Programming Notes 15: Dev C++ Projects ‘n’ Stuff

As usual, more examples of the notes are in the example program.

1. Files

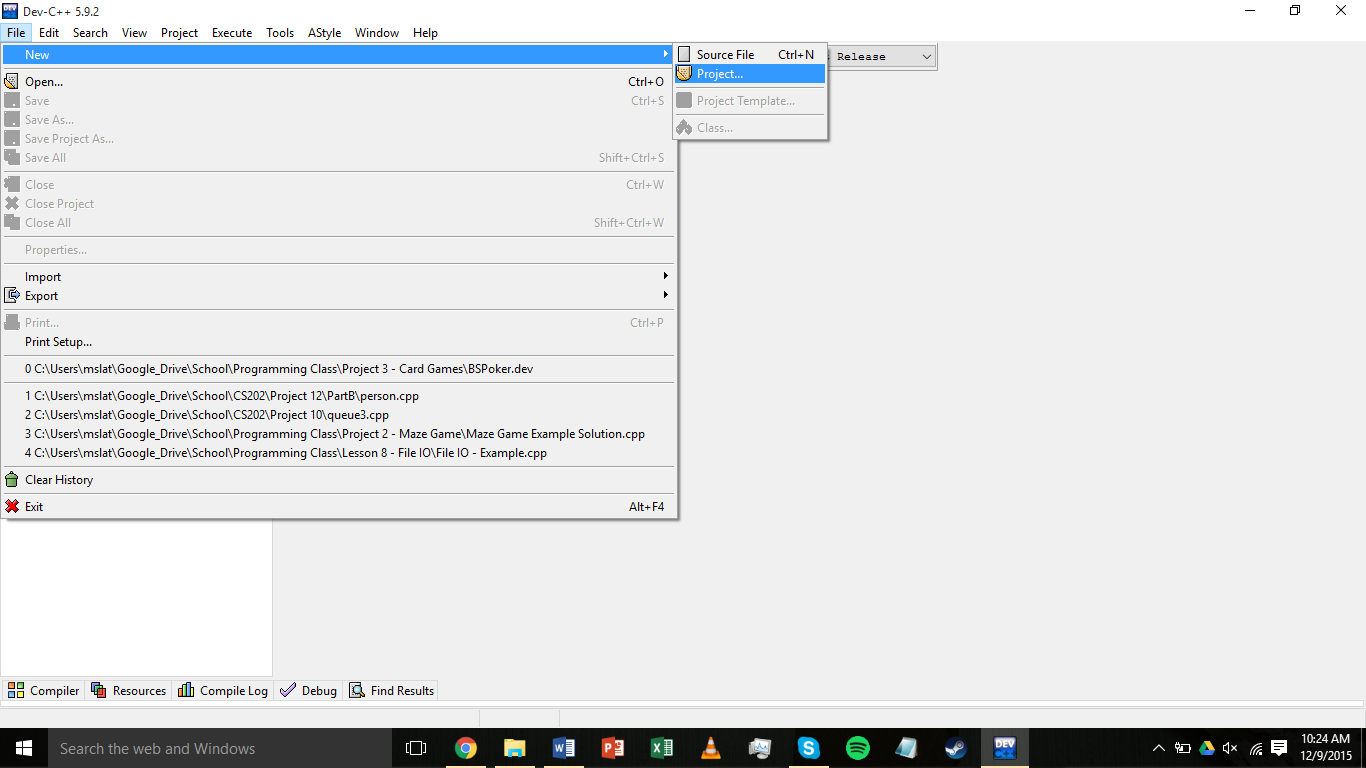
As of yet, we’ve been making all of our classes in the same .cpp file as our main function. However, it is conventional and useful to split classes into multiple files. Generally, the definition of your class (meaning the actual “class name” part) goes in its own .h file, and the class implementation (all your name::func functions) go in their own .cpp file. There are several benefits of doing this: first and most obvious, it allows you to much more easily organize your code. Second, if other people are using your classes—or you, to be honest—you can easily open the .h file of your class to see what and how to use the member functions of your class. Finally, because your classes are in separate files, the compiler only has to compile the code that has changed. This means that the first time you compile your program, it will compile each class (and the main file) individually, and henceforth will only need to recompile specific parts of your program that change. For example, if you change one class, only that one class will need to be recompiled, vastly speeding up compile times. Finally, because your class definition is included in an .h file, simply #include that file wherever you need to use your class. Note that because it’s a file that you’ve created, you should use quotes (“”) instead of pointy brackets (<>). This also means that because you only have to include class definitions where you use them, you may not need to include all of your classes in your main file. For example, if you had a “card” and a “deck” class, the “card” class would obviously need to be included in the “deck” class, but if your main file does not have to create cards, it does not have to be included in your main file.

1. Dev C++ Projects

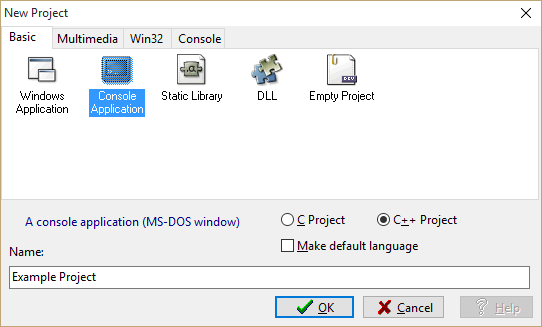
Dev C++ makes it pretty easy to manage projects with many files using their “project” feature.

Here are a bunch of screenshots showing how to set up a project:

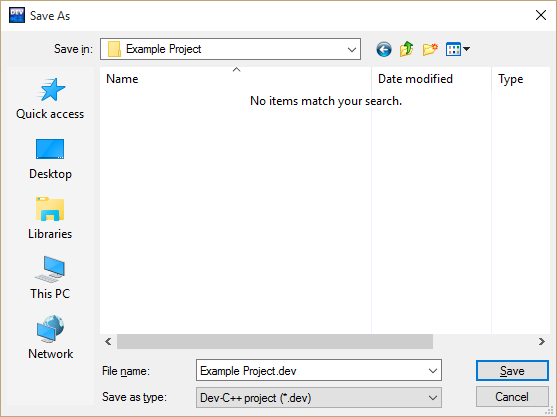
Create a project:



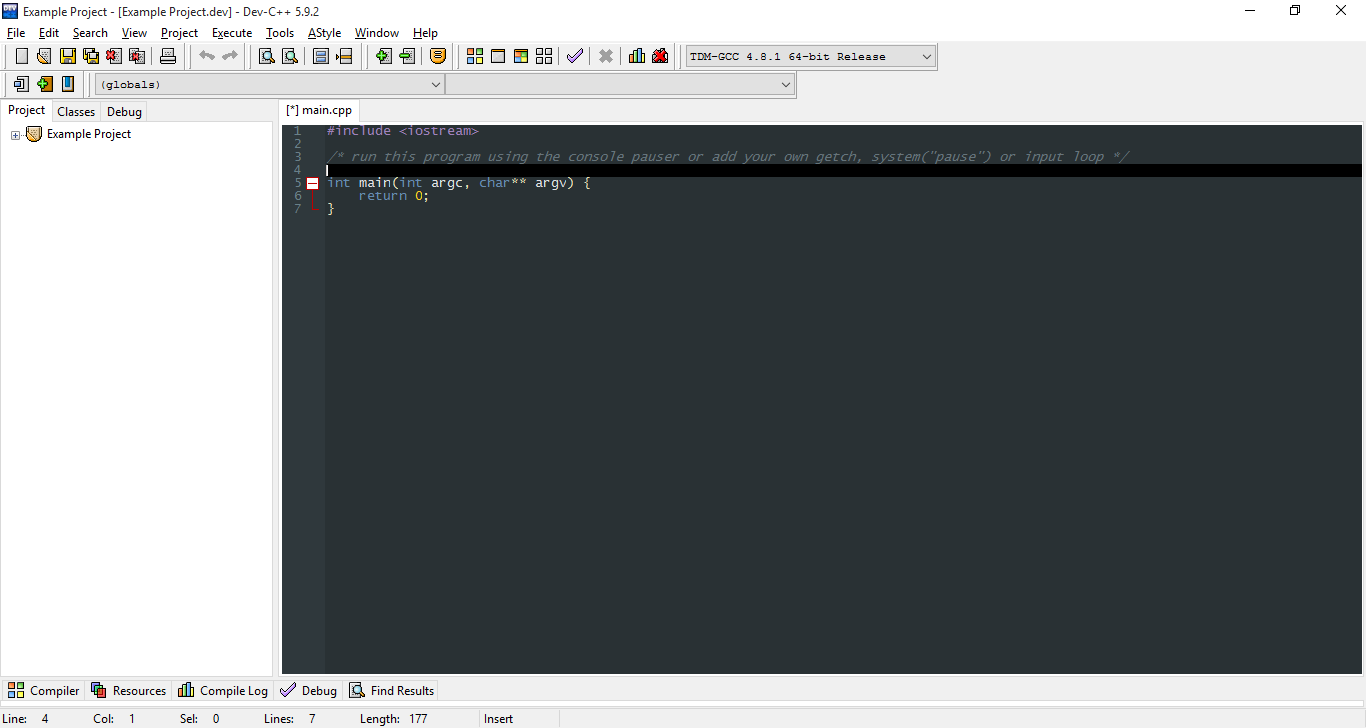
Select “console application,” “C++ project,” and name your project:



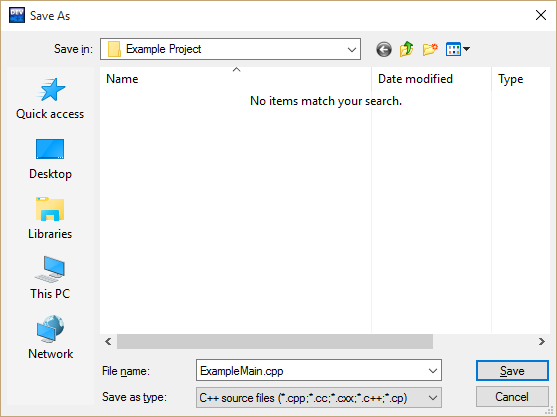
Save your project in a new folder:



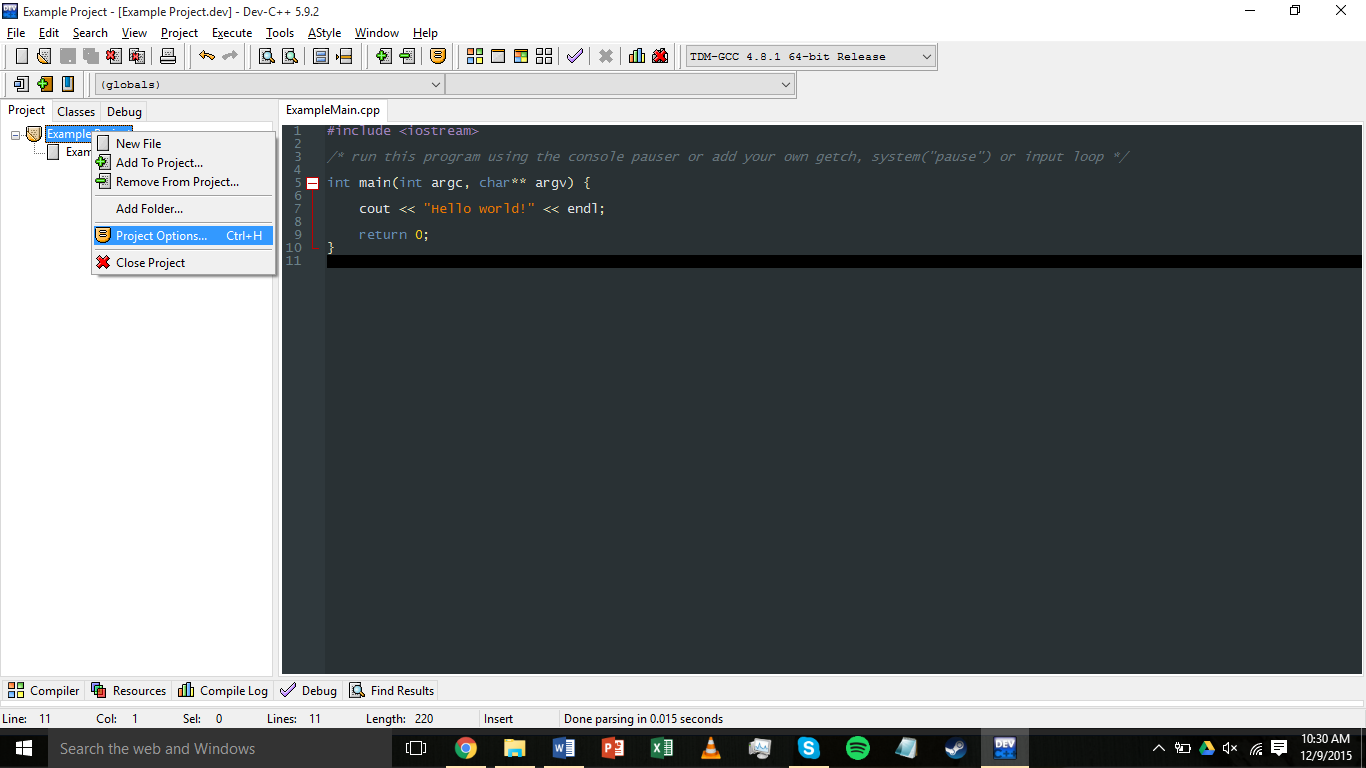
Your project should now look like this:



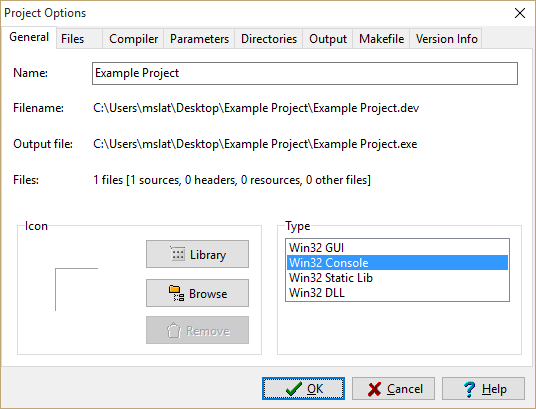
Save the main.cpp file (to the same folder), rename it if you’d like:



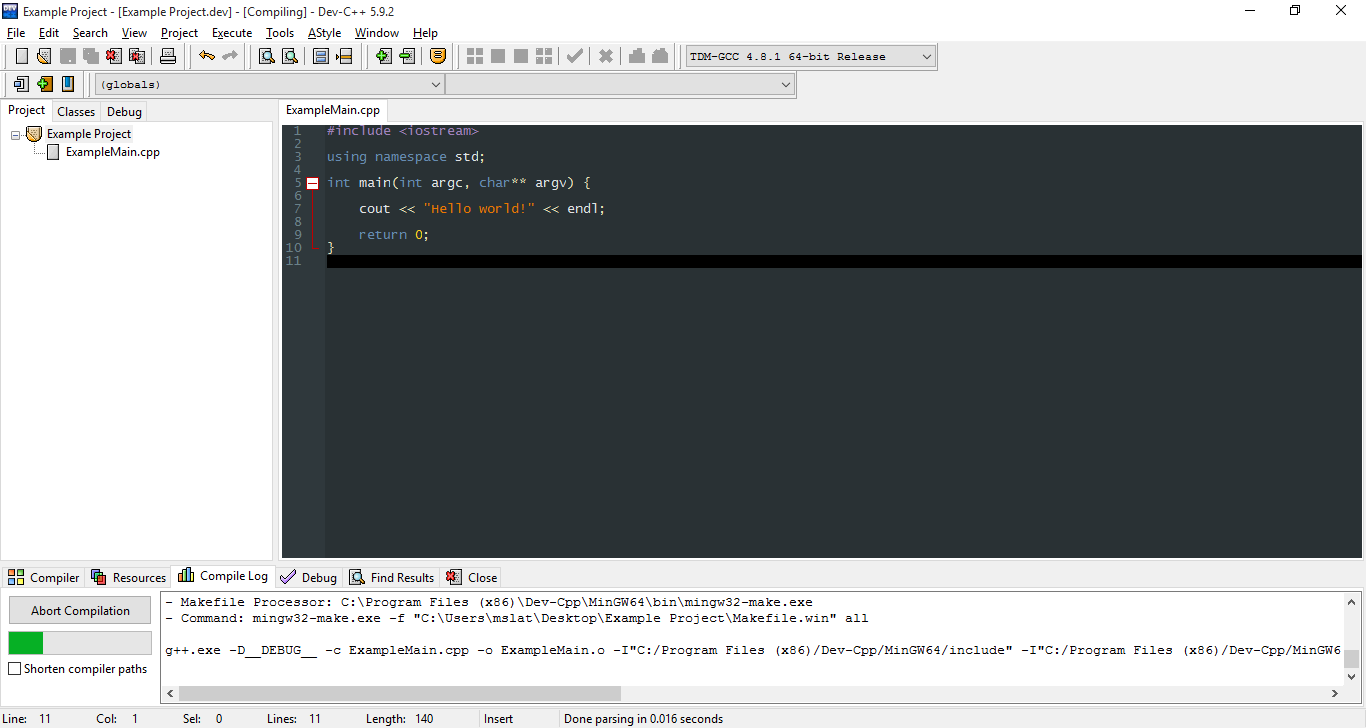
You should now have a working project. To change compiler/linker settings:



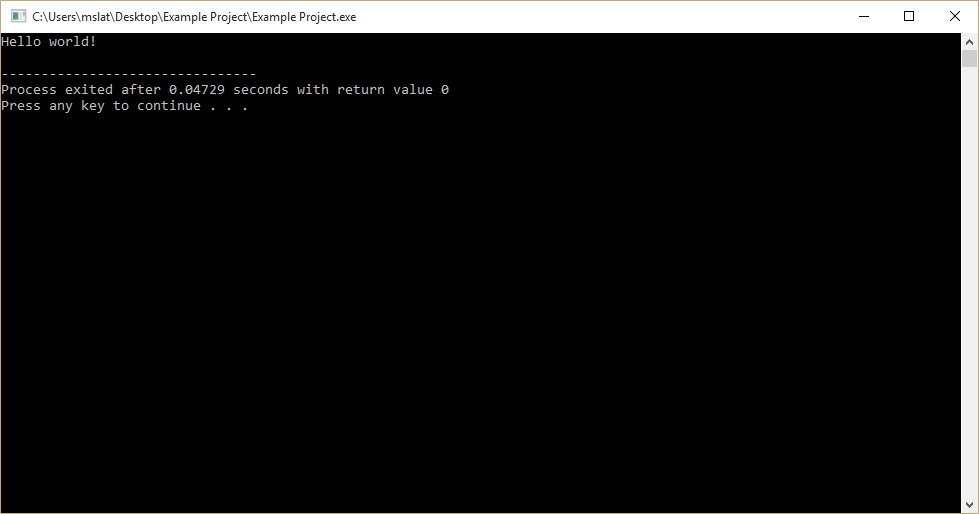
Here you can change project settings like name, type, icon, etc. For now you will only be interested in the “General,” “Output,” and *maybe* the “Compiler” and “Directories” tabs.



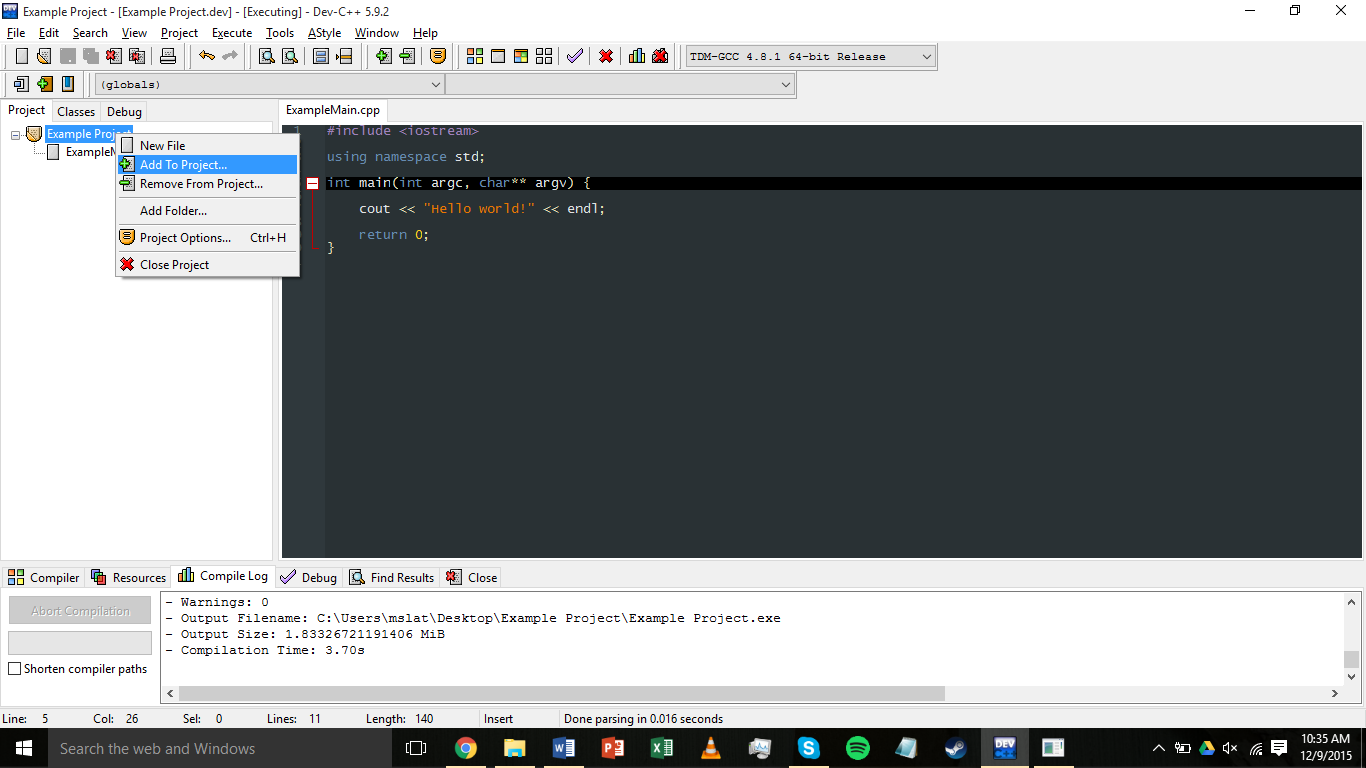
If you are satisfied with your settings, write a “hello world” program, and press F11 to compile and run:



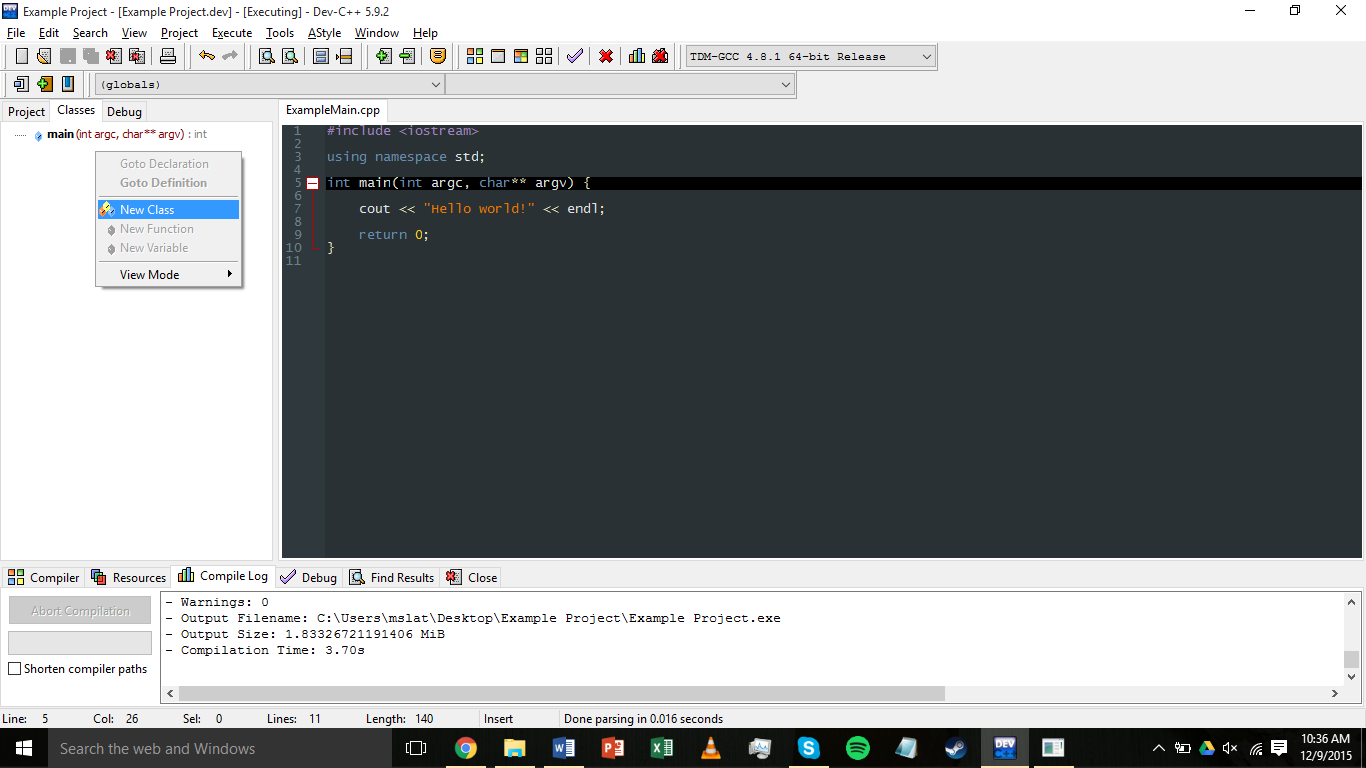
If you set up your project correctly, you should get what you expect.



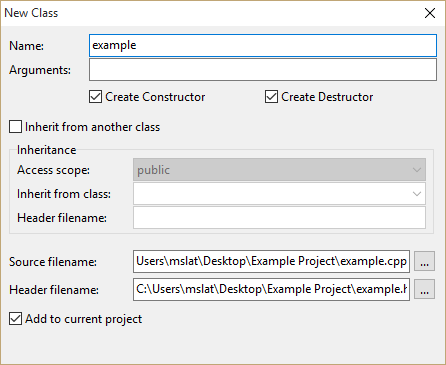
To add non-code files to your project, such as text or data for input:



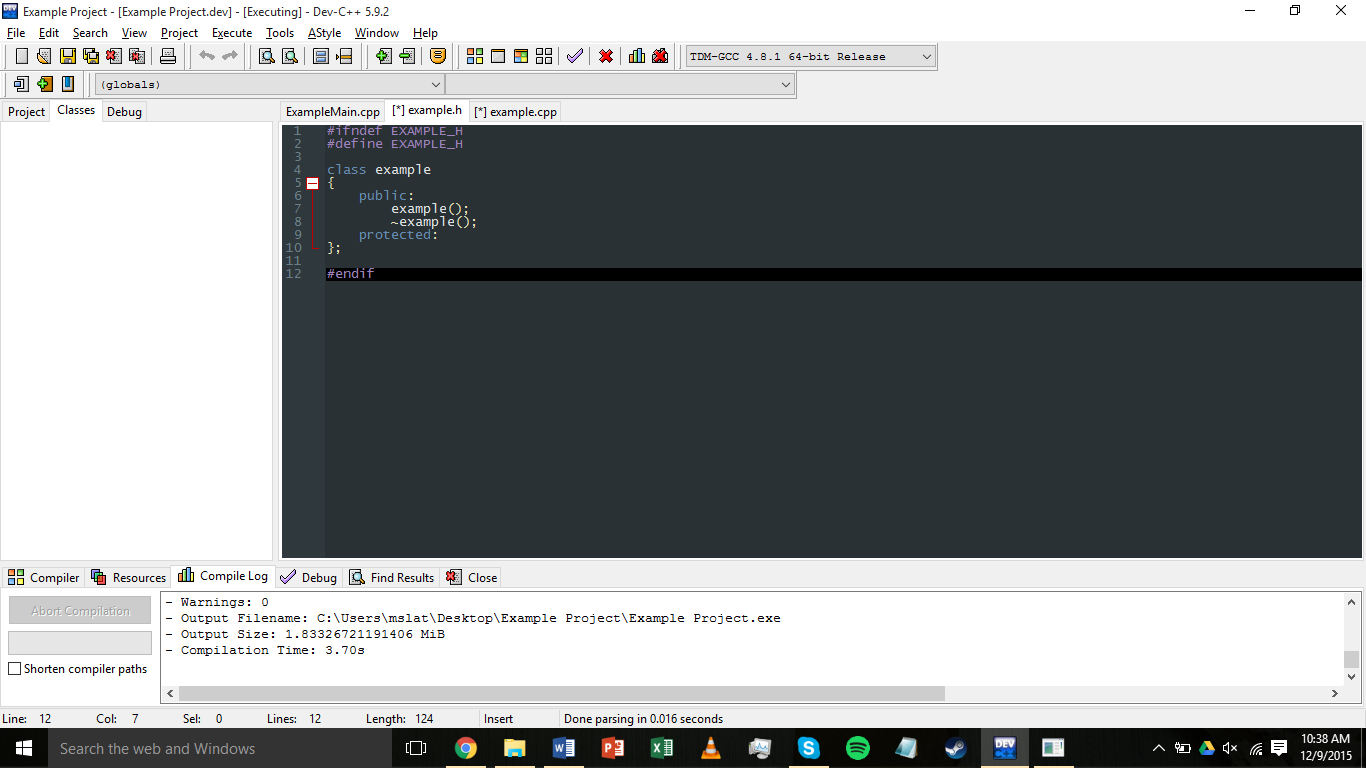
Finally, to add a class:



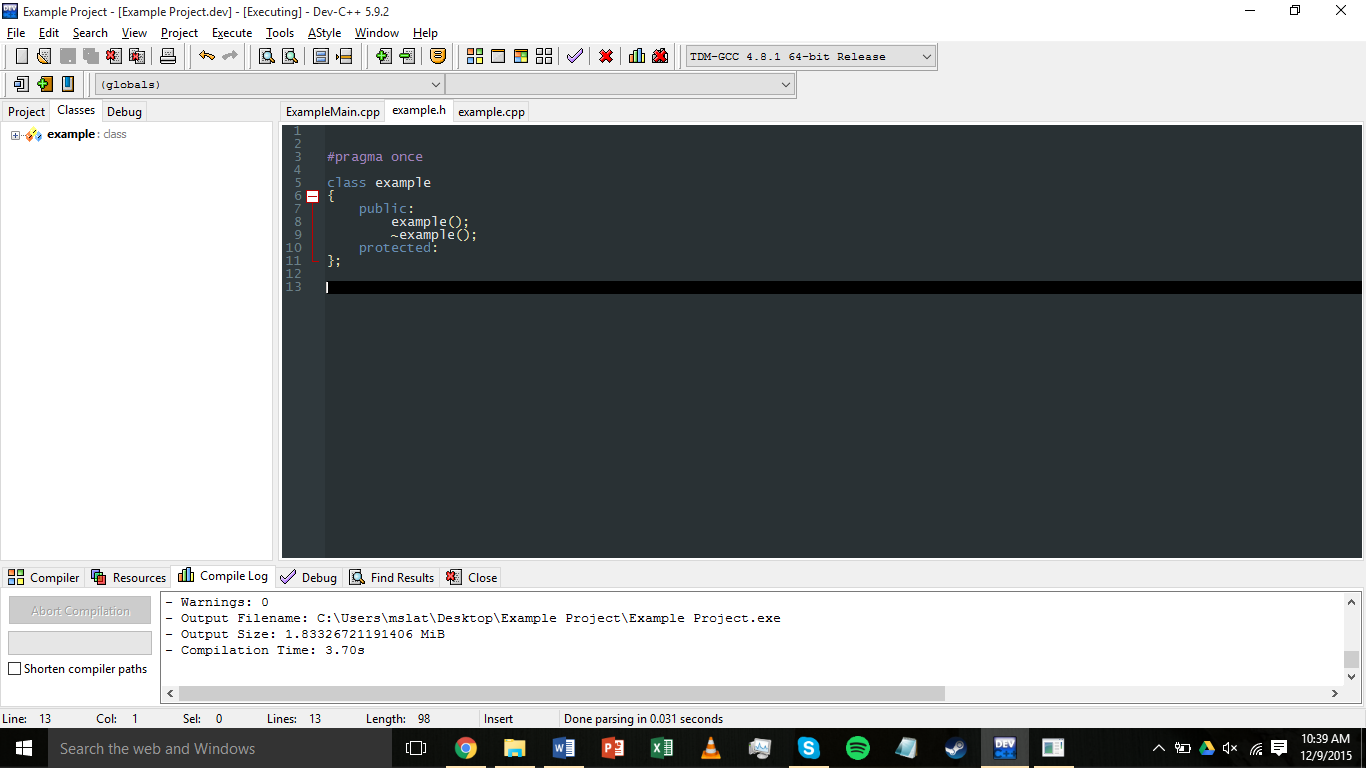
Here, just name your class, and add a default constructor/destructor:



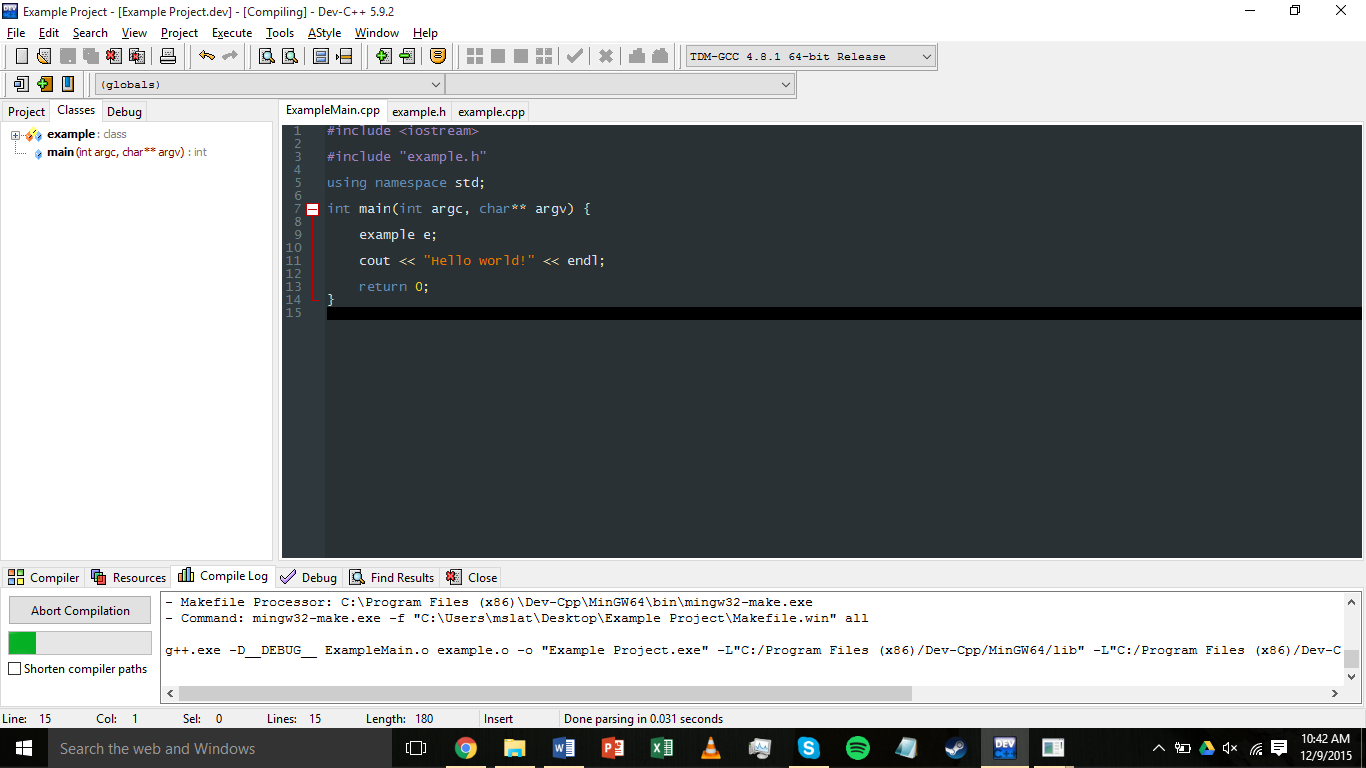
An .h and .cpp file for your class should appear. Save them both to your project folder.



In the .h file, you can replace all of the # statements with “#pragma once” if you’d like. This makes sure that if your .h file would be included more than once, it is only included once:



To use your class in the main file, simply include your new .h file, using quotes. Try compiling this, to make sure everything is working. To add more classes, simply repeat this process.



Finally, you are finished! Remember to save your project settings when you close Dev C++. Your project folder should now have several unfamiliar files. The .dev file is your project, and is what you will open when you want to work on your project. The .layout, .o, and makefile files are all byproducts of compiling with multiple files, so leave them be and you should be good.

