Lesson 1 - Hello World!
Basics of Program Creation and Structure

Summary

We want to display hello world, get some input from the user and store the input in a variable.

New Concepts

The Main Function, variables, user input/output (command line) and namespaces.

Hello World

Below we have our first C++ program. Let's step through it:

```
 1 #include <iostream>
 2 
 3 using namespace std;
 4 
 5 int main() {
 6 
 7    cout << "Hello" << ' ' << "World" << "\n";
 8    std::cout << "What is 7 + 2?" << "\n";
 9    int x;
10    cin >> x;
11 
12    return 0;
13 }
```

helloworld.cpp

On line 1 we first include the libraries we want. `iostream` is a very useful library and allows the use of the `cout` identifier (something we will use in our program). An identifier is simply a way of naming something in C++. There are many different "types" of "somethings" in C++, but for now we are concerned with `cout`. `cout` is an object of class `ostream` and represents standard output. Objects and classes will be covered later, but for now we can simply consider it as a mechanism to send text to standard output. Standard output is by default sent to the screen, but it could be sent to a file, network or anything which the computer can send data to.

On line 3 we declare that we are using the namespace `std`. All standard libraries must be used within the `std` namespace. A namespace is simply a way of grouping related identifiers together and avoiding confusion. For example, we may have an identifier "add" that is in the namespace "numbers". We may also have an identifier "add" in the namespace "list". The namespace mechanism allows these two different "adds" to exist in the same program by allowing the programmer to describe their context (adding numbers or adding to a list).

Identifiers that are recognised as standard in C++ all belong to the `std` namespace. By describing that we are going to use the `std` namespace throughout our program we don't have to worry about it. The term "standard" actually alludes to a recognised understanding of what is and is not C++. A group of individuals and companies decide this, but not all versions of C++ are mirror images of the standard.

On line 5 we begin the `main()` function. A function is a collection of code statements that do something (we hope they do something!). `main()` is special as it tells the computer that this is where our program starts (all programs start at `main()` in C++ when you run them). `int main` means that when the function is finished an integer is returned (i.e., whoever asked for the function to execute will receive an integer value when the function finishes). The value of the integer is typically used to signal the presence of any errors if the program could not execute.

All functions have a start and an end and this is shown by a `{` and a `}` respectively. So, all the statements inside `{ }` make up a function. Each line in the function is a statement and the computer knows when we reach the end of a statement as the special character `;` is reached. A statement allows a programmer to conveniently identify the steps in a program. In this first statement on line 7, we can see that `cout` is been passed the phrase `"Hello"`, `' '`, `"World"` and the odd looking `"\n"`. Using double speech marks (`"`) is the way in which C++ identifies a string and a string is simply an ordered list of characters. Single speech marks (`'`) identify a single character in C++ (which is a space in this statement). The `"\n"` is a special command used to identify end of line (i.e., it says that the next output will appear on a new line). The `\` is what we usually call an escape character as it describes that the next character(s) are to be treated differently (in our statement we escape from one context (string) to another (control)).

The `<<` symbol is an operator which `cout` uses to identify what is to be sent to the output. Operators are simply items that apply some well known (well known to the programmer at least) operations. For example, $+$ and $-$ are well known mathematical operators in C++.

The statement on line 8 actually shows how to use a namespace on a per identifier basis. If we had not used `using namespace std;` at the start of the program then this is what we would need to do for the previous `cout` as well. Notice the use of the scope operator (`::`) in linking the namespace to an identifier.

The statement on line 9 creates a variable that can hold an integer value. The identifier of the variable is x and we can refer to x throughout the remainder of this program. We call this type of statement a declaration (we are declaring a variable with an identifier x of type `int` or integer).

We have seen how we can send text to the standard output but what if we want to get something from the outside world? `cin` on line 10 gets items from the standard input (which by default is the command line). The command line is simply the standard window that opens when you run this program. What `cin` does is read a value into the variable x. After `cin` has finished (by reaching the `;` symbol), x will hold whatever the user typed in. Notice how the operators are in the opposite direction for `cin` as opposed to what they were for `cout`. Someone thought this was intuitive (point into the program as opposed to pointing out).

On line 12, `main` returns a value on exiting. This will indicate to the operating system the manner in which it exited. The value 0 usually indicates that the program exited OK. Anything else may indicate problems. However, there is no real standard governing this.

Under the Hood

C++ code is compiled directly into the native assembly instructions of the host machine. Programming languages like Java and C#, on the other hand, compile source to an intermediate byte code which is in turn executed by a virtual machine (usually written in C/C++). This is part of the reason why C++ provides better performance than some other languages. C++ gives programmers greater freedom to manipulate the host platform to squeeze the most efficiency from their machines. In addition, the Object Orientated features that C++ provides, aid programmers in the design of large software projects (i.e. like a Computer Game). As a consequence, C++ is still widely regarded as the language of choice for Game Engine Programming where speed and good design are critical.

Exercises

1. Find out what is the difference between a “signed” and an “unsigned” integer. When might you use an unsigned integer instead of a signed integer?

2. You have seen the “fundamental data type” called `int` but you will eventually need to use others. Amend the program to create a `char` data type, a `bool` data type and a `double` data type. Give them appropriate names and values and print their values to screen using `cout`.

3. Amend the code so that it asks for two numbers from the user. Store these numbers in separate integer variables and then print to screen their sum and product on separate lines. Use `cin` and `cout`—as well as the new line character—to accomplish this task.

4. Amend the code from Lesson 1 so that the program asks for your first name and then prints "Hello (your name)!" to the screen. Use the `cin` and `cout` classes to accomplish this. In addition, use the `string` variable type in the `std` namespace to store your name.