**Pre-instructions:** Go to <http://www.codeblocks.org/downloads> and follow the instructions for downloading the binary release. If you have any issues, let me know and I’ll help you out.

**Instructions:** Create a C++ project through Code::Blocks.

1. You will open the Code::Blocks IDE.
2. Click on ‘Create a new project’.
3. Choose ‘console application’.
4. Select C++ from the drop-down menu.
5. Give you project the project name: ‘My first c++ project’.
   1. Ensure that your ‘folder to create project in’ (your directory) is set to your desktop
   2. For future projects we will have created a folder on your desktop. We will change the project file directory at that time. Don’t worry about it now.
6. This next window is titled ‘Console application’.
   1. Ensure both check boxes are checked. Then click finish.
   2. Doing this step ensures you can use the debugger and get an executable when you’re done. Don’t worry about the semantics at this point.

**Part 1:**

Great! Now you have created and started your first project. There are a lot of options from within your project’s IDE! Don’t worry about it for now. You’ll likely not use a lot of the options even after you’re C++ proficient. What I do want you to notice is the window to the left that has a file directory:

Workspace

Example

Sources

Click on the plus sign next to Sources to open it up and you’ll see: main.cpp

Double click on main.cpp

Note how you have code to work with. More importantly, it has its own tab up top. Keep this in mind as we get into bigger projects/multiple projects. You can have the code from more than one C++ source file running in Code::Blocks at the same time! This is convenient if you’re swamped with projects, need to reference an old project for inspiration, or simply want to feel busy :P

Before we dive into the code, I want to explain what a cpp file is (file with an extension of .cpp). A cpp file, simply put is a definition file for you C++ projects code. The compiler will take a .cpp file in order to compile and run your code. The compiler can use this file to create and executable (.exe) for a user to run (so that they never have to see or understand your code). Frequently in the class setting (or when collaborating on a project), you will send your .cpp file and not the .exe. This ensures the other user can see the code running the programming (to check your work, see your process, work on the project, etc… Whatever the case may be). Again, do not worry about the definition or semantics of the .cpp or .exe just yet, as you will be exposed to these files a LOT as you start developing c++ projects. Just remember this for now:

.cpp allows you to view the code with an IDE

.exe allows you to run the program without an IDE

**Part 2:**

Now that we’ve established how to create a project and what the files are, I want you take a look at the code. I know that some of you may already be familiar with much of this, but as a review:

#include <iostream>

using namespace std;

int main()

{

cout << “Hello World!” << endl;

return 0;

}

Notice that your IDE numbers your lines of code. This is to help find specific spots for editing, bug fixing, and ease of reading. Please go to the beginning of line 1 and hit enter to move #include <iostream> down to line two. Now type the following into line 1: //This is my first comment!

This line you just type is grayed out because it is a comment (in-line comment). This will help you and future programmers understand your code. It is usually good practice to use many comments to explain your process exactly.

When you need to make a large amount of commenting for a single section of code (or if you want to “turn-off” some of the code you’ve written to test a specific section) you can comment using a different method. It is done in the following way:

/\* All your comments in here \*/

Anything in between your start /\* and finish \*/ will now be grayed out as comments.

Programs commonly have a header at the very top of their code to identify the author (you), the version (date you finished this version), a description of the program and what it does, and any other piece of information that might be useful to someone reading your code later (self-included). Please create a header for you program now utilizing one of the commenting methods I’ve just shown you. [Place your header under your ‘//This is my first comment!’ and before the #include <iostream> line. You can leave blank spaces in-between these sections of code if you wish. Leaving blank lines is known as **white space**. Your IDE and compiler do not care if you use white space and it helps to format the code so that it is easier to read. I will not require you to use white space at this time, but as projects get bigger I will expect that you will have your own “style” of programming that will utilize white space in a way that makes code easier to read.

**Part 3:**

Now I’m going to briefly explain what the pre-populated lines of code mean. You may already be familiar with some of these.

#include <iostream>- The iostream package is included in every C++ (package as far as I’m aware) and is required for even the simplest of tasks to be performed by your code.

using namespace std;- This ensures your program has access to the standard C++ library. I’m not entirely sure of everything that is in this library, but I do know that it is also necessary for a properly functioning C++ program.

Note: Later down the road, our C++ projects will want to utilize different functions and methods which will require us to use more #include statements. Eventually you will memorize some of the more common includes, but know that you can always look up what types of includes you’ll need online or in API documentation. API documentation (Application Program Interface) is the documentation other programmers have written so that future programs can utilize their code within their project. What this means from a computer science stand-point is: Somebody already wrote code that you need to run your program and your code, and you are now including their code with an include statement.

int main()- This is your main() function. Every C++ program requires that you have at minimum one function, and that function will always be called main(). Later on, we will be utilizing more functions than main(), but do not worry about this for now. All of your code will be contained within main()’s brackets {}.

cout << “Hello World!” << endl;- Code::Blocks will provide this line of code with every new project you make. It has some historical significance within programming. For this you only need to know:

cout- A command to print a statement or variable.

<<- Informs the compiler that information is flowing out to the cout statement.

“”- Anything within the quotation marks is what will be printed with your cout.

endl- This will print the end of a line. Looks like hitting enter in Microsoft Word.

return 0;- Every main() function requires this statement to end the program. The short answer is that main() is expecting you to return an integer value to finalize its function call, so we give it zero to terminate it.

**Your task:** Use commenting to explain what each of these lines do in your own words (the lines that were pre-generated and present when you started up your project). When you’re done, make sure you’ve saved your project. Then click the ‘build and run’ button or press F9 (The button looks like a small yellow gear with a yellow arrow in front of it). Your project should compile correctly and then greet you with a friendly “Hello World!”. Save your project again, then upload your project to Github (Drag your entire project folder into the upload section. This will keep your files nice and tidy for GitHub). Now that you have it saved you can refer back to it for reviewing the basics in the future.