

Lesson 1

Libraries

Like in Processing, C++ uses libraries to expand the number of functions available to us. In Processing the basic functions are available to us without needing to import any extra libraries, whereas in C++ you will need to include libraries for almost anything you want to do. To include a library we do as below.

```
#include <iostream>
```

iostream stands for input output stream and is needed to print to console and read from console.

Namespace

In the event that two libraries you are using have a function with the same name we need a way to differentiate them. We use namespaces to do this.

Using the cout function from the std library.

```
std::cout << "Hello World";
```

As we will be primarily using functions from the std library we can state that unless specified that the functions will be from the std library. We do this with the following code put above main().

```
using namespace std;
```

Java vs C++

C++ is a lower level language than Java meaning that you have more control over how the program runs. This control - especially the ability to directly interact with specific chunks of 1s and 0s in memory - is both an advantage and a disadvantage. (Java manages memory allocation and deallocation automatically). We will discuss this in more depth later.

C++ Hello World program

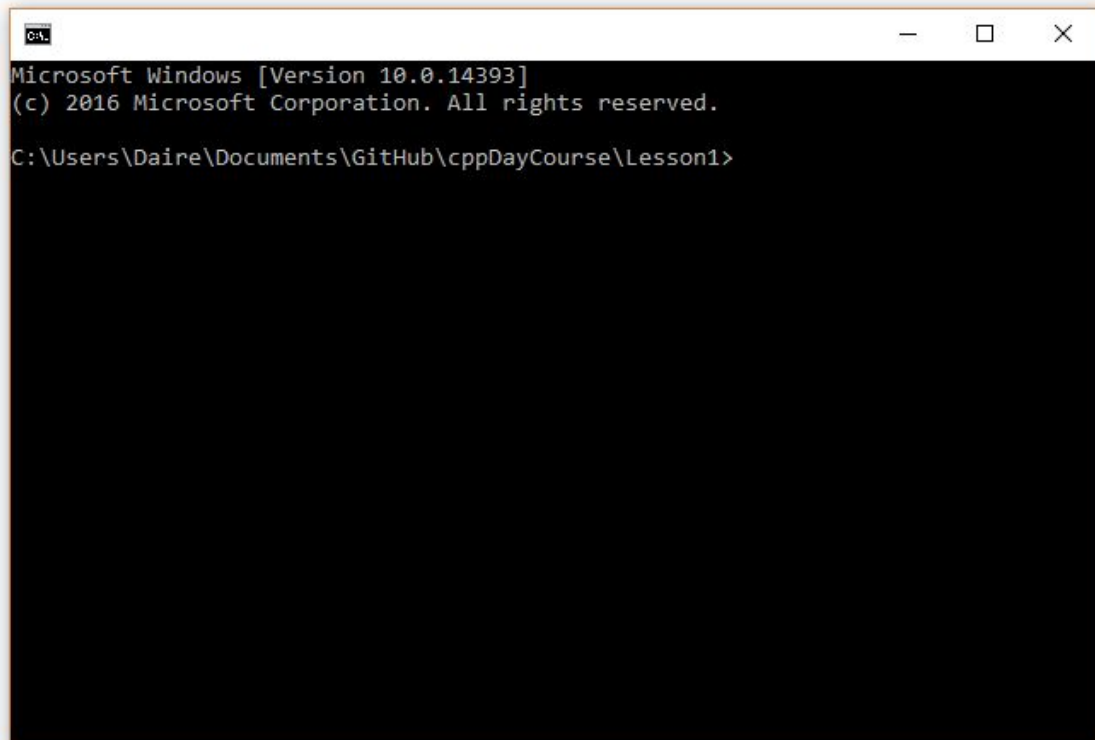
Follow the example code below to write your first C++ program

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      cout << "Hello World";
7  }
```

N.B. Save your program before the next step as something.cpp to get syntax highlighting.

Compiling and Running

As in Lesson 0 we are going to right click on the pane to the left of the Atom editor. This will bring up a list of options, select "Open Terminal at Root". This will open up the console window that will look like this.



Ensure that the console is in the right folder by looking at the name after the last \. In my case this folder is called Lesson1 and is where I saved my program.

First we have to compile our program. To do this type "g++ [the name of your program with extension] -o [the name you want the executable to have]" as I have done below.

```
C:\Users\Daire\Documents\GitHub\cppDayCourse\Lesson1>g++ Lesson1.1.cpp -o Lesson1.1
```

The command above is compiling a program called Lesson1.1.cpp into an executable file called Lesson1.1. **N.B Make sure to save your program before compiling to have any changes in the program reflected in the executable file.**

Next we want to run our newly compiled program. To do this we simply write ".\"(the name of the executable)" as below.

```
C:\Users\Daire\Documents\GitHub\cppDayCourse\Lesson1>.\Lesson1.1
```

This is running an executable file called Lesson1.1. If nothing appears to happen it might be that the program is expecting some user input.

N.B. Be very careful with the names you use when creating the executable file as this can overwrite a program file.

User Input

Before we are able to receive user input we need somewhere to store the data.

- Declare an integer variable called `aNumber` (the same way you would declare an integer in Processing).
- Ask the user to "Pick a number" using `cout`.
- Accept the user input and store it in `aNumber`.

```
cin >> aNumber;
```

- Print out the number.

C++ Functions

Next we are going to look at functions in C++. One important difference to note between functions in C++ and Java is that C++ functions must be declared before they are used (meaning above `main` in the program).

You can think of *main* acting like *void setup* in Processing. Here is where we write the actual program that is going to execute which can call functions from other parts of the program.

There are two ways to do this, the first is to write the whole function before `main` in your code.

```
void exampleFunction() {
    cout << "Example";
}

int main() {
    exampleFunction();
}
```

The second is to declare the function before `main` and define it after. There are various arguments for both styles of function declaration. For now it's sufficient to understand that both are valid options.

```
void exampleFunction();

int main() {
    exampleFunction();
}

void exampleFunction() {
    cout << "Example";
}
```

Following the example above

- Change the code from the previous program into a function
- Write a new function that prints out your name