain two characters, they still represent single characters.

Newline \n	Carriage return [Home] \r	Question mark \?	Octal number \ooo
Horizontal tab \t	Form feed [Top of next page] \f	Single quote \'	Hex number \xhhh
Vertical tab \v	Bell \a	Double quote \"	
Backspace \b	Backslash \\	Integer 0 \0	

The character 'Z' for example can be represented by any of the following:
Decimal: 90
Octal: '\132'
Hexadecimal: '\x5A'

Declaring Variables in a Program:

Variable is a named memory location to hold a particular type of data and the content of a variable can be changed during the execution of the program.
Variables are declared in a C++ program by following the type name with the variable name.

```
E.g.: char ch;
      int item_no;
      float x;
```

Assignment:

It just involves the equals (=) sign known as the assignment operator.

```
E.g:  int x;
      x = 4;
      int y = 10;
      y = 5;
```

Constants:

Constant is a named memory location to hold a particular type of data and the content of a constant cannot be changed during the execution of the program.
This is done with the keyword const.

```
E.g.:  const float PI = 3.141;
       const int CHAR_BUFFER = 80;
```

It is common practice to write constant names in upper case to differentiate them from variable names which are typically in lower case.

Arithmetic Operators:
Arithmetic operators are + , - , * , / and %(remainder).
The remainder operator has equal precedence with multiply and divide.

```
E.g.:  int x = 4 + 2 * 3;   //10
       x = (4 + 2) * 3;   //18
       x = 5 / 3;   //1
       int y = 5 % 3;   //2
```

Unary Operator Shorthand:
These convert what would be dyadic expressions (more than one operands) to unary.

```
E.g.:  int counter =1;
       counter++;   //counter = counter + 1;
       counter--;   // counter = counter - 1;
```

Other Expression Shorthand:
We also have shorthand for changing the value of a variable by arithmetic on its existing value.
In general terms variable_name = variable_name ? n can be replaced with variable_name ?= n where '?' means any one of the five arithmetic operators.

```
E.g.:   counter += 5;   //counter = counter + 5;
        counter -= 3;   //counter = counter - 3;
        counter *= 5;   //counter = counter * 5;
        counter /= 4;   //counter = counter / 4;
        counter %= 3;   //counter = counter % 3
```

Prefix and Postfix Notation:
Postfix notation: counter++ or counter--
Prefix notation: ++counter or --counter
In prefix notation, the operator will execute before the rest of the expression, but in postfix notation it will be executed afterwards.

```
E.g.:  int ctr = 1;
        int x = ctr++;   //x = 1 and ctr = 2
        int y = ++x;   //y = 2 and y = 2
```

Relational Operators:

= =	equal to
!=	not equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

Logical Operators:

&&	AND
||	OR
!	NOT

Functions:
C++ programs are basically just a series of functions which are able to call each other.
The general format of a C++ function is,

```
return_type function_name(data_type parameter1, data_type parameter2, …)
{
  function_body
}
```

The return type of a function is int by default, so if no return type is declared, the compiler expects the function to return an integer.
If no data is to be returned from the function, then its type is declared as void.

The parameter list may also be void, when there are no parameters to the function, and the brackets may be left empty if this is the case.
Any function which is not void must contain a return statement as the last statement in the function.
E.g.:
```
int sqr(int value_in)
        {
          int squared=value_in*value_in;
          return squared;
        }
```
The return statement may return the result of an expression directly instead of via a variable.
E.g.:
```
int sqr(int value_in)
        {
          return value_in*value_in;
        }
```

The main Function:
All C++ programs start executing at a special function called main.
It frequently looks like,
```
void main()
{
  …
}
```

Example:
```
void main()
{
   int x;
   int y;
   int z;
   x=4;
   y=2;
   z=x+y;
}
```
You could even compile, link and run this program, but we can't see any output.

Output:
Unfortunately C++ has no I/O syntax.
C++ has its own set of I/O library syntax defined in a header file called iostream.h.
We must include this header file in our program before main, as follows.
#include <iostream.h>
To output data onto the screen, we use the word cout followed by the insertion / put to operator (<<).
E.g.:
```
cout<<x;
cout<<“This is a string literal”;
cout<<“Value of x is: ”<<x<<“  Value of y is: ”<<y;
cout<<“Value of x is: ”<<x<<endl<<“  Value of y is: ”<<y<<endl;   //endl forces a line feed
```

Example:
```
#include <iostream.h>
void main()
{
   int x;
   int y;
   int z;
   x=4;
   y=2;
   z=x+y;
   cout<<"Value of z is "<<z<<endl;
}
```

Output:

```
Value of z is 6
```

Input:

To input data from the keyboard, we use the word cin followed by the extraction / get from operator (>>).

```
E.g.:  cin>>x;
       cin>>x>>y;
       cout<<"Enter a number: "; cin>>x;
```

Example:

```
#include <iostream.h>
void main()
{
    int x,y,z;
    cout<<"Please enter an integer: ";
    cin>>x;
    cout<<"Please enter the second integer: ";
    cin>>y;
    z=x+y;
    cout<<"Total is "<<z<<endl;
}
```

Output:

```
Please enter an integer: 4
Please enter the second integer: -6
Total is -2
```

Comments:

There two ways of writing comments in C++.

```
// This is a single line comment
```

and

```
/* This is a
multi line comment */
```