

Remember "Reviewing C++" has many more examples and topics.

Tutorial 1 **Some C++ Basics**

Basics

C++ is what's called a high level programming language. It is more friendly to see and use compared to the lower level languages such as Assembly language.

Every C++ program, at the BARE MINIMUM, will look like this:

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

int main(){

    return 0;
}
```

If you ran the program, it will do absolutely nothing! There is no code that will either output something or ask for an input. Let's dissect some of the things above.

The **#include** statement is used to import any library that will be used by your C++ program. Here, the `iostream` library will be used. This is the main library as it handles the input/output for a program. Without this, there is no i/o.

The **using namespace std;** statement will allow you to use anything in the "standard" namespace of C++. Some of those are `cout` and `cin` which we will see soon.

The **int main()** is called a function (see tutorial 5) and will be the starting point for ANY PROGRAM. When the program runs, the main function is "called" and the program will begin.

The **return 0;** statement will tell the computer that the program is finished and to return to what it was doing before. In a more general sense, it tells the computer that the function called main is finished (again, see tutorial 5).

Example 1: **Hello World**

Remember "Reviewing C++" has many more examples and topics.

This program is the bare minimum of anything. Here, it will output a friendly message to the screen and terminate.

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

int main(){
    cout << "Hello world!" << endl;
    return 0;
}
```

There are some new things introduced here. First, there is a **cout** statement. Notice the word out nested inside the statement. It is used for outputting things on screen. the cout statement is being used from the standard library. The format for a cout statement is the two less than signs next to each other. After those signs, there may be a variable, string, character, end line etc...

The next thing that is new is the **endl**. This stands for end line and is used to go to the next line for output purpose. This is similar to the carriage return for typewriters.

Before viewing the next example, a program can also have input with it. In C++, you would use **cin** followed by two greater than signs >> followed by the variable(s). Variables will be discussed in the next tutorial section (tutorial 2).

Documentation

An essential part of any program is the documentation. Without it, you may have a lot of trouble debugging or get lost if you have a long program. Also, if you are working on a project for a corporation and you switch control over to another person or group, they would like to have an idea about what the project is doing.

In C++, you can do this in two ways. The first is an **in-line comment**. It is denoted by two forward slashes. The rule is that only what is on that line will be "commented out." Below is what this looks like. The comment line appears above the include statement:

```
// our first program
#include <iostream>
using namespace std;

int main(){
    cout << "Hello world!" << endl;
    return 0;
}
```

Remember "Reviewing C++" has many more examples and topics.

```
}
```

Another form of a comment is a **block comment**. With this, more than 1 line can be used for a comment. In terms of format, it begins with a forward slash and then an asterisk. Whatever appears between the begin and end comment mark will be "commented out". Here is what this looks like:

```
/* our first program
   by www.cstutoringcenter.com
*/
#include <iostream>
using namespace std;

int main(){
    cout << "Hello world!" << endl;
    return 0;
}
```

Just notice that the end mark is a bit different than the beginning mark. In general, comments can be placed anywhere in a program. The compiler will not read the comments as the symbols tell the compiler to ignore what is next to or in between the symbols.

Intro to Strings

The program below, example 2, uses a **string** variable to store the name of the user. By definition, a string can be anything grouped together by quotes. A more technical definition is a variable more than 1 character in length and non-numeric. One string can be "anfgebrdasjdnqiuhr48" or "Hello there" or "12345". You get the drift.

In order to be able to use the string variable type, you must include another library to the program called the string library. More on this will be discussed in tutorial 8.

Example 2: Hello [your name]

This program will ask the user to input their first name. It will then output a message using their name.

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

int main(){
    string name = "";
```

Remember "Reviewing C++" has many more examples and topics.

```
cout << "Enter your first name: ";  
cin >> name;  
  
cout << "Hello " << name << endl;  
return 0;  
}
```