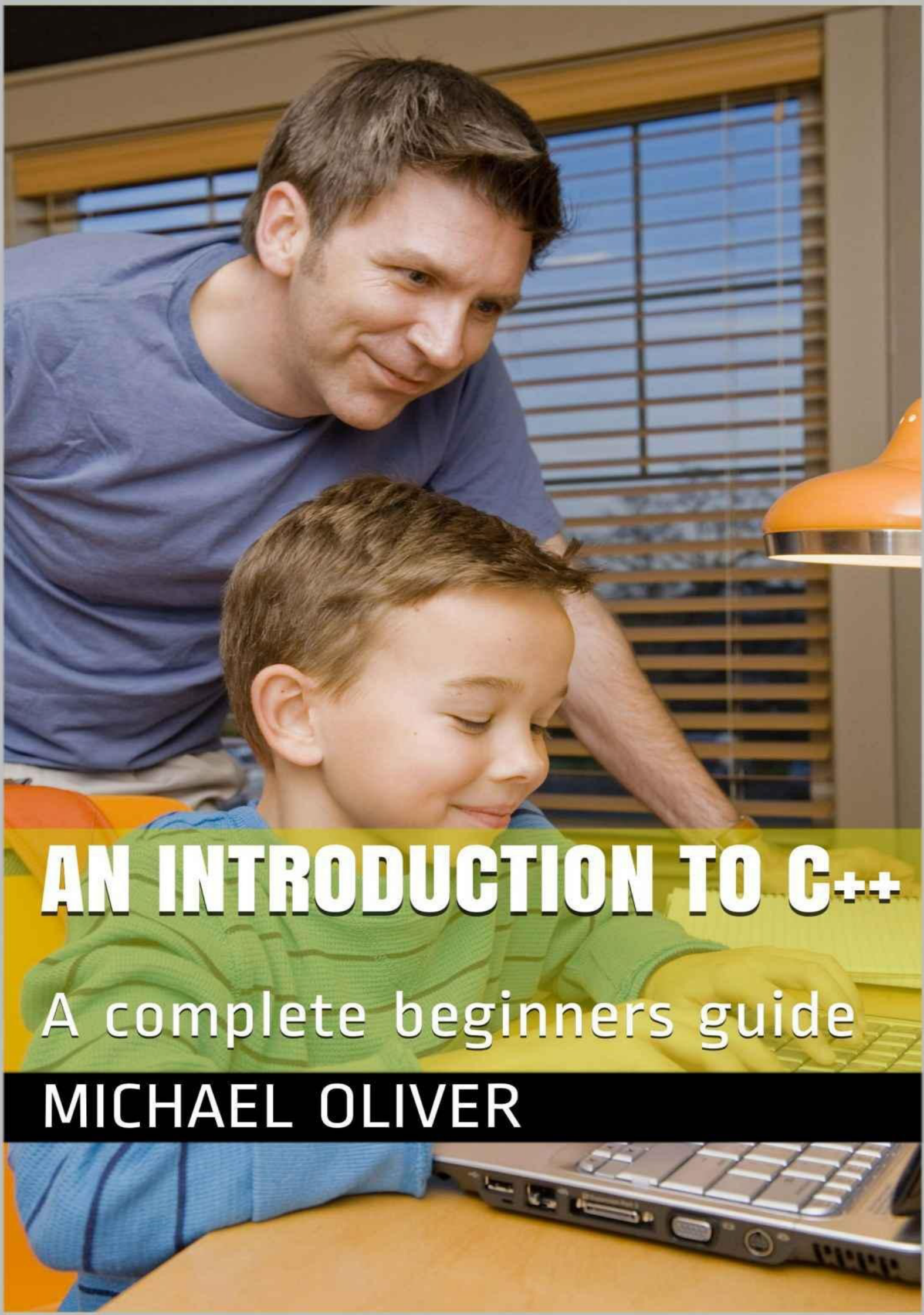


AN INTRODUCTION TO C++

A complete beginners guide

MICHAEL OLIVER

Introduction to Programming with C++

C++ Programming Essentials

Key Words				
break	case	char	default	do
double	else	float	for	goto
if	int	long	operator	signed
static	switch	unsigned	void	while

Data Types	
char	A single character
int	An integer is a whole number
float	A floating point number, correct to six decimal places
double	A floating point number, correct to ten decimal places
bool	A Boolean value of true (1) or false (0)
string	A group of characters

Library Classes	
<iostream>	Reads and writes to the console
<iomanip>	Manipulates input and output
<fstream>	Reads and writes to a text file
<string>	Stores and manipulates text streams
<sstream>	Converts strings to other data types
<vector>	Stores and manipulates data in vector array elements
<stdexcept>	Reports logic errors and runtime errors
<ctime>	Reports date and time

Introduction to C++

C++ was developed by Bjarne Stroustrup of AT&T Bell Laboratories in the early 1980's, and is based on the C language. The name is a pun - "++" is a syntactic construct used in C (to increment a variable), and C++ is intended as an incremental improvement of C. Most of C is a subset of C++, so that most C programs can be compiled (i.e. converted into a series of low-level instructions that the computer can execute directly) using a C++ compiler.

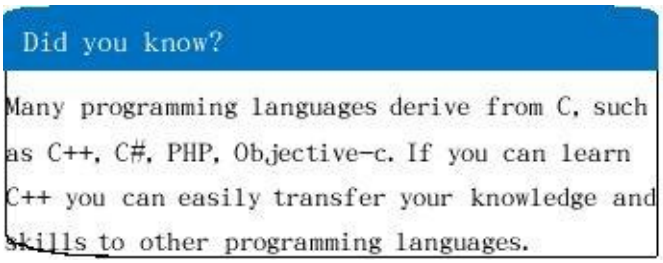
C++ is considered to be a multi paradigm language, as the way it can be implemented can vary. This book will cover C++ as a procedural language, and conforms to the ANSI (American national standard institute) and ISO (International standards organisations). This means that you can take all the source code from this book, and run it in any compiler.

To be able to successfully run the source code in this book you will need the following:

- A computer (Windows, OSx, Linux)
- An IDE (Integrated development environment)
  - Windows (Visual Studio)
  - OSx (X code)
  - Linux (Code Blocks)

For a free version of this software, then please visit www.bcotcomputing.co.uk

All source code in this guide has been implemented using Visual Studio 2013 professional and can be found on the school's website. This software has been installed on all of the machines in the school of computer of science. It can also be downloaded for free using your dreams spark account by clicking this link

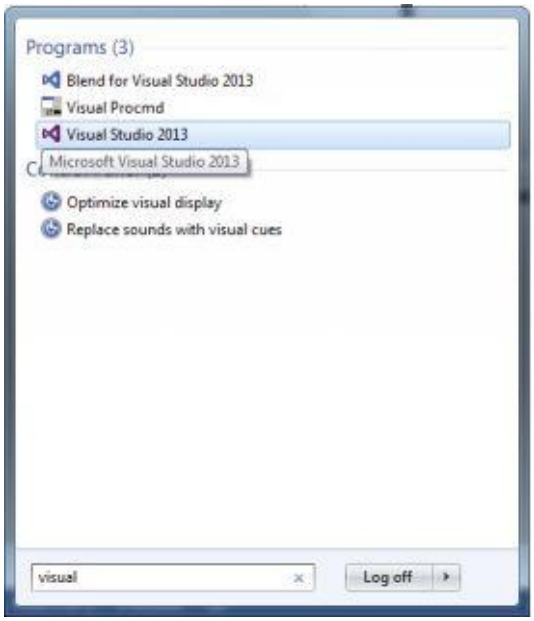



Visual Studio
Professional 2013

Creating your first C++ project in Visual studio 2013

This guide has been created to show how to setup, create and compile your first C++ program “Hello world”.

Before you begin to code your program, you will need to load up an IDE (Integrated development environment) To do this click the start icon , and type visual. You should see the following:

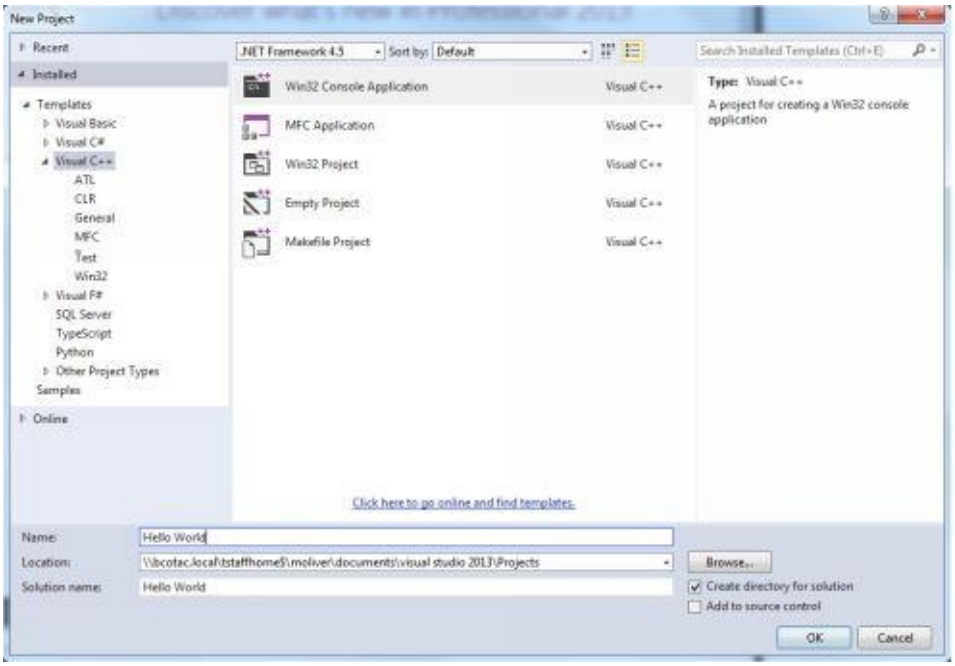


Click “Visual Studio 2013” and the Visual Studio loading screen will appear on your desktop:

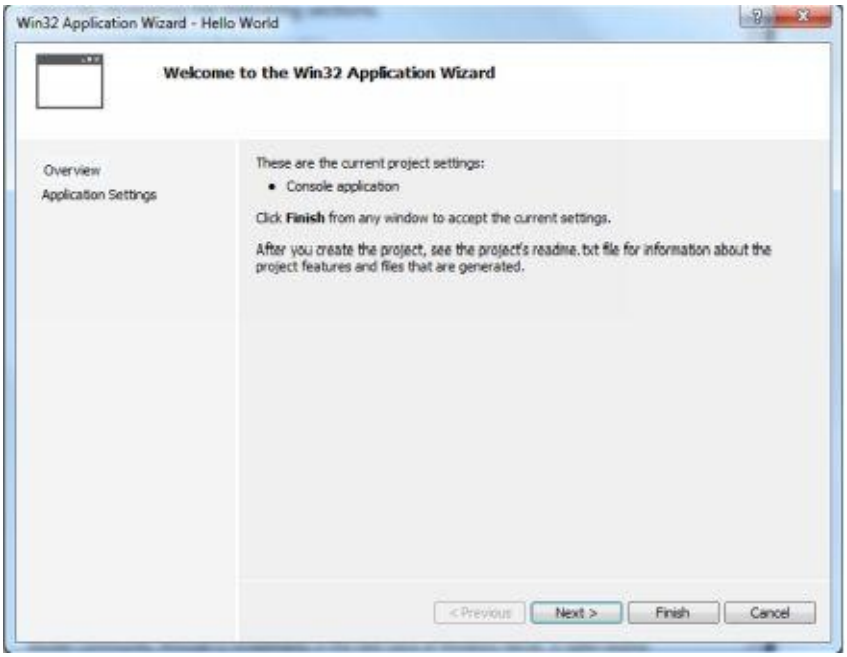
After the loading screen, the Visual Studio start page should load. In this screen you can create new projects, open existing projects or even access guidance and resources. To create a new project we need to click “new project”.

Once you click on new project, you will be presented with a new window (See below) in the left hand pane you will a series of programming languages. By default visual C++ will

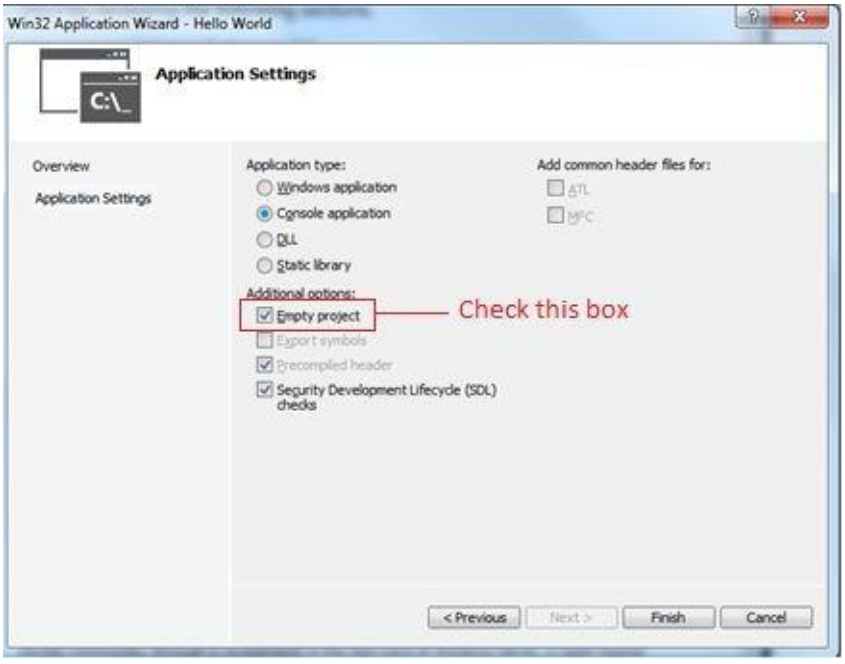
be displayed as the main language. Under visual C++ you will see an option “win32”, click this option to be presented with “win32 console application”. Click this and give your project the name of “hello world”



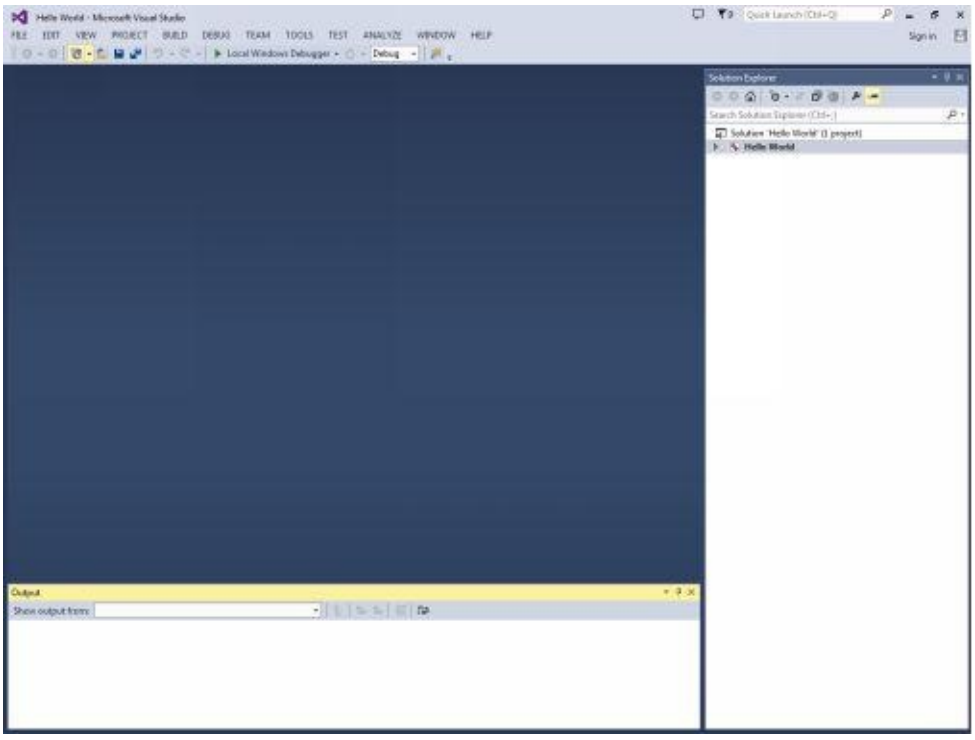
Once you have named the project, click next and you will be presented with a win32 Application wizard. To proceed click next.



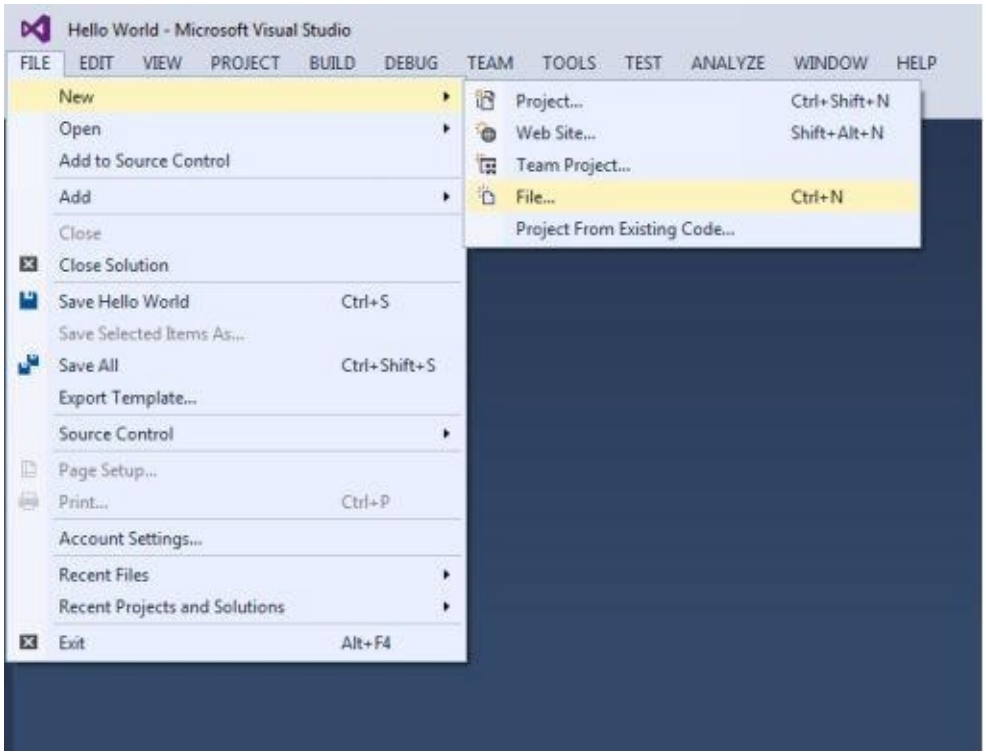
On this screen you need to ensure the radio button called “console application” is checked and the checkbox “empty project” is also checked. Once this has been completed, click finish. If you fail to do this part correctly you will have to start your project again.



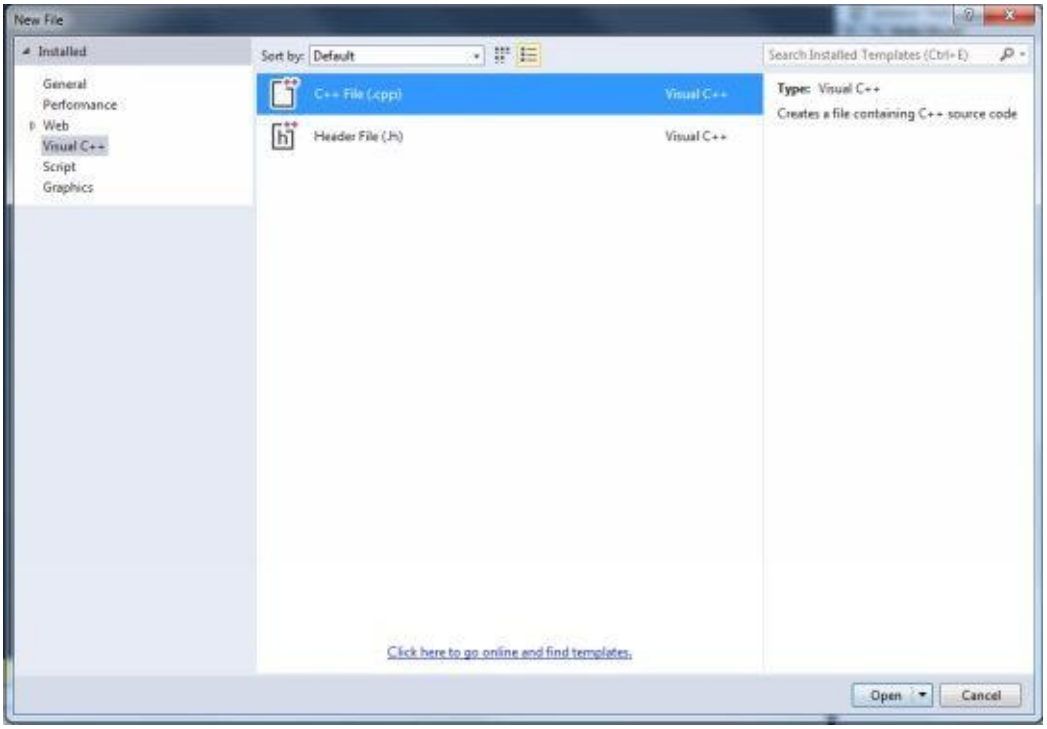
Your wizard will have closed, and you will have been brought to the main screen which shows you, your project files. At the moment, we haven’t got any active C++ files. So you will need to create one before we can begin coding.

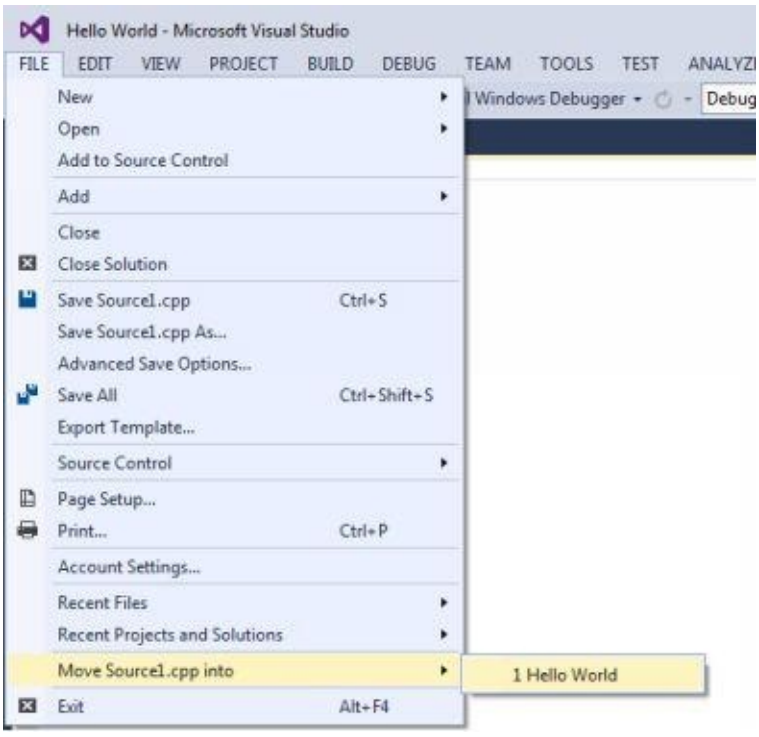


Click file > new > file, to add a new file to your C++ project, or press ctrl + N.



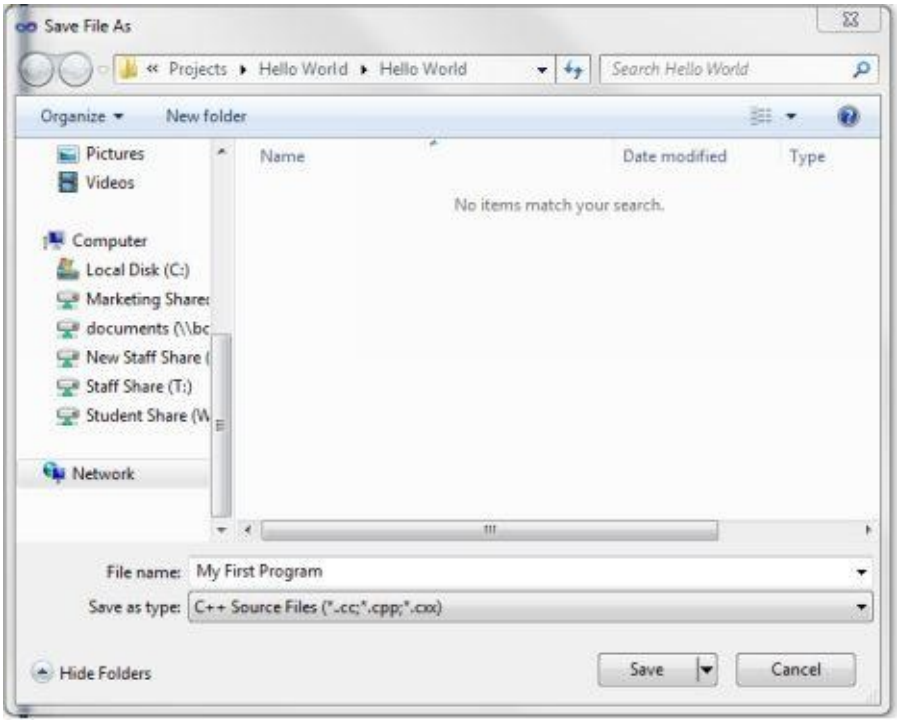
You will come to a new window that will require you to choose the file type needed as part of your project. Choose visual C++ in the left hand pane (Default) and then select “C++ file”. Then click open, which will create a new file and open in your project.





Your nearly there, your file has now been created, but we need to save it to the right location otherwise any code placed inside your file will not work.

To move the file to the correct location, click file, and at the bottom of the menu will be an option called “Move source1.cpp into”, move your mouse over this option and a new menu, will appear called “1 Hello World”. Select this option to move your C++ file into the correct location.



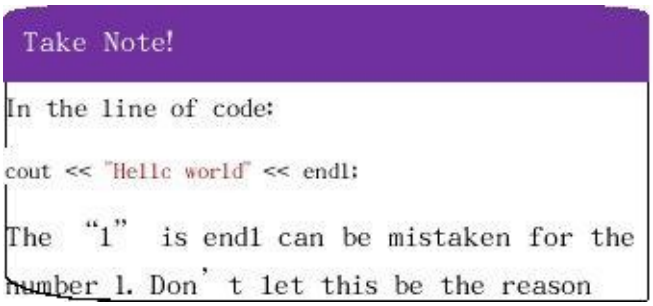
You will be prompted to name your C++ file before you save it. Give it a name that's relevant to your program, but do not give it the same name as your project as you could run into an errors when compiling it.

Well done, you have now successfully setup your C++ file, and it's now ready to code. You will have noticed that the file structure in the left hand side pane, will now have your newly named C++ file in it.

Now you are ready to code and compile your first C++ Program

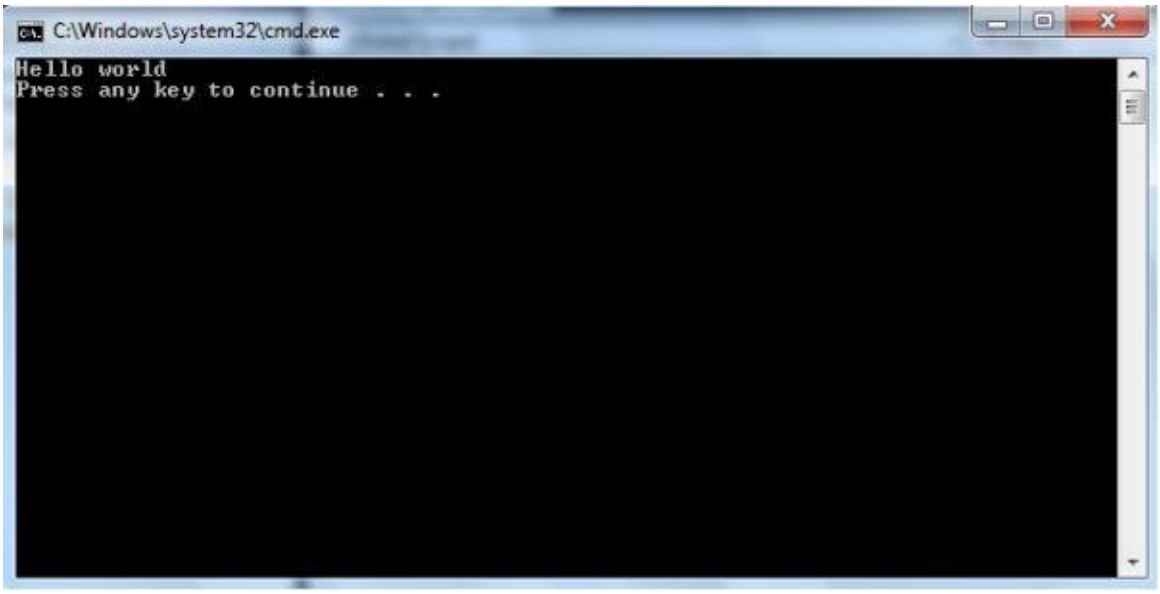
Now let's code your first program in C++. Using the C++ source code below, copy the code into your own Hello World Project.

```
//My first C++ program
#include <iostream>
using namespace std;
int main()
{
        cout << "Hello world" << endl;
        return 0;
 }
```



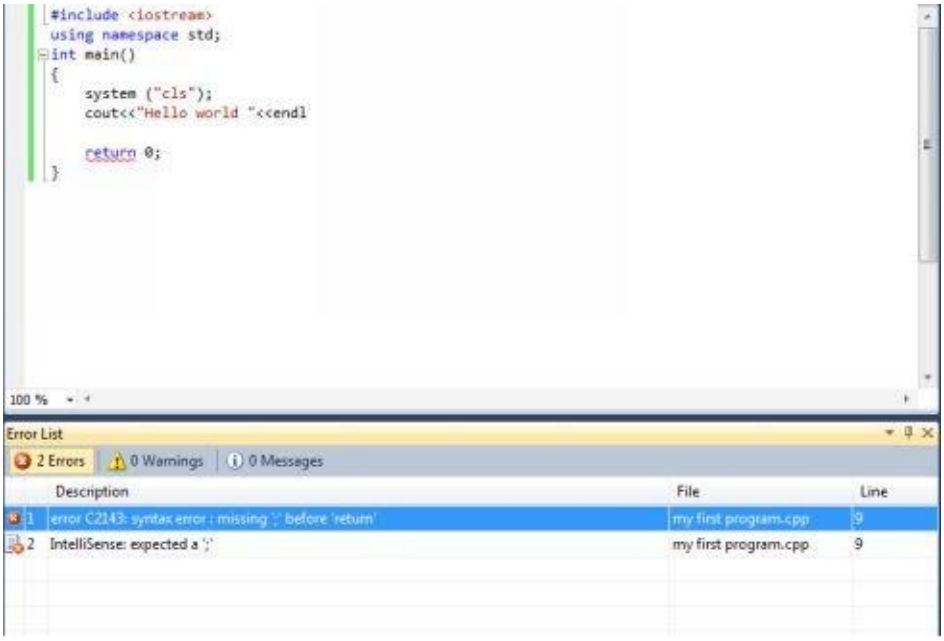
Once you have entered this code we are ready to compile the program. This can be done by pressing ctrl =F5 and clicking ok in the dialog box that appears.

Once this is done then the console screen will display with your first hello world program.



Well done you have successfully completed your first C++ program.

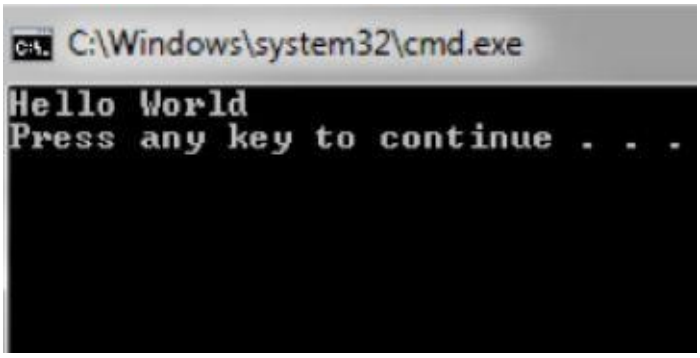
Got an error or the program didn't run. Check the error box and review the error messages. The error box can be found in by using the toolbar at the top, view>



All errors will appear in the error list, which will automatically appear and give you an error message. In this case the error is on line 9, which is missing the ; after the <<endl code.

Understanding your 1st C++ Program

Now that you have successfully created your first C++ program called“Hello World”, let’s look at the structure of your program, and break down the code bit by bit. Letslook at the“Hello World” program below.




```
//My first C++ program
#include <iostream>
using namespace std;
int main()
{
cout<<“Hello World” <<endl;
return 0;
}
```

//My first C++ program

This is a single comment line which has not affect on the performance of a program. We use comments to annotate our code, to explain what the program does. There are two ways to display comments (//) is used for a single line. To write a description of a particular program or source code, you may wish to go over several lines of code. To do this we use (/* */)

#include <iostream>

In programming we use a directive that tells the compiler (Visual studio) how it should process its input. Directives can vary from compiler to compiler and behave differently and are processed by a pre-processor. A pre-processor is a program that processes its input data, to produce an output. Different languages use different pre-processor, but in C languages, the most common is pre-processor, takes lines starting with # as directives. The above #include <iostream> tells the pre-processor of the compiler to include the iostream standard file. This file includes the fundamental delectations of the input-output library in C++. Without this your C++ would not compile.

using namespace std;

All the elements of the standard C++ library are declared within what is called a namespace, the namespace with the name std. So in order to access its functionality we

declare with this expression that we will be using these entities. This line is very frequent in C++ programs that use the standard library.

int main()

This line corresponds to the beginning of the definition of the main function. The main function is the point by where all C++ programs start their execution, independently of its location within the source code. It does not matter whether there are other functions with other names defined before or after it - the instructions contained within this function's definition will always be the first ones to be executed in any C++ program. For that same reason, it is essential that all C++ programs have a main function.

The word main is followed in the code by a pair of parentheses (()). That is because it is a function declaration: In C++, what differentiates a function declaration from other types of expressions are these parentheses that follow its name. Optionally, these parentheses may enclose a list of parameters within them.

Right after these parentheses we can find the body of the main function enclosed in braces ({}). What is contained within these braces is what the function does when it is executed.

cout<<"Hello World" <<endl;
This line is a C++ statement. A statement is a simple or compound expression that can actually produce some effect. In fact, this statement performs the only action that generates a visible effect in our first program. cout is the name of the standard output stream in C++, and the meaning of the entire statement is to insert a sequence of characters (in this case the Hello World sequence of characters) into the standard output stream (cout, which usually corresponds to the screen). cout is declared in the iostream standard file within the std namespace, so that's why we needed to include that specific file and to declare that we were going to use this specific namespace earlier in our code.

Notice that the statement ends with a semicolon character (;). This character is used to mark the end of the statement and in fact it must be included at the end of all expression statements in all C++ programs (one of the most common syntax errors is indeed to forget to include some semicolon after a statement). <<endl; is used after the output statement to move any other statements onto its own separate line of the output window. As seem above in the output window "Press any key to continue" is on its own separate line. If the <<endl; was removed then the output statement "hello world" would be on the same line as "press any key to continue".

return 0;

The return statement causes the main function to finish. Return may be followed by a return code (in our example is followed by the return code with a value of zero). A return code of 0 for the main function is generally interpreted as the program worked as expected without any errors during its execution. This is the most usual way to end a C++ console

program.

Solving Errors in C++

Take Note!

C++ is case sensitive, all C++ code is written in lower case, unless you create any functions/ variables that use capital letters.

When you begin coding in C++, you will notice that you may make several errors in your code, such as missing the semi-colon (;) or using uppercase letters when in fact you should be using lower case. The best way to become familiar with any language is to practice rectifying errors in the code.

Activity 2: Solving errors

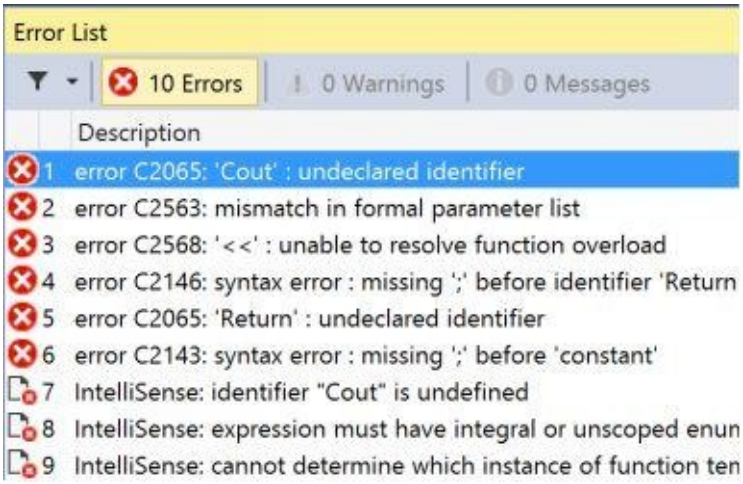
Below is a C++ program, which has 7 errors. Create a new project and copy the code below into your new project and resolve the issues. *HINT: use the code from your first program to help*. Once you have finished, save and compile your program.

```
#include <Iostream>
using namespace std;
int Main()
{
        Cout <“Please Fix my Errors “ << endl
                Return 0;
};
```

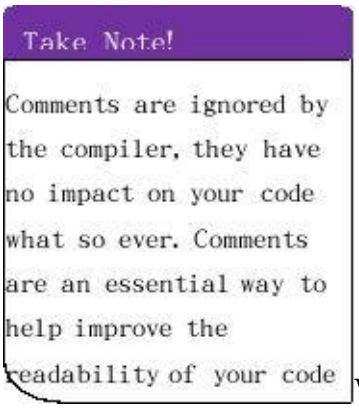
To view your errors in C++, you can use the built in error tool, which tells you where the errors have occurred within your program. To access this tool, click view on the top toolbar and select error list. Double click on each error in this list to be taken to the line of code where the error has occurred.

Hot Tip!

When handling errors in C++, start with the error closest to the top of your code as one error could cause lots of other errors. Start from the top of your code and work your way down. Don’ t forget to use the built in error tool in



Annotating Your Code



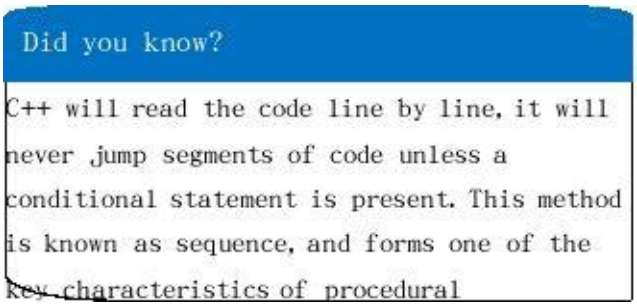
When you code in any programming language, it's extremely important that you use comments to explain what is happening. This is so that other programmers, can understand your code and what your program is trying to achieve.

Activity 3: Annotating your code

Below is a C++ program that shows you how to use comments. Create a new project called "comments" and copy the below program, save and compile the code.

```
#include <iostream>
using namespace std;
int main()
{
// This is how you add a comment on to a single line

/*This is how to add a comment over multiple lines. The compiler will not read this. Comments are used to annotate your code*/

        cout << "Hello World" ;

return 0;
 }
```

It will be expected that as you progress through the work book, that you will be required to annotate all of your programs.



You have seen three C++ programs, now it’s time to try it yourself. Create a C++ program that prints out your name on to the screen. You can use the previous programs as a guide, but try doing without looking.

Don’t forget to annotate your code using your comments!!

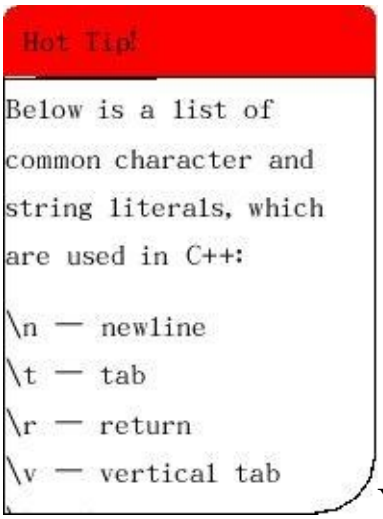
Adding Multiple Line of Code

Now that you have coded your first simple C++ program, let's look at making it a bit more complex, by adding more lines of code.

Activity 4: Adding Multiple lines of code

Create a new C++ project called"multiple lines", and using the code below, copy the code into your program, then save and compile your program.

```
#include <iostream>
using namespace std;
int main()
{

  cout<<"This is line 3" <<endl; //endl is used to place the code on a new line
  cout<<"This is line 5\n"; // \n is another way of putting the code onto a new line
   return 0;
 }
```



You will have noticed that you have used (/n) and (<<endl) to print the code on to a new line. As you continue to learn C++ you will choose a method that is suited to your programming style. There is no wrong or right way, just the way that suits you.

This book will use <<endl throughout, but you can change this to suit your needs.

Now that you know how to print code on separate line, create a C++ program that tells the story of the three little pigs and the big bad wolf.

Your code, will need to compile and you will need to annotate the key parts to your code.

Introduction to Variables

Variables are temporary storage that exist as long as they need to, then are disposed of. Imagine a calculator, the numbers you input are stored in the calculator until you use the clear button or turn the calculator off, then the values are gone. Variables are stored in the main memory of a computer system, and given a unique address so that they can be accessed by the compiler at any time.

Variables are declared by the type of data to be expected, such as number, a letter or a sentence. Declaring the data type is essential to ensure that the correct machine code is generated by the compiler to be used by the CPU (Central processing Unit).

What Data types can a variable hold?

Integers (int)

An int is short for an integer which is a whole number such as 0,1,2,3,4,5,6,7,8,9 etc. They can be both signed (negative) and unsigned (positive). Integers can range in size depending on the type of integer that is being used.

Characters (char)

A char which is short of character is a data type that can hold single letter, symbol or number such as ‘A’, ‘b’, ‘@‘, ‘$’, ‘1’ etc. A simple char will use at least 1 byte (8 bits) of storage space.

Float (Decimal)

A float, which is short for floating point number is a way of storing a number, which has a decimal place such as 3.2, 6.8, 7.1 etc. Depending on the type of float used will depend on their precision. There are two types of floating point numbers which are Floats and doubles. Doubles have a greater precision and generally the preferable data type to be used within a C++ program.

Strings

Strings are very powerful and used to not only store words, but are used to store a combination of words and numbers. Strings are useful when you want to capture information such as the full name of the person (First name and surname) street address etc.

Boolean

Boolean in C++ is known as bool, which is represented in either two states; true or false. You may see that this data type is used with binary numbers, this again reflects the two states of Boolean. 0= false and 1 = true.

Declaring Variables in C++

To declare a variable in C++, we need to declare the type of data that we expect the variable to hold, this can be done by typing the data type name in, as shown below in the table:

Data type's: How to declare			
Integer	int	Float	double
Character	char	string	string
Float	float	Boolean	bool

When you type the data type in C++, you will notice that the text will change colour, this is to show that it will be used by the compiler for a variable assignment.

Once you have chosen the data type for your variable, you will need to assign it a name. You can name the variable anything you like, as long as you follow the rules:

Variable Name Rules

When variables are created, you give them a name such as num1, housename, fname, etc. Variables are very flexible with names, however there are some golden rules to creating them:

- A Variable name consists of any combination of alphabets, digits and underscores. Some compiler allows variable names whole length could be up to 247 characters. Still it would be safer to stick to the rule of 31 characters. Please avoid creating long variable name as it adds to your typing effort.
- All variable names must begin with a letter of the alphabet or an underscore(_). For beginning programmers, it may be easier to begin all variable names with a letter of the alphabet. The first character must not be a number or any special symbol
- After the first initial letter, variable names can also contain letters and numbers. No spaces or special characters, however, are allowed.
- Uppercase characters are distinct from lowercase characters. Using all uppercase letters is used primarily to identify constant variables.
- You cannot use a C++ keyword (reserved word) as a variable name.

Examples of variables being assigned

```
int num1; //num1 is the variable name I have chosen which expects an intger data type.
float num2; // num2 is the variable name chosen which expects a float data type.
```

```
char letter; // letter is the variable name chosen which expects a char data type
string name; // name is the variable name chosen which expects a string or more than one.
```

Using Integers and Characters in C++

In this part of the program we are going to create a C++ program that is used to store a simple whole number (Integer) in the program. We will declare a variable that uses an integer data type, and initialise it with a preset value.

Activity 5: Using Integers

Create a new C++ program below called “my first variable” and enter the following code:

```
#include <iostream>

using namespace std;
int main()
{
        int num; // variable declared to store a whole number
        int num = 10; //variable initialised with value 10

        cout << "My Number is :" << num; //prints out my number
```

To save lines of code, you can declare and initialise a variable on the same line of code as shown below:

```
int num =10; // variable that has been declared and initialised
```

Activity 5b: Using Characters

Now that you have created a program that uses a variable integer how about creating a variable that uses a char (character). Below is another C++ program that shows you how to declare a char in C++. Create a new C++ program called “My second variable”, and enter the below code:

```
#include <iostream>
using namespace std;
int main()
{
        char letter = ‘A’; //char variable declared and initalised


        cout << “My letter is :” << letter; //prints out my letter


        return 0;
 }
```

Did you notice, that we declared and initialised the letter ‘A’ on the same line? Again this reduces the amount of code your program has. Did you also notice that A, was initialised in single quotation marks (‘A’). This tells the compiler to use the ASCII value of A, If you try to declare A without the single quotation marks it will present you and error as its seen as an “Undeclared identifier”.

The ASCII table will play a huge part in your C++ and can be found at the back of this book.

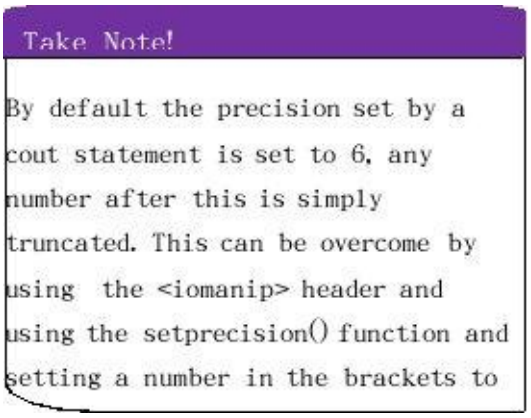
So far, you have been able to declare and initialise an integer and a char in C++. Using these two data types only create a C++ program that stores a variety of chars and integers in several different variables.

This program is entirely up to you how you structure it, but it must compile and all the code must be annotated.

Using floats in C++

We are now going to look at Floats in C++ and see how they can be implemented. Floats are used to show numbers after the decimal place. Depending on the type of float used, will depend on the precision (how many numbers are after the decimal point). For example using the float in C++ can show up to 7 places after the decimal point, where a double can show up to 16. The two programs demonstrate this.

Activity 6



Create a new C++ program called “My Float” and copy the code below, compile and run.

```
#include <iostream>

using namespace std;
int main()
{

        float number1 = 3.123456789;
        double number2 = 3.123456789;

        cout << number1 << endl;
        cout << number2 << endl;

        return 0;
 }
```

Further reading and on floating point numbers can be found here

Activity 7 – Using Multiple variables

Hot Tip!

Did you notice that the line of line char letter; letter ='A'; " has been placed on the same line. This is a poor way to declare and initialise a variable on the same line. Instead you should do it like:

Create a new C++ called "Multiple Variables" and copy the below code:

```
#include <iostream>
using namespace std;
int main()
{
	char letter; letter ='A'; //declared variable and then initialised
	int number;  number =100; //declared variable and then initialised
	float decimal = 7.5; //declared and initialised
	double pi = 3.14159; //declared and initialised
	bool isTrue = false; //declared and initialised

	cout<<"Char letter: "<<letter<<endl;
	cout<<"int number: "<<number<<endl;
	cout<<"float decimal: "<<decimal<<endl;
	cout<<"double pi: "<<pi<<endl;
	cout<<"Bool isTrue: "<<isTrue<<endl;
	return 0;
 }
```

Using strings

Strings are used when you need to store data that contains more than one character such as a name, email address, or postcode. Strings are different to all other data types in C++, as you have to call a new programming library.

Activity 8 – Using Strings

In this program you are going to create your first C++ program that displays “Hello world” in a string variable. Create a new C++ program called “intro to strings”, enter the code below and compile the program.

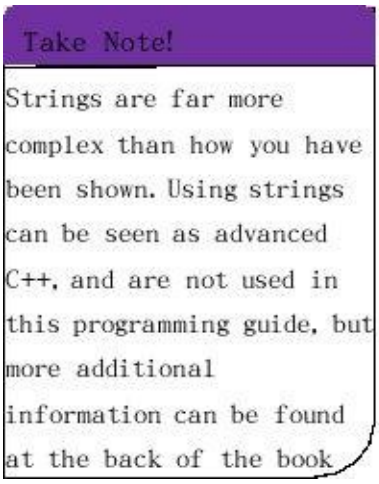
```
#include <iostream>
#include <string>//the new string header has to be used or string will not work.
using namespace std;
int main()
{
        string hello =“Hello World”;
        cout << “Please enter your first name: “ << endl;
        cout << hello << ” Welcome to C++”<<endl;

        return 0;
 }
```

If you wanted to store a combination of words and letters into the same variable, rather than using a string, you could use a Char Array. Arrays are covered towards the end of this guide and is the start of intermediate C++.

Another program below has been included to show how two separate strings can be concatenated together to form one string.

```
#include <iostream>
#include <string>

using namespace std;
int main()
{
```

```
        string firstname = "Frodo ";
        string secondname = "Baggins";
        string fullname;

        fullname = firstname + secondname; //example of concatenation
        cout << "First name: " <<firstname << endl;
        cout << "Second name: " <<secondname <<endl;
        cout << "Your full name is: " << fullname <<endl;

        return 0;
}
```



Getting User Input

There will be times where you will need to get the user to input some form of data in to your program, that you can use either use for conditional statements and loops or if your write certain instructions out onto the screen.

To gather input we need to use the “cin>>” command followed by the name of the variable that will be holding the data. For example cin>>name;

Activity 7 – User input (Integer)

In this program you are going to create a C++ program that allows a user to input a number and display that number back out on the screen. Create a new C++ project called “intger input” and enter the below code, compile and run.

```
#include <iostream>
using namespace std;

int main()
{
        int num;

        cout << “Please enter a number”;
        cin>>num;
        cout<< “You entered “ <<num <<endl;

    return 0;
}
```

Activity 8 – User input (String)

In the below program you will create a program that will allow a user to enter a name (without spaces) and display it back out to the user. Enter the code below, save, compile and run.

```
#include <iostream>
#include <string>//the new string header has to be used or string will not work.
using namespace std;
int main()
{
        string name;
```

```
        cout << “Please enter your first name: “ << endl;
        cin >> name;
        cout << “Hello “ << name << ” Welcome to C++”<<endl;


    return 0;
}
```

Now that you know how to declare variables, allow user data input and display the results back out to the user it's time to create your own C++ program that captures user information. Your program will need to capture the following data:

- First name,
- Surname,
- Age,
- House number,
- 1st line of address,
- City
- State
- Postcode

You can use previous exercises to help you design this program. In this program you will expected to do the following:

- Use comments to explain your code
- Indent your code correctly
- Use the correct variable type and appropriate variable names

As an extension to this exercise concatenate the two string variables used to hold the first and second name into one string, so that it can be printed out on one line.

Variable Arithmetic

Variable Arithmetic: What is arithmetic? Arithmetic is math, calculating the sum of a range of numbers. For example you can add two variables together and store the answer in a new variable.

Activity 9 - Addition

In this activity you will be creating a C++ program that asks the user to enter two numbers and the program will add the numbers together and produce the total amount, which will be displayed on the screen. Create a project called “Variable Maths” save, compile and run the below code.

```
#include <iostream>
using namespace std;
int main()
{
        int num1, num2, sum;

//addition program

        cout<< “Please enter your first number to add: “;
        cin>>num1;
        cout<< “Please enter your second number to add: “;
        cin>>num2;
    sum = num1+num2; //add variable num1 and num2, and store new number in      variable sum

    cout<<num1<< “+” <<num2<< “=” <<sum;

return 0;
 }
```

Activity 10 - Multiplication

The below program, is a multiplication program that mutliplys two numbers together, and displays the result. Create a new project called “Multiplication” and code, save, compile the below code.

```
#include <iostream>
using namespace std;
```

```
int main()
{
        int num1, num2, sum;
        cout << “Please enter your first number to add: “;
        cin >> num1;
        cout << “Please enter your second number to add: “;
        cin >> num2;

        sum = num1*num2; //times variable num 1 and 2, and store new number in variable sum
        cout << num1 << “x” << num2 << “=” << sum;

        return 0;
 }
```

Activity 11– Peas in a Pod

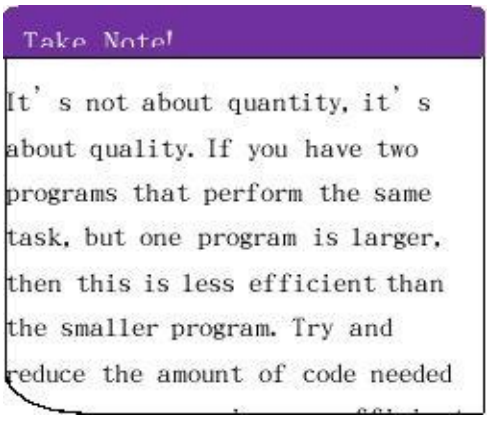
Create a new project called “Peas and Pods” and enter the below code.

Enter the following program, **making sure that intentional errors introduced are corrected**. The case of all letters must be copied as shown.

Save, compile and run the program.

```
#include <iostream>
using namespace std;
int main()
{
        int number_of_pods;
    int peas_per_pod;
    int total_peas;

        cout << “Enter the number of pods:\n”;
        cin >> number_of_pods;
        cout << “Enter the number of peas in a pod:\n”;
        cin >> peas_per_pod;

        total_peas = number_of_pods * peas_per_pod;

        cout << “If you have “;
```

```
        cout << number_of_pods;
        cout << " pea pods\n";
        cout << "and ";
        cout << peas_per_pod;
        cout << " peas in each pod, then\n";
        cout << "you have ";
        cout << total_peas;
        cout << " peas in all the pods.\n";

    return 0;
}
```



Now that you have successfully compiled the above program, you will need to try and shorten the program, by reducing the lines of code. Based on previous examples shown in the guide so far, shorten the code above onto a fewer lines as possible.

Global and local variables

We know how to use variables, but so far you have bene using local variables. Local variables are variables that are declared within side a function, and are made only available to that function. The other type of variable is a Global Variable, which is declared outside a function and is made available to all functions in a C++ program.

Global variables are useful, but the value they hold can easily be manipulated by the different functions that use them. Many developers avoid using global variables and instead pass data from one function to another to avoid the data being lost.

Activity 12 – Global and Local Variables

In this program, we are going to declare one global variable, and two local variables. One of these variables will be a standard local variable and the other will be declared as a constant. A constant is a variable that must have a value initialised before the program runs, and its value will remain the same throughout the program. Constants can't have their value changed at run time.

```
#include <iostream>
using namespace std;

int num1; //I am a global variable, I can be used by any function within C++

int main()
{
        const int num2=5; // I am a constant local variable. I can be used within   this function only and my value will never change.
        int num3; //I am a local variable and can only be used within the "main" function

        cout<<"Please enter a number for global variable num1: ";
        cin>>num1;
        cout<<"We are going to add "<<num1 <<"to our constant variable, which has the value of 5"<<endl;
        cout<<num1 << " + " <<num2 << " = " <<(num3=num1+num2) <<endl;

        return 0;
 }
```

Intro conditional statements

Conditional Statements: A condition statement in C++ is when we check to see if a certain condition has been met. Condition statements are great, when you want to display different results depending on what the user selects. In programming a condition is also known as if/else statements. In this exercise you will cover if/else statements.

They are also a great way to change the direction depending on what the user selects. In programming a condition is also known as if/else statements. In this exercise you will cover if/else statements. Conditional statements can have multiple conditions, which are tested in order.

An example of an conditional statement is:

If Joshua eats all his dinner, then he can go out and play. If not then he must go tidy his room. Depending on the result of Joshua eating his dinner with depend on the outcome.

IF Joshua = Eat All dinner **Then go out and play**	**This is condition 1**
Else IF Joshua = doesn't eat his dinner **Then Tidy your room**	**This is condition 2**

In this section of C++ you will be introduced to new symbols that you will need to familiarise yourself with. The table below explains these symbols.

==	Equal too	&&	and
!=	Not Equal too	||	Or
>	Greater Than	>=	More than and equal too
<	Less than	<=	Less than and equal too

Take Note!

Be careful! The operator = (one equal sign) is not the same as the operator == (two equal signs), the first one is an assignment operator (assigns the a value to a variable and the other one (==) is the equality operator that compares whether both expressions in the two sides of it are equal to each other.

Activity 13:

In this activity we are creating a simple if statement where a user is prompted to enter a number between under 10. This program has two outcomes/condition, depending on the number that the user enters will depend on what condition becomes true.

Create a new C++ project and call it “condition statements”. Enter the below code, save and compile the program.

```
#include <iostream>
using namespace std;
int main ()
{
   int num;

   cout<<“Please enter a number under 10 : “;
   cin>>num;

   if (num <=10) //outcome 1
      cout<<“Your number was 10 or less”;
   else //outcome 2
      cout<<“Your number was greater than 10”;

   return 0;
 }
```

Well done you have just created your first C++ program that uses a condition statements.

In the next exercise you will learn how to code a C++ with more than two outcomes, which can develop a solution if the user enters a number of 10.

Create a C++ program that uses a conditional statement for a teacher who wants to know what action should be taken for a student who has not completed their homework. The program will need to be called “Teacher Program” and have two conditions:

- The 1st condition being if the student does the homework
- The 2nd condition being if the student hasn’t done their homework.

Note: This program will need to use a Char data type to check the condition not an integer

Activity 14

In this activity you are going to create a C++ program that tests for multiple conditions, using if and else if statement's. Create a new C++ Project and call it "else if statement" and Enter the following code below, save, compile and run.

```
#include <iostream>
using namespace std;
int main()
{
	int num;

	cout << "Please enter a number between 1 and 5 : ";
	cin >> num;

	if (num == 1)
	{
		cout << "Your entered the number 1"<<endl;
	}

	else if (num == 2)
	{
		cout << "Your entered the number 2" << endl;
	}

	else if (num == 3)
	{
		cout << "Your entered the number 3" << endl;
	}
	else if (num == 4)
	{
		cout << "Your entered the number 4" << endl;
	}
	else if (num == 5)
	{
		cout << "Your entered the number 5" << endl;
	}
```

```
        return 0;
}
```

Using the above code as a guide create a program that tells the user that they pressed a specific set of letters ‘Q’, ‘W’, ‘E’, ‘R’, ‘T’, ‘Y’.

Remember that when using letters in C++ that have an impact on the state of the program that they need to be in single quotation marks, as seen above.

Logical Operators and Conditional statements

The logical operators && and || are used when evaluating two expressions to obtain a single relational result. The operator && corresponds with Boolean logical operation AND. This operation results true if both its two operands are true and false otherwise. The following panel shows the result of operator && evaluating the expression a && b:

&& (AND) OPERATOR

a	b	a && b	
true	true	true	The && operator is used if you want to check if multiple conditions have been met. For example. If Joshua eats his dinner (A) and does his homework(B) he can go out and play. Unless both A and B are true, then the statement is false
true	false	False	
false	true	False	
false	false	False	

The operator || corresponds with Boolean logical operation OR. This operation results true if either one of its two operands is true, thus being false only when both operands are false themselves. Here are the possible results of a || b:

|| (OR) OPERATOR

a	b	a || b	
true	true	true	The || operator is used to check if any condition has been met, despite the other statements. For example. If Joshua eats his dinner (A) or does his homework (B) then he can go out and play. If A or B is true then, then the statement is true.
true	false	True	
false	true	True	
false	false	False	

The Operator ! is the C++ operator to perform the Boolean operation NOT, it has only one operand, located at its right, and the only thing that it does is to inverse the value of it, producing false if its operand is true and true if its operand is false. Basically, it returns the opposite Boolean value of evaluating its operand. For example:

```
!(5 == 5)    // evaluates to false because the expression at its right (5 == 5) is true.
!(6 <= 4)    // evaluates to true because (6 <= 4) would be false.
!true        // evaluates to false
!false       // evaluates to true.
```

Activity 15 – OR Operator

In this activity we will use the || (or) operators so we can test for either of the conditions being true. Create a new C++ project called “OR operator” and enter the below code, save and compile.

```
#include <iostream>
using namespace std;
int main ()
{
   char answer;

   cout<<“Please enter the letter A : “;
   cin>>answer;

   if ((answer == ‘A’) || (answer == ‘a’))
// here we have the == (equal) and the ||(or) in C++
   {
      cout<<“well done you entered the letter A”;
   }
   else
   {
     cout<<“You did not enter the letter A”<<endl;
   }
   return 0;
}
```

In this program, we checked against a letter or a number we have to use single quotation ‘A’ marks to identify what letter or number we are looking for. In the above program we have used the || (or) operand. The reason we have used this is we are checking for ‘A’ and ‘a’ which are two different conditions. C++ is case sensitive so ‘A’ is different to ‘a’. We have used brackets to show the program the different conditions we are checking.

```
          //condition 1   OR    //condition 2
   if ((answer == ‘A’) || (answer == ‘a’))
```

Create a C++ program that checks the condition of a number entered, ask the user to enter a number such as 5 OR 10.

Ensure that your program compiles and runs that all code is annotated.

Activity 16 - AND Operator

In this activity we will use the && operators so you can see how we can test for both conditions being true.

Create a new C++ Project and call it “And Operators” and enter the following code below:

```
#include <iostream>
using namespace std;
int main()
{
	int num1,num2;
	cout<<“Please enter the number 5: “;
	cin>>num1;
	cout<<“Please enter the number 3: “;
	cin>>num2;
	if ((num1 ==5) && (num2==3))// if number 5 and 3 are entered then this statement is True
	{
		cout<<“Well done you entered the number 5 and 3”;
	}
	else
	{
		cout<<“Oops you didnt do as your told”<<endl;
	}
	return 0;
}
```

In this activity we are checking both conditions to be true. The conditions are the user entering number 5 and 3. If the user enters number 5, but enters 2 then the statement is false, because they need to enter both numbers.

```
//condition 1 AND condition 2
```

Take Note!

In activity 15 and 16 you will have noticed that each condition was wrapped in a pair of brackets like the below example:

If ((condition 1) && (condition 2))

Each condition must be contained in its own bracket, and then all the conditions you are checking must be wrapped in in a single bracket. See the below example for checking against 3 conditions in the same statement:

If ((condition 1) && (condition 2) && (condition 3))

if ((num1== 5) && (num2==3))

Activity 17: Using Not Operators

In this activity we use the ! (NOT) operator so you can see how we can test for a condition that is not true.

```
#include <iostream>
using namespace std;
int main()
{
	int num1;

	cout<<"Please enter any number, other than 5: ";
	cin>>num1;

	if (num1 != 5)// if number is NOT 5
	{
		cout<<"Well don your number was not 5"<<endl;
	}
	else // if number is 5
	{
		cout<<"Your number was 5"<<endl;
	}
	return 0;
 }
```

You have created a series of program that uses a variety of conditional statements. Now it's time to take what you have learnt and put it into practice! Design and build the following program:

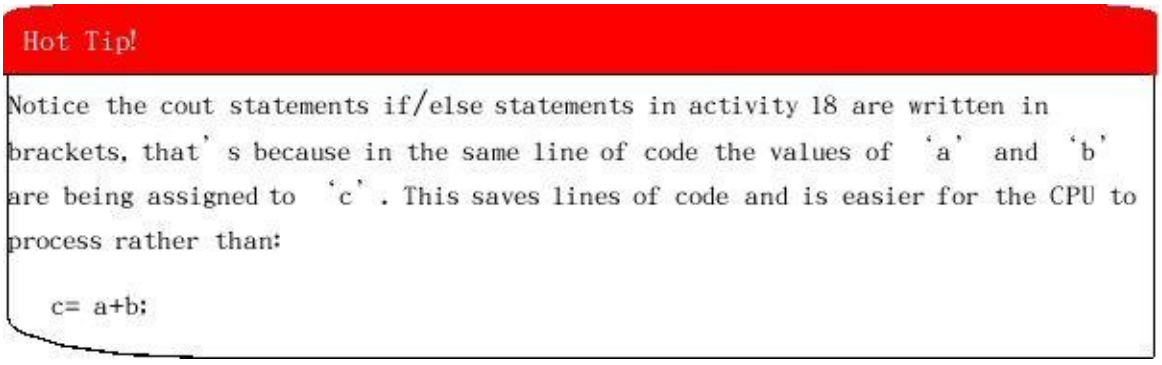
You need to create a grading programing where the lecturer can put in a grade percentage and the program will state what grade the student got. The outputs could be something like this

- Excellent you got 90%, you got Distinction
- Well done you got 75%, you got a Merit
- You got 60%, you got a pass. Try Harder!

You will need to compile and run your program, as well as annotate all your code.

Creating a calculator using conditional statements

Activity 18- Creating a calculator



In this C++ you code a simple calculator to perform any calculation, copy the below code into a new project called "Calculator", save and compile.

```
#include <iostream>
using namespace std;
int main()
{
        float a, b, c;
        char m;
        cout<<"Do you want to add(+), subtract(-), multiply(x), or divide(/)?:" ;
        cin>>m;
        cout<<"Enter a number: " ;
        cin>>a;
        cout<<"Enter a second number: " ;
        cin>>b;
        cout<<"The answer is: ";
        if (m == '+')
        {
                cout<<(c = a + b) <<endl;
        }
        else if (m == '-')
        {
                cout<<(c = a - b) <<endl;
        }
        else if (m == 'x')
        {
                cout<<(c = a * b) <<endl;
```

```
        }
        else if (m == '/')
        {
                cout<<(c = a / b) <<endl;
        }
        else
        {
                cout<<"Invalid Symbol" <<endl;
        }
        return 0;
}
```

Nesting If/Else statements

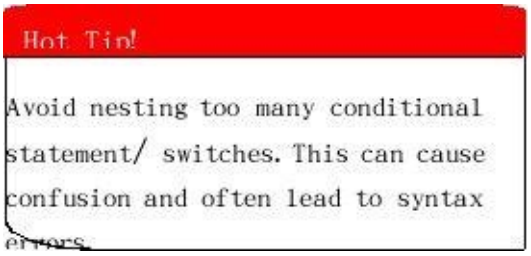
Nesting is a term when we insert one object inside another. For example you might create a conditional statement and insert a loop or another conditional statement. This is very common and something you will do in future exercises to come.

Activity 19 - Nesting Conditional Statements

In this activity we are going to be nesting a conditional statement inside another. Create a new C++ project and call it “nesting conditional statements”. Copy the below code, save and compile.

```
#include <iostream>
using namespace std;
int main ()
{
  int num1, num2;
 cout<<“Please enter a number between 0 - 100 “<<endl;
 cin>>num1;
 if ((num1 >= 0) && (num1<=50))
 {//brackets must be used as there is more than one line of code in this statement
            cout<<“Your number was between 0 - 50”<<endl;
            cout<<“Please enter a new number under 50”<<endl;
            cin>>num2;
                        if((num2 >= 0) && (num2<=50))
                                    cout<<“Well done your number was below 50”<<endl;
                        else
                                   cout<<“Your number was over 50, you didnt listen”<<endl;
 }
 else if ((num1>=51) && (num1<=100))
 {
            cout<<“Your number was between 51 - 100”;
            cout<<“Please enter a new number between 50 - 100”<<endl;
            cin>>num2;
                        if((num2 >= 50) && (num2<=100))
                                    cout<<“Well done your number was between 50 - 100”<<endl;
                        else
                                   cout<<“Your number was under 50, you didnt listen”<<endl;
```

```
 }
 else
            cout<<“Invalid Number”;
          return 0;
}
```



If/else statements are not the only way to look for a certain condition. The other is option is to use a switch. These are covered in the next few activities.

Introduction to Switch Statements

A switch statement which is also known as a case statement is used to test a condition against a list of values. Each value is called a case. A switch statement is a lot like an if/else statement that you have used before. Switch statements are more efficient and faster to use than if/else statements. Switch statements are ideal for the use of menu systems.

Activity 20: Creating your first switch statement

In this program we are going to create a simple grading program where a grade is entered a comment is outputted to the user. Create a new C++ project called “Grade”, enter the below code, save and compile.

```
#include <iostream>
using namespace std;
int main ()
{
  char grade;

  cout<<“Please enter the Students grade “;
  cin>>grade;

  switch(grade)
  {
                    case ‘A’ :
                              cout << “Excellent!” << endl;
                              break;
                    case ‘B’ :
                              cout << “Well done” << endl;
                              break;
                    case ‘C’ :
                              cout << “You passed” << endl;
                        break;
                    case ‘D’ :
          cout << “Better try again” << endl;
          break;

      default :
          cout << “Invalid grade” << endl;
  }
```

```
    cout << “Your grade is “ << grade << endl;


    return 0;
}
```

Try entering the value ‘A’ in to your program. What was your result? Now try typing the value ‘a’ into the program. Did you get an invalid reponse?

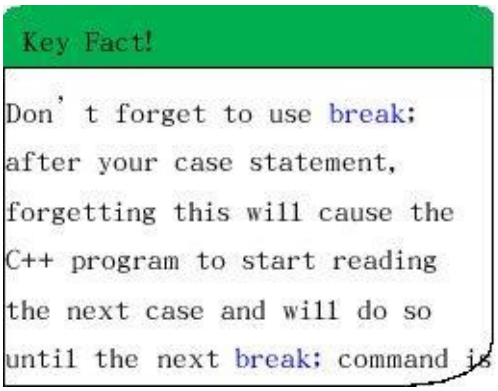
Remember the value ‘A’ and the value ‘a’ are completely different. Lets =go back and amend the code we have created.

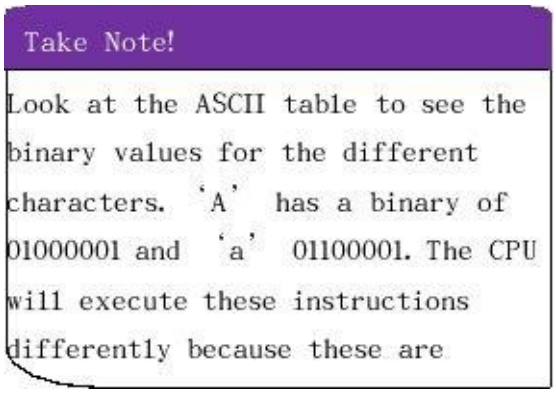
Activity 21: Checking multiple charhcters in a switch statement

Using the previous program, amend your code to look like the new code below. Save, and compile the program.

```
#include <iostream>
using namespace std;
int main ()
{
  char grade;
  cout<<"Please enter the Students grade ";
  cin>>grade;
  switch(grade)
  {
                    case 'A': case 'a' :
                              cout << "Excellent!" << endl;
                              break;
                    case 'B': case 'b' :
                              cout << "Well done" << endl;
                              break;
                    case 'C': case 'c' :
                              cout << "You passed" << endl;
                        break;
                    case 'D': case 'd' :
          cout << "Better try again" << endl;
          break;

      default :
          cout << "Invalid grade" << endl;
  }
  cout << "Your grade is " << grade << endl;

  return 0;
}
```

Try re-entering the values ‘A’ and ‘a’ in again. Did you notice that you got the same result? Well done you have compared the value you have entered against the case of your program.





Activity 22: Creating a menu using switch statements

In this activity you will use integers to create a simple menu for a café. Create a new program called “Café Menu” and enter the below code, save and compile.

```
#include <iostream>
using namespace std;
int main ()
{
 int choice;
 cout<<“Option 1: Eat Breakfast”<<endl;
 cout<<“Option 2: Wash up the dishes”<<endl;
 cout<<“Option 3: Make a drink”<<endl;
 cout<<“What would you like to eat? Please enter a number to make your choice “;
 cin>>choice;

 switch (choice)
 {
        case 1:
                cout<<“Bacon Bap”<<endl;
        break;
```

```
        case 2:
                cout<<"Sausage roll"<<endl;
        break;

        case 3:
                cout<<"Bowl of porridge "<<endl;
                  break;

        default:
                cout<<"invalid choice"<<endl;
    break;
}
        return 0;

}
```

So you now have used case statements for both chars and integers, but something was missing from both programs that can improve the readability of both programs, did you know what? To improve the readability and prevent the chances of something going wrong we should use parentheses {} for each case. So each case should look like this:

```
case 1:
{
        //code here
        break;
}
```

Go back through both of your programs that use switch statements and place the parentheses in, ensure that you compile and test each program to ensure that they work.

Nesting switch/case statements

Nesting switches it's exactly the same as nesting if/else statements it's a switch inside a switch.

Activity 23 – Nesting switch statements

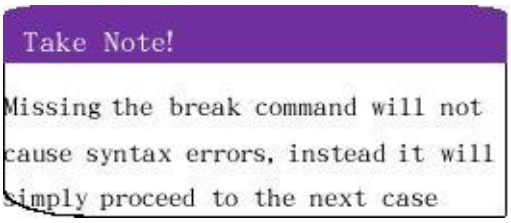
In this program you will create a C++ program that has a nested switch for a cinema system. Create a new program called nesting switch, copy the code, save and compile.

```
#include <iostream>
using namespace std;
int main ()
{
 int choice, film;
        cout<<"1. View Prices"<<endl;
        cout<<"2. View Times "<<endl;
        cout<<"3. View Films"<<endl;
        cin>>choice;
        switch(choice)
        {
                case 1:
                                {
                        cout<<"You have chossen to view prices";
                        cout<<"Student: £5.59";
                        cout<<"Adult: £7.00";
                        cout<<"Concession: £5.40";
                        break;
                }
        case 2:
                                {
                        cout<<"what film would you like to watch? " <<endl;
                        cout<<"1. The Hobbit: The Battle of Five Armies "<<endl;
                        cout<<"2. The Imitation Game"<<endl;
                        cout<<"3. Frozen"<<endl;
                        cin>>film;
                        switch(film)
                        {
                           case 1:
                {
```

```
                cout<<“The Hobbit: The Battle of Five Armies” <<endl;
                cout<<“1. Monday”<<endl;
                cout<<“2. Tuesday”<<endl;
                cout<<“3. Wednesday”<<endl;
                cout<<“4. Thursday”<<endl;
                cout<<“5. Friday”<<endl;
                break;
        }
        case 2:
        {
                cout<<“The Imitation Game”<<endl;
                cout<<“1. Monday”<<endl;
                cout<<“2. Tuesday”<<endl;
                cout<<“3. Wednesday”<<endl;
         cout<<“4. Thursday”<<endl;
                cout<<“5. Friday”<<endl;
                break;
            }
```

```
        case 3:
        {
                cout<<“Frozen”<<endl;
                cout<<“1. Monday”<<endl;
                cout<<“2. Tuesday”<<endl;
                cout<<“3. Wednesday”<<endl;
                cout<<“4. Thursday”<<endl;
                cout<<“5. Friday”<<endl;
                break;
        }

              default:
        {
               cout<<“You have entered an invalid menu choice”<<endl;
                        cout<<“Please try again”<<endl;
                        break;
        }
          }
```

```
            break;
          }
         case 3:
         {
                        cout<<"The Hobbit: The Battle of five armies"<<endl;
                        cout<<"The Imitation Game"<<endl;
                        cout<<"Frozen"<<endl;
                        break;
         }

             default:
         {
                        cout<<"You have entered an invalid menu choice"<<endl;
                        cout<<"Please try again"<<endl;
                        break;
         }
     }
  return 0;
}
```

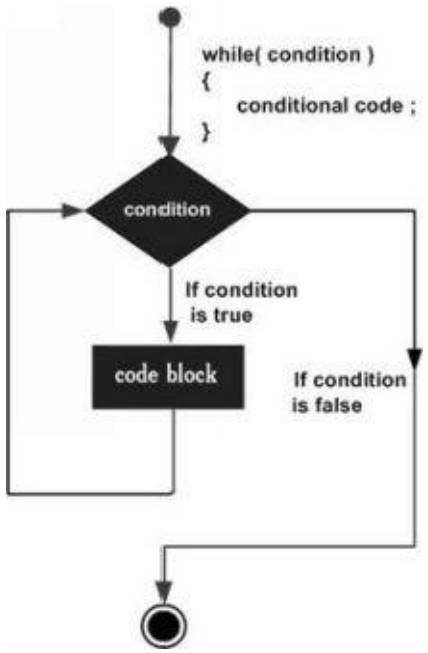


Important! Nesting switches can get confusing and most of the errors you will encounter will be because you haven't closed your case using parentheses {}. Please ensure that you check that you close all your cases.

You will have noticed the above program with the nested switch does not contain the times for each film, only the days. You will need to create another nested statement for each for film. This means that your program will contain 3 nested switches.

Introduction to Loops

In programming we use loops to repeat an aspect of a program until a certain condition has been met. Every pass through the loop is known as iteration, which is one of the basic three structures of programming (sequence, selection and iteration). In programming we use three types of loops white, for, and do. While.

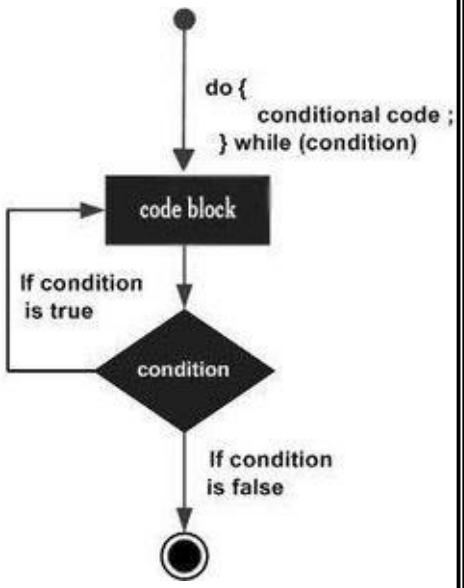


While Loops

While loops will repeat a statement while a condition is true. This loop is the only loop that will test the condition **before** the loop command is executed. The flow diagram to the left shows how a while loop works.

While loops are based on Boolean conditions, just like conditional statements

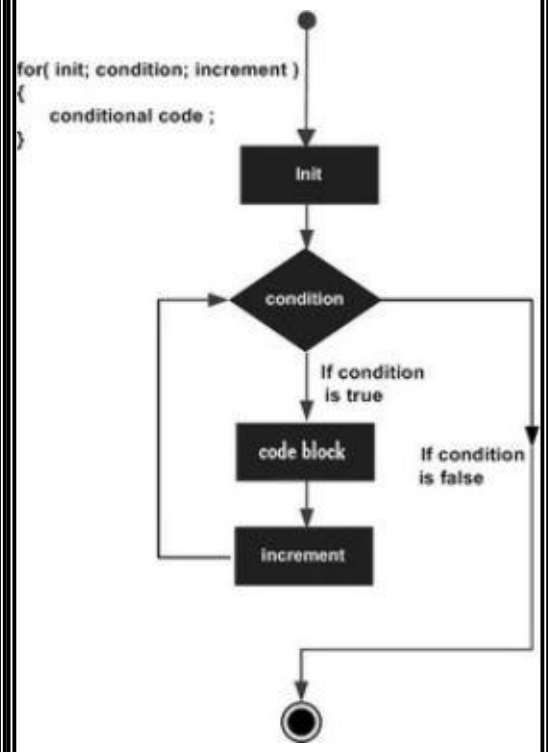
```
while (true)
{

  //do stuff

 }
```



Do.. While Loops

A do.. while loop is the opposite to a while loop as this will check the condition after the code has been executed. Unlike a while loop, a do.. while loop is guaranteed to be executed at least once. The flow chart to the left shows how a do while loop works.

```
do {

  //do stuff
} while (condition);
```



For Loops

For loops are designed to be executed a number of times and are ideal if you want a specific of code to be repeated. For example for loops are commonly associated with arrays as we are able to print multiple elements of an array in one go.

```
for (INITIALIZATION; CONDITION; AFTERTHOUGHT)
{
   // Code for the for loop's body
   // goes here.
}
```

```
for (int i = 0; i < 100; i++) {

        //code here
}
```

Loop counters

A loop counter is used to control the amount of iterations a loop performs. Loop counters can be used to increment the iterations of a loop denoted by ++ or can be decrement a loop —. A common identifier for counters are used with variable names called I, j, and K etc.

Take Note!

Sometimes it is necessary to break out of a loop, this can be achieved by using the break command. The break command you have used in previous activities such as the switch statement, to break out of the case.

Key Fact!

Loops are the act of repeating something to achieve a target or goal. Each repetition of a loop, is known as an iteration. Iteration is describes the style of programming used in imperative programming

While loops

The while loop is the simplest kind of loop that you can use in C++. The while loop will repeat the statement whilst the condition is true. When the loop is executed, if the condition was to change to false, then the loop will no longer run.

Activity 24 – While loops (decrement counter)

Create a C++ program, called “Lift off” and copy the below code, save and compile.

```
#include <iostream>
using namespace std;

int main ()
{
 int num = 10;

 while (num>0) //repeat while num is less than 0
 {
  cout << num << “, “;
  —num; //decremental counter. Every time the loop repeats num-1.
 }

 cout << “liftoff!\n”;

 return 0;
}
```

Activity 25 – While loop (Increment counter)

Create a C++ program called “counting” and copy the below code, save and compile.

```
#include <iostream>
using namespace std;

int main ()
{
 int num = 0;
```

```
while (num<10) //repeat while num is less than 10
{
  cout << num << ", ";
  num++; //incremental counter
}

cout << "finished\n";

return 0;
}
```

Create a grading program that asks the user to enter a grade. While the grade is below 70% display the message "That's not good enough try again".

Do… While Loops

The do..while loop is guaranteed to execute the code contained inside at least once, as the condition of the loop is checked at the end. If the condition is true, the loop will run again, if not then the loop will not run.

Activity 26 – Do While.

In this program, we ask the user to enter a number greater than 10, if they enter a number below 10, they will be prompted to enter another number. Copy the code below into a new project called “Do while” save and compile.

```
#include <iostream>
using namespace std;
int main()
{
        int num;
        do
        {
                cout<<“Please enter a number greater than 10: “;
                cin>>num;

        }while(num<10);
        return 0;
 }
```

Activity 27 – Do while, with NOT operation

Create a new C++ program enter the below source code, then save and compile.

```
#include <iostream>
using namespace std;
int main()
{
        int num;
        bool done = false;
do{
        system (“cls”);//clears the screen down
        cout<<“Please enter a number: “;
        cin>>num;
```

```
        cout<<num<<endl;
        system("pause");
        int num = 0;


        }while (!done);
return 0;
 }
```

In activity 23, you were asked to code a C++ program which had nested switch statements. Using the same code for that program, wrap the entire code in a do while loop, to allow the user to repeat the program again without exiting out of it.

Activity 26 - Fix the errors

Below is a C++ program that has several errors in, using what you have learnt create a new C++ project called "Errors with Loops", copy the below code and fix the errors. Save and compile to ensure all errors have been resolved.

```
#include <iostream>
using namespace std;
main()
{
        int num1, num2;
        char again;

        do
        cout << "Enter a number: ";
        cin >> num1;
        cout << "Enter another number: ";
        cin >> num2;
        cout << "Their sum is " << (num1 + num2) << endl;
        cout << "Do you want to do this again? ";
        cin >> again;
        while (again == 'y' || again ++ 'Y');
        return 0;
 }
```

Key Fact!

Break statements can be used anywhere in a loop, they are especially useful for when a condition has been met, and the loop needs to be terminated.

Take Note!

Be very careful where you position your break statements. If they are positioned in the wrong place they will cause an infinite loop. If you start an infinite loop press the ctrl+c, and

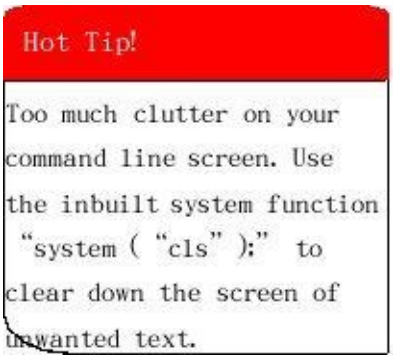
Activity 27: Why will is this loop still looping?

In the below code, is a program that compiles, but we have something called an infinite loop. An infinite loop is a loop that once started, will continue to loop and not break. Your task is to find where the infinite loop takes is, and resolve it:

```
#include <iostream>
#include <string>
using namespace std;
string first, DoB;
char gender;
int main()
{
        int choice, info, loop = 1;
        do {
                while (choice = 2)
                {
                        cout << “Enter following information is inputted:” << endl;
```

```
                        cout << “1. Name: “ << first << ” “ << last << endl;
                        cout << “2. Date of Birth: “ << DoB << endl;
                        cout << “3. Gender: “ << gender << endl;
```

```
                cout << "4. Complete application." << endl;


                cin >> info;
                switch (info)
                {
                case 1://name
                {
        cout << "Please enter you first name: " << endl;
                cin >> first;
                break;
                }
                case 2://age
                {
                cout <<"Please enter your date of birth(DD/MM/YYYY):"<< endl;
                                cin >> DoB; break;
                }
                case 3: //gender
                {
                        cout <<"Please enter your gender(M/F):" << endl;
                                        cin >> gender; break;
                }
                case 4://complete application
                {
                                cin >> choice = 0;
                                break;


                }
                default: "not a valid option";
                        break;
                }
            }
        } while (loop = 1);
        return 0;
}
```



You have now learned two types of loops, create a program of your choice, that uses both a while and a do.. While loop. You can use previous programs to help you structure your loops.

Activity 28:

Below is a program that calculates your age, based on your year and month of birth and compares it to the current month and year. Adapt this program for an off license that while the user's age is below 18 the user can't have any alcohol.

```
#include<iostream>
using namespace std;
int main()
{

        int birthmonth, birthyear;
        int currentmonth, currentyear;
        int agey, agem;


        cout << " The Age Calculator"<<endl;
        cout << "Enter Your Birth Year(Eg:1993):"<<endl;
        cin >> birthyear;

        cout << "Enter Your Birth Month(Eg:5):"<<endl;
        cin >> birthmonth;
```

```
        if (birthmonth > 12 || birthmonth < 1)
                return 1;

        cout << “Enter The Current Month(Eg:7):”<<endl;
        cin >> currentmonth;
        cout << “Enter The Current Year(Eg:2010):”<<endl;
        cin >> currentyear;
        agey = currentyear - birthyear;
        agem = 12 - birthmonth;
        cout << “Your Age is “ << agey << ” Years And “ << agem << ” Months “<<endl;

        return 0;
}
```

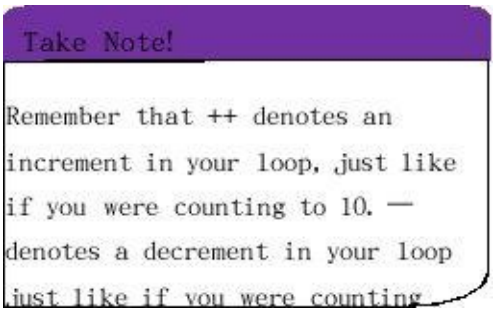
For Loops

A for loop is a special type of loop that will loop a specific amount of types, unlike a while and do while which will only loop if a condition is true. For loops are particularly useful when dealing with data structures like arrays, which will be covered later in this guide.

Activity 28

In this program, we are going to count how many loop iterations have taken place. A loop iteration is known as how many times a loop is executed. Create a new C++ program called “for loops” enter the below code, save and compile.

```
#include <iostream>
using namespace std;
int main()
{
	int i;
	for (i=1; i<11; i++)
	{
		cout<<“loop iteration: “ <<i<<endl;
	}
	return 0;
}
```



How many iterations did you get? If coded correctly you would have gotten 10. If you change the number 11 in the above program to another number you will get (n-1) of iterations. Give it a try and see what you get.

Activity 29

In this activity we have changed the rules of the for loop to count down from 10 rather than count up. Compare the conditions and the counter from the previous program. Create a new C++ project called “count down”, copy the code, save and compile.

```

```

```
#include <iostream>
using namespace std;
int main()
{
	int i;
	for (i = 10; i>0; i—)
	{
		cout << “loop iteration: “ << i << endl;
	}
	return 0;
}
```

Take Note!

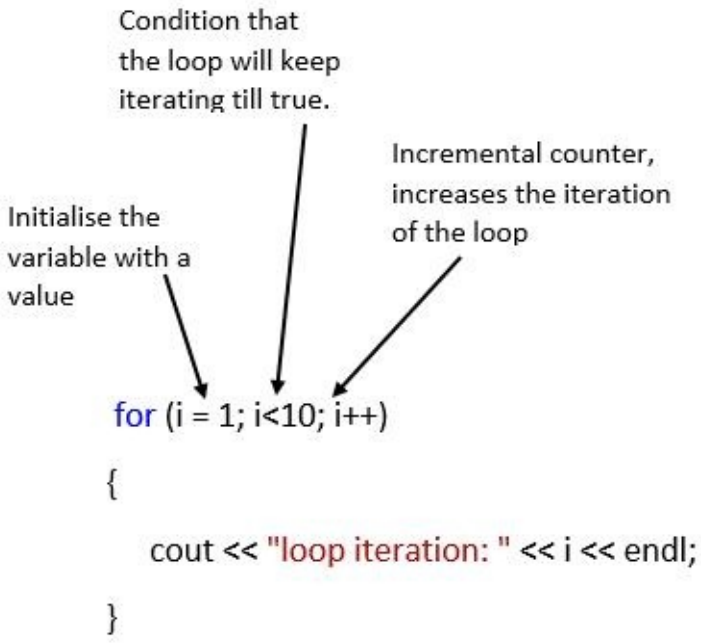
The variables used in for loops can be declared in two ways:

```
for (int i=1; i<4; i++)
```

or

```
int i;
for (i=1; i<4; i++)
```

Either way is fine, however when you aim to



Understanding the syntax of a for loop

Activity 30 (Nesting Loops)

In this program you are going to create a program that uses nested for loops. Create a new project called “nested for loops”, copy the code below, save and compile.

```
#include <iostream>
using namespace std;
int main()
{
```

```
        int i, j;
        for (i=1; i<4; i++)
        {
                cout<<“loop iteration: “ <<i<<endl;
                for(j=1; j<4; j++)
                {
                        cout<<” Inner loop iteration: “<<j<<endl;
                }
        }
        return 0;
}
```

Introduction to Functions

Did you know?

That you have been using functions since your first C++ program? "main" is a type of function, and is the primary function of

Functions are dedicated blocks of code that perform a specific task. They are used to break code into smaller chunks making the program easier to understand. Functions promote reusability and are useful for saving space in the program.

The activities below show the different types of functions, and how they are used within the C++ environment.

Activity 31

In this program, we care going to create a simple function that uses the type void. Void should only be used when a function does not return any value. For example a function that only prints out statements (like the one below) will only need to use the void type. Create a new program called "My first function", copy the below code, compile and run.

```
#include <iostream>
using namespace std;

void banner()//code function here
{
        cout << "I am a simple function"<<endl;
}

int main()
{
        banner();//call function here
        return 0;
}
```

Take Note!

To prevent errors and save lines of code, ensure that your functions are declared be above your main function.

Key Fact!

Remember no matter how many functions you have, and the order they are coded in, the "main" function will always be called first. This is the same for all C++

Create a program that uses the void function that you have learnt from the previous program. Your program will need to at least call this functions three times in different locations of your program.

Activity 32

In this program, we are using functions to pass data between each other. This method is in replacement of Global variables, and is the best practice for allowing specific functions to access certain data, without compromising its integrity. The program below, has a function called “add” that is responsible for adding two values together and returning the sum of these two values to the “main” function, which is then printed out to the screen.

Create a program called “Function Addition”, copy the code below and compile.

```
#include <iostream>
using namespace std;

int add(int num1, int num2)//code function here with parameters
{
        int sum = num1 + num2; //
        return sum; //value of sum is the number that is returned to main
}

int main()
{
        int result = add(2, 3); //pass the values of 2 and 3 to the add function
        cout << “The result is “ << result <<endl; //
}
```

Understanding what the program is doing line by line.

This program has two functions, one called “add”, and the other called “main”. These functions have both been declared as type integers, as they are both dealing with numbers.

int add(int num1, int num2)

In this line, we are creating a function called “add” that is of type integer, which has two parameters. A parameter is a variable that expects data to be passed to it from another function, these are known as arguments. This function has two parameters both of type integer called num1 and num2.

int sum = num1 + num2;

A local variable called sum, has been created to add the values of num1 and num2 together. This is something you should be comfortable with by now, as you have used this type of declaration in previous C++ activities.

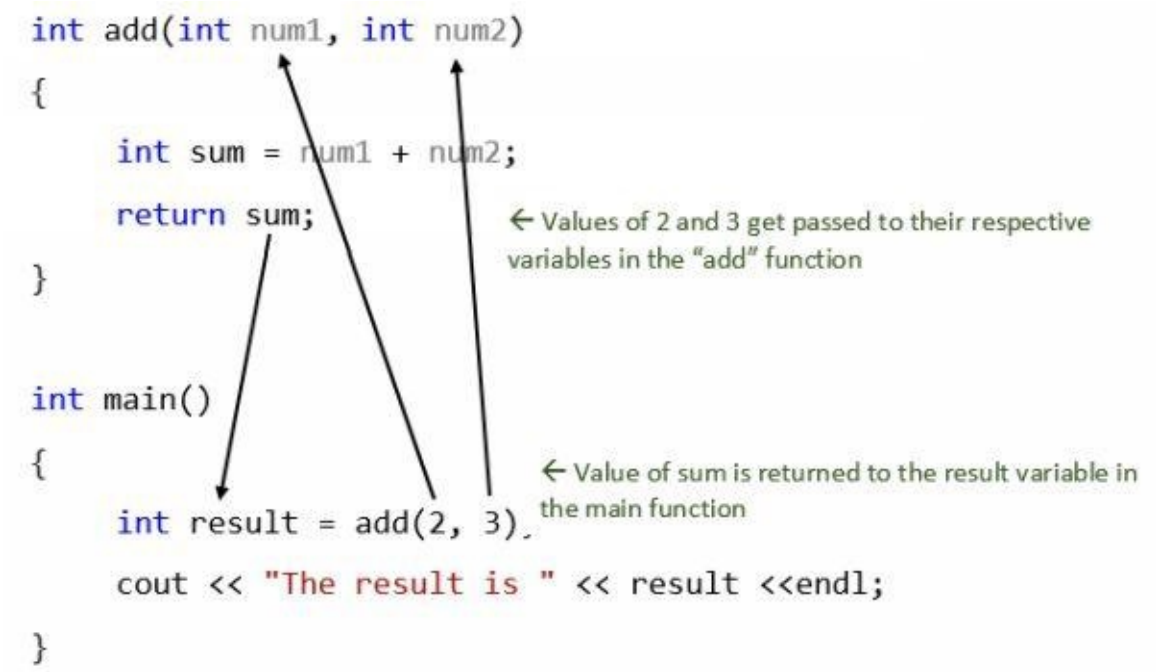
return sum;

Like return 0; this ends the “add function”, but instead it carries the total value of num1 and num2. The value will transferred to the function that called the “add” function. In this program that value would be passed to the “main” function.

int result = add(2, 3);

In this line of code we create a new variable called result that uses type integer. We also call out function “add” and we pass two arguments to that function.

These arguments are the values 2 and 3, which are stored in the variables num1 and num2 in our “add” function. In the last line of codereturn sum the total of 2 and 3 which is 5, is returned to the main function. The value of 5 is stored in our variable calledint result.This result is then printed to the screen.



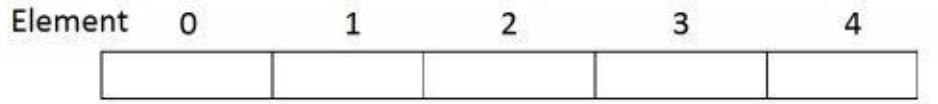
Do it yourself

Exercisell

You have been showed, how to create two types of function, one a simple function that uses type void and the other that uses type integer. In this activity you will need to create a C++ program that uses two functions. The program will need to ask the user to enter two numbers, which will multiply them together and display the final sum output.

Introduction to Arrays

Arrays are a type of data structure that stores a set of a particular data, which has been placed in contagious memory location. Arrays should be looked at more than just a set of data, but a set of variables of the same data type. Each piece of data stored in array is known as an element and his its own unique identifier. Regardless on the length or type of array that is used the first element is always a 0. Take a look below at the below example.



In the above example, we have 5 blocks which can store 5 sets of data of the same type. When we declare an array, we declare it like a variable as we have to give a data type and a name, but we also have to declare how many elements we expect our array to hold. We do this by using square brackets [] with a number inside. Based on the example above our arrays has 5 elements [5]. Below is an example of how a simple array is declared in C++:

Key Fact!

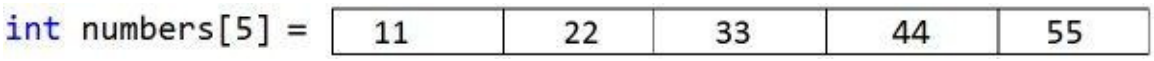
The arrays that you are working with are known as one dimensional or linear arrays. Like one dimensional, Multi-dimensional arrays allocate a single

```
int numbers[5];
```

Like variables arrays can either be initialised with predefined values when they are declared. To initialise an array with a set of values, we need to use a set {} to store that data. Based on the examples we have used so far, this is how you would initialise an array:

```
int numbers[5] = {11, 22, 33, 44, 55};
```

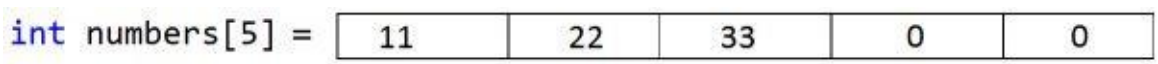
As we have initialised all elements of our array, this is how our array would look like based on the first example above:



You may find yourself in situation, where you only need to initialise a few elements of the array, for example you may initialise the array like below:

```
int numbers[5] = {11, 22, 33};
```

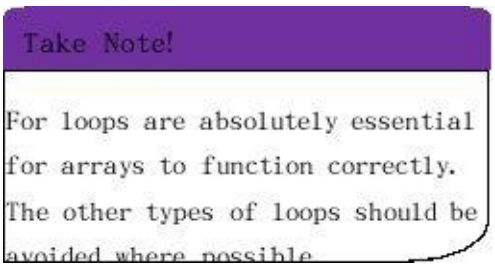
By initialising the first 3 elements of the array, the other elements have been left undefined and as a result, will default to the value of 0. See the below example.



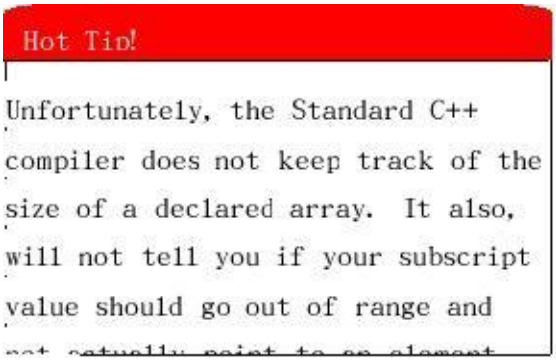
Activity 33: My first array

In this program we are going to create the exact array that we have been working with in the example above, and we are going to print all our elements out to the screen. To print out each element of the array, we need to cycle through it using a for loop. Create a new project called “My First Array” copy the below code, save and compile.

```
#include <iostream>
using namespace std;
int main()
{
	int numbers[5]{11, 22, 33, 44, 55};// array declared and initialised

	for (int i = 0; i < 5; i++)//for loop created with correct loop iterations
	{
		//loop iterations i, will print the correct element of the array
		cout << numbers[i] << endl;
	}
	return 0;
}
```



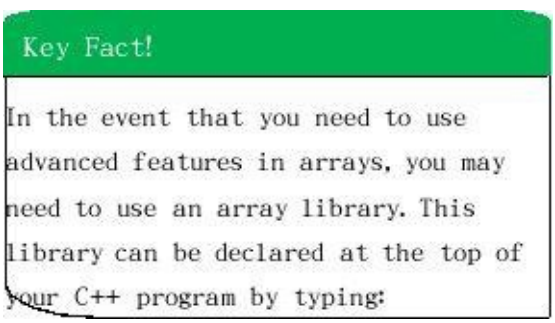
In this program the for loop is the key to the working of the array. We have created a new variable calledint i which is used to help us locate which element in the array needs to be printed, once the for loop completes its iteration the counter then goes to the next element of the array.



Activity 34: Entering Data into an array at run time

In this activity we are going to declare a new integer array and initialise the value of the array at runtime. Create a new project called “Array Input”, enter the code below, save and compile.

```
#include <iostream>
using namespace std;
int main()
{
	int numbers[5];
	int i;//int i declared here so both loops can access the data stored
	cout << “Please enter 5 numbers: “;
	for (i = 0; i < 5; i++)//for loop created to get input
	{

		cin >> numbers[i];
	}

	cout << “The numbers you entered are: “;

	for (i = 0; i < 5; i++)//for loop created to display output
	{

		cout<< numbers[i]<<” “;
	}
	cout<< endl;
	return 0;
}
```



Have you noticed that in this program that the variable “i” we used for the loops, was declared outside of the for loop? This is because

when coding you should consider memory space, and either though you are saving only several bytes, in a larger program it could save much more.

You know how to create a basic array that stores a variety of integers. Create a program in C++, which uses a char array that has the correct amount of elements to store your entire name. Try to aim to print your name out on one line.

Extension Activities

Below is a series of extension activates, with given scenarios. Create a separate program for each scenario.

Extension Activity 1

The RentCo Car Rental Company rents cars for £35 per day plus 10 pence per mile. Write a program for the rental company so that the user can enter the date that the car was rented, the number of days it was rented for, the mileage when the car was rented, and the mileage when the car was returned. Calculate the total charges and display the results.

Extension Activity 2

A simple address book system holds names, addresses and telephone numbers for friends and family. The system stores:

- Surname and first name
- Two lines of the address and a post code
- Telephone number
- Date of birth
- Email address

Write a program to allow the user to input these details and then display them on screen.

Extension Activity 3

Temperatures can be converted from Fahrenheit to Centigrade. Research the formula required to convert a temperature from Fahrenheit to Centigrade. Write a program to input a Fahrenheit temperature and output the equivalent temperature in Centigrade.

Extension Activity 4

Write a program to work out your pocket money spending by carrying out the following steps:

Input a variable pocketmoney to the amount of money you get each month.

Output the title ‘Pocket Money’

Output the number of pounds at the start of the month

Input the number of pounds spent on food

Input the number of pounds spent on your mobile phone

Output the amount of pocket money left

Extension Activity 5

Write a program to calculate the area of a circle. You will have to research formula to calculate the area of a circle based on the radius.

Assign the value 3.14159 to a variable called pi.

Input the radius of the circle

Output the area of a circle

Extension Activity 6

Enter the length, width and depth of a rectangular swimming pool. Write a program to calculate the volume of water required to fill the pool and output the volume.

The ASCII Table

Computers can only understand one language, which is binary. Computers can't understand characters such as '@' or 'y', they can only understand their binary equivalent.

The ASCII (American standard code of information interchange) table shows the numerical representation of a character in decimal, which is then converted to binary by the compilers interpreter.

- Character codes 0 – 31 of the ASCII table are non-printing characters.
- Character codes 32-127 are printable characters and character's
- Character codes 128-255 are the extended ASCII codes for printable characters.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

The ASCII table is extremely important as it helps us understand that no two values are the same. See the below example:

The character 'A' and 'a' to humans have the same meaning (obviously one is a capital letter). But to a computer system they are completely different with two different meanings.

'A' has the value of 65 which in binary is: 01000001

'a' has the value of 97 which in binary is: 01011101

It's important that we understand that these values are different and the CPU handles these values in a different way.

ASCII Table (Binary values 31-127)

Below is the ASCII table for values 31-127. The table shows the decimal, octal, hexadecimal and HTML value of each character within the above range.

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
32	040	20	00100000		&#32;		Space
33	041	21	00100001	!	&#33;		Exclamation mark
34	042	22	00100010	”	&#34;	&quot;	Double quotes (or speech marks)
35	043	23	00100011	#	&#35;		Number
36	044	24	00100100	$	&#36;		Dollar
37	045	25	00100101	%	&#37;		Procenttecken
38	046	26	00100110	&	&#38;	&amp;	Ampersand
39	047	27	00100111	’	&#39;		Single quote
40	050	28	00101000	(	&#40;		Open parenthesis (or open bracket)
41	051	29	00101001	)	&#41;		Close parenthesis (or close bracket)
42	052	2A	00101010	*	&#42;		Asterisk
43	053	2B	00101011	+	&#43;		Plus

44	054	2C	00101100	,	&#44;		Comma
45	055	2D	00101101	-	&#45;		Hyphen
46	056	2E	00101110	.	&#46;		Period, dot or full stop
47	057	2F	00101111	/	&#47;		Slash or divide
48	060	30	00110000	0	&#48;		Zero
49	061	31	00110001	1	&#49;		One
50	062	32	00110010	2	&#50;		Two
51	063	33	00110011	3	&#51;		Three
52	064	34	00110100	4	&#52;		Four
53	065	35	00110101	5	&#53;		Five
54	066	36	00110110	6	&#54;		Six
55	067	37	00110111	7	&#55;		Seven
56	070	38	00111000	8	&#56;		Eight
57	071	39	00111001	9	&#57;		Nine
58	072	3A	00111010	:	&#58;		Colon

59	073	3B	00111011	;	&#59;		Semicolon
60	074	3C	00111100	<	&#60;	&lt;	Less than (or open angled bracket)
61	075	3D	00111101	=	&#61;		Equals
62	076	3E	00111110	>	&#62;	&gt;	Greater than (or close angled bracket)
63	077	3F	00111111	?	&#63;		Question mark
64	100	40	01000000	@	&#64;		At symbol
65	101	41	01000001	A	&#65;		Uppercase A
66	102	42	01000010	B	&#66;		Uppercase B
67	103	43	01000011	C	&#67;		Uppercase C
68	104	44	01000100	D	&#68;		Uppercase D
69	105	45	01000101	E	&#69;		Uppercase E
70	106	46	01000110	F	&#70;		Uppercase F
71	107	47	01000111	G	&#71;		Uppercase G

72	110	48	01001000	H	&#72;		Uppercase H
73	111	49	01001001	I	&#73;		Uppercase I
74	112	4A	01001010	J	&#74;		Uppercase J
75	113	4B	01001011	K	&#75;		Uppercase K
76	114	4C	01001100	L	&#76;		Uppercase L
77	115	4D	01001101	M	&#77;		Uppercase M
78	116	4E	01001110	N	&#78;		Uppercase N
79	117	4F	01001111	O	&#79;		Uppercase O
80	120	50	01010000	P	&#80;		Uppercase P
81	121	51	01010001	Q	&#81;		Uppercase Q
82	122	52	01010010	R	&#82;		Uppercase R
83	123	53	01010011	S	&#83;		Uppercase S
84	124	54	01010100	T	&#84;		Uppercase T
85	125	55	01010101	U	&#85;		Uppercase U
86	126	56	01010110	V	&#86;		Uppercase V

87	127	57	01010111	W	&#87;		Uppercase W
88	130	58	01011000	X	&#88;		Uppercase X
89	131	59	01011001	Y	&#89;		Uppercase Y
90	132	5A	01011010	Z	&#90;		Uppercase Z
91	133	5B	01011011	[	&#91;		Opening bracket
92	134	5C	01011100	\	&#92;		Backslash
93	135	5D	01011101	]	&#93;		Closing bracket
94	136	5E	01011110	^	&#94;		Caret - circumflex
95	137	5F	01011111	_	&#95;		Underscore
96	140	60	01100000	`	&#96;		Grave accent
97	141	61	01100001	a	&#97;		Lowercase a
98	142	62	01100010	b	&#98;		Lowercase b
99	143	63	01100011	c	&#99;		Lowercase c
100	144	64	01100100	d	&#100;		Lowercase d
101	145	65	01100101	e	&#101;		Lowercase e

102	146	66	01100110	f	&#102;		Lowercase f
103	147	67	01100111	g	&#103;		Lowercase g
104	150	68	01101000	h	&#104;		Lowercase h
105	151	69	01101001	i	&#105;		Lowercase i
106	152	6A	01101010	j	&#106;		Lowercase j
107	153	6B	01101011	k	&#107;		Lowercase k
108	154	6C	01101100	l	&#108;		Lowercase l
109	155	6D	01101101	m	&#109;		Lowercase m
110	156	6E	01101110	n	&#110;		Lowercase n
111	157	6F	01101111	o	&#111;		Lowercase o
112	160	70	01110000	p	&#112;		Lowercase p
113	161	71	01110001	q	&#113;		Lowercase q
114	162	72	01110010	r	&#114;		Lowercase r
115	163	73	01110011	s	&#115;		Lowercase s

116	164	74	01110100	t	&#116;		Lowercase t
117	165	75	01110101	u	&#117;		Lowercase u
118	166	76	01110110	v	&#118;		Lowercase v
119	167	77	01110111	w	&#119;		Lowercase w
120	170	78	01111000	x	&#120;		Lowercase x
121	171	79	01111001	y	&#121;		Lowercase y
122	172	7A	01111010	z	&#122;		Lowercase z
123	173	7B	01111011	{	&#123;		Opening brace
124	174	7C	01111100	|	&#124;		Vertical bar
125	175	7D	01111101	}	&#125;		Closing brace
126	176	7E	01111110	~	&#126;		Equivalency sign - tilde
127	177	7F	01111111		&#127;		Delete

To see the full ASCII table from 0-255 then visit www.ascii-code.com

Glossary

Below is some of the key terminology that is used in programming, it is essential that you familiarise yourself with this, as it will be used in lesson.

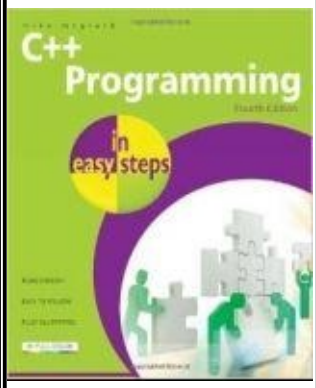
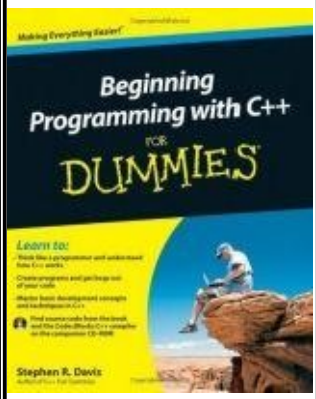
Algorithm	A set of unambiguous steps to solve a given problem
ASCII	American Standard code for information interchange is a character encoding scheme based on the English alphabet, which encodes 128 characters into 7 bit binary numbers.
Array	A group of data objects that have the same size and data type, in a contagious location in memory.
Boolean	A binary variable that can have one of two possible states, true or false.
Character	A single unit of data that could be a letter, symbol or number.
Compiler	A compiler is responsible for turning a programming language into machine code (binary) to be executed by the CPU. Remember the CPU does not understand anything but binary.
Float	A float is a data type that is a
Function	A block of code decimated to perform a particular task.
IDE	Integrated development environment is a set of tools to be able to develop software using a particular programming language. Normally they contain their own compiler.
Integer	A whole number that has no decimal places.

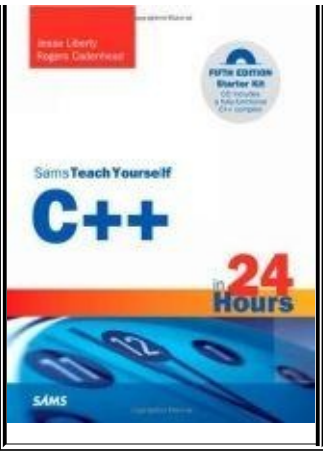
Iteration	Is the act of repeating a particular action
Loop	See definition of iteration
Parameter	A parameter is a variable that is part of a function that receives an argument from another function.
Selection	Also known a decision, is a type of logic structure that depending on the condition will depend on the action that is performed
Sequence	A sequence is a set of steps that go in a particular order. Making a cup of tea is an example of a sequence of steps.
String	A data type used to store a collection of characters, including symbols and numbers.
Variable	Stores a particular type of data, used by a computer program. Depending on the conditions data can change within side the variable.

Further Reading

To help aid you in your experience and development of C++, are some several books and websites that can help you.

http://www.tutorialspoint.com/	http://www.learncpp.com/
http://www.cplusplus.com/	https://www.daniweb.com/
https://isocpp.org/	http://stackoverflow.com/

	**C++ programming in easy steps** ISBN: 978-1840784329 Now, in its fourth edition, C++ Programming in easy steps begins by explaining how to download and install a free C++ compiler so you can quickly begin to create your own executable programs by copying the book's examples. It demonstrates all the C++ language basics before moving on to provide examples of Object Oriented Programming. The book concludes by demonstrating how you can use your acquired knowledge to create programs graphically.
	**Beginning programming with C++ for dummies.** ISBN: 978-0470617977 C++ is an object–oriented programming language commonly adopted by would–be programmers. This book explores the basic development concepts and techniques of C++ and explains the "how" and "why" of C++ programming from the ground up. Assuming no prior experience, Beginning Programming with C++ For Dummies is a fun and friendly guide to learning the C++ language.
	Teach Yourself C++ in 24 hours ISBN: 978-0672333316 From the compulsory explanation of basic syntax and data types

	through to object-orientation, polymorphism and more this is a measured treatment of the subject at hand which is written in an engaging way making it a pleasure, rather than a chore, to read. The time estimate of one hour per lesson seems a little on the conservative side and especially latter chapters could benefit from rather more than 60 minutes of study, but this is a minor criticism at the end of the day.

-

Notes

Table of Contents