

Beginner's Guide To C++

Lesson 1: The basics of C++

This tutorial is written for everyone: even if you've never programmed before or if you have used other languages and want to expand into C++! It is for everyone who wants the feeling of accomplishment from a working program.

Getting started

If you don't have a compiler, I strongly suggest that you get one. A simple compiler is sufficient for our use, but make sure that you do get one in order to get the most from these tutorials. Personally I use Dev C++ and this will be the compiler I use in this tutorial, it has an easy to use GUI and can compile directly from this interface opening in a separate console screen.

C++ is an object orientated programming language in other words it can have several parts known as classes or object which can be called by a driver program, the main function. This means that we can also include other files and use functions from them, however as this is only a basic guide to start you programming we only look at this from the perspective of including some basic libraries that come with most compilers.

The first thing you need to know is the basic code you need to start programming

First off we need to include the iostream file to do this, at the top of our document we type:

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

This allows us to use classes from the iostream library, these are used for the basic input and output commands, also we must have a main() function, at least if we want to run the program (its not needed if your creating class to be used by others, however this is outside the scope of this tutorial.) inside the "main()" we write our code as you can see below:

Note: comments can be added using "/" for one line comments or /* comment */ for more then one line as seen below

```
/* this is a program to demonstrate the basics of a c++ programme,
```

```
it includes comments, header files, the main() function and cout*/
```

```
#include <iostream>    //includes the header file iostream  
using namespace std; //tells the compiler to use the std library
```

```
int main()           // declares the function main which returns an int  
{                   //tells the program to take everything between here and the close as main  
cout<<"Hello World!\n"; // displays Hello world on the console
```

```
cin.get();          // stalls the program until a key is pressed  
return 0;           // returns 0 (a 1 would indicate an error  
}                   // closes main
```

Let's look at the elements of the program in more detail. The #include is a "pre-processor" that tells the compiler to put code from the header called iostream into our program. By including header files, you can gain access to many different functions. the cout function requires iostream. Following the include is the statement, "using namespace std;". This line tells the compiler to use a group of

Beginner's Guide To C++

functions that are part of the standard library (std). By including this line at the top of a file, you allow the program to use functions such as cout. The semicolon tells the compiler that this is a line that should be executed. You will see later on that the semicolon is used to end most but not all commands in C++. The "cout" function tells the compiler to display the contents on the screen. cin.get() is used to hold the program from completing until a key is pressed, and finally the "return 0;" is used to return the exit code 0(no errors)

The cout function can be manipulated in many ways, it can display hard coded text but also it can display variables, each type must be separated using << for example:

```
cout<<"the number is "<<var1<<endl;
```

The above will print out "the number is " the value contained in var1 and move to the next line. The command cin.get() is another function call: it reads in input and expects the user to hit the return key. this command keeps the program from closing until the user enters a key it expects you to hit enter. At the end of main, the closing bracket, our program will return the value of 0 (and integer, hence why we told main to return an int) to the operating system. This return value is important as it can be used to tell the OS whether our program succeeded or not.

You can cut and paste the code above into a file, save it as a .cpp file(tells the OS its c++ source code). In Dev C++ you can compile and run the code by clicking the colourful button in the top left hand corner of the window. You might start playing around with the cout function and get used to writing C++.

Comments are critical for all but the most trivial programs and this tutorial will often use them to explain sections of code. When you tell the compiler a section of text is a comment, it will ignore it when running the code, allowing you to use any text you want to describe the real code. When you are learning to program, it is useful to be able to comment out sections of code in order to see how the output is affected.

Variables

A very important part of this tutorial is the use of variables, C++ allows for many types of variables these include integers (int) number values, floating points (float) numbers with decimal points, characters (char) ASCII characters, strings (string) strings of characters. The advantage of variables is that their value can change at any point, all variables must be declared, they are given symbolic names instead of values and when called read the data stored at the variables address.

```
/*to demonstrate variables */
```

```
#include <iostream> //includes the header file iostream  
using namespace std; //tells the compiler to use the std library
```

```
int main()  
{  
int number;  
float decimal;  
char character;  
string word;  
  
number = 1;  
decimal =1.1;  
character = 'f';
```

Beginner's Guide To C++

```
word = "hello";

cout<<number<<" "<<decimal<<" "<<character<<" "<<word<<endl;

cin.get();
return 0;
}
```

Copy this code compile it and check out the result, have a try at switching it around a bit see what you can do, variables can be used to store calculations as well. for example you switch the line "number = 1;" to "number = 12 + 14;" and run it again, this gives number the value 26, there are several other operators, "+" means addition, "-" means subtract, "/" means divide, "*" means multiply, "%" means modulus, in case you don't know modulus gives you the remainder of a division, play around try out the different operators.

You can also add, subtract, multiply and divide variables like `var1 = var2 + var3`, however you do have to assign a value to `var2` and `var3` before hand.

User input

The `cin` function reads in values in much the same format that `cout` outputs values, there are however some flaws that must be addressed, if you try to read in multiple values after entering the first value the program reads the value into the variable how ever the next `cin` might take the return pressed on the first time, this is solved using `cin.ignore()` which is a dummy that is used to take the return without assigning it to a variable, a simple input should look like this:

```
/* to demonstrate cin function*/

#include <iostream> //includes the header file iostream
using namespace std; //tells the compiler to use the std library

int main()
{
int var1;
char var2;
cin>>var1; //reads in a value from the keyboard
cin.ignore(); //reads a dummy
cin>>var2; //reads in a value from he keyboard
cin.ignore(); //reads a dummy
cout<<var1<<var2<<endl; //prints values entered

cin.get();
return 0;
} //end main
```

`cin` can be used to scan in strings as well in the same way.

```
/* to demonstrate cin function with strings*/

#include <iostream> //includes the header file iostream
using namespace std; //tells the compiler to use the std library

int main()
```

Beginner's Guide To C++

```
{  
string var1;  
cin>>var1; //reads in a value from the keyboard  
cin.ignore(); //reads a dummy  
cout<<var1<<endl;//prints values entered  
  
cin.get();  
return 0;  
}//end main
```

Now we have come to the end of this tutorial you should be able to make some simple programs in the next tutorial I will cover if, while, do while and for statements, I will also cover the basics of arrays.