

Lab 0

27 August 2024

In this course, we'll use two systems to develop our programming work. One is a bit easier to get started with (but runs into limitations before too long); the other has a bit more of a learning curve but leads to a much greater degree of control over how your programs will work.

For now, we'll stick to the web-based solution; in your CMSC 161 class you'll start learning about the other, and we'll begin phasing it in here as well in a few weeks.

System 1: Codeboard

Hopefully you've already created an account on `codeboard.io` and submitted your username on the "Sign up for Codeboard" assignment on Canvas, but if not, do that right now.

Also, pull up the textbook and look again at the section you read for today, including the first program.

Today we'll walk through two slightly different ways to work in Codeboard: in projects you create yourself (to try things, to play around, etc) and in projects that are set up as assignments.

First, click the green "Create new" button. You have to give the project a name (something like "Hello" would be fine), and you can skip the description, but you have to change the language. Make sure you choose "C++14" rather than "C++"—it doesn't matter for the first couple assignments but soon we'll be using language features that were not introduced until the 2011 or 2014 updates.

When you click the "Create" button, it takes you to a project page; click "Open in IDE" to go to the editor. (IDE stands for "Integrated Development Environment" and lets us edit and run files in the same window.) In the left pane, click on the file labeled "main.cpp"—the primary file you'll edit in every project for at least the next month or so—to open the editor on that file.

You'll notice that Codeboard has already pre-filled several lines of a program! Click "Compile" and then "Run" to see its output at the bottom of the

window. Then, edit the file so that it more exactly matches the program in listing 2.1 from the book, and compile and run that as well.

Before you move on, click the “Project” menu in the upper left of the window and select “Save changes”. **If you don’t save changes in codeboard you will lose them.** Codeboard does not auto-save your work! Once you’ve saved, then select “Exit project” from the Project menu.

If you’ve already submitted your codeboard login to me, your main codeboard screen should have a project on it called “160 Lab 0: Hello”. This is the form that lab assignments in codeboard will take: on lab day, the assignment will show up here, already available for you to work on. Click on its “Open IDE” button.

Inside it looks similar, but with a little bit less filled in (this will vary from assignment to assignment). The most important difference is that in Codeboard-based assignment files, the word `main` will be replaced with something specific to the assignment (here `lab0_main`).¹

Once again, refer back to the listing in the book to fill in the missing lines here, making it identical to what you see in the book (except leave the name `lab0_main`). Click Compile and Run to verify that it works, and if it doesn’t compile (or runs incorrectly), try to fix it.

Finally, click the green “Submit” button. This is both how the assignment is submitted for grading (Lab 0 is not graded, but real labs will be), and also how you see whether it passed some tests. In this case, if you typed in exactly the program from the book, the test will fail this time—use its feedback about “Expected output” to revise your program, then Compile and Run to verify the change, then Submit again. You can submit as many times as you want or need, and you’ll get the test feedback every time.

Once you’ve submitted at least once, if you’re stuck on getting it to work perfectly, flag me down for some help!

...now move on...

Hop back to the book and look at Section 2.3. It shows a program that is essentially the C++ version of what we wrote in class yesterday. Create a new

¹This is a mild departure from the C++ standard—normally a program *will not run* unless there is a `main`, specifically, but this is an adaptation so that Codeboard will be able to auto-test your program on assignments that I personally have provided you. If you write a program in any other context, it’ll need to be `main`.

Codeboard project and in it, type in the code from Fragment 2.2. *In general* you should be trying out code that you see in the book (or elsewhere!) to try it on for size and see what it does, and I won't usually dedicate class time to it but I want to get us started.

Once you've typed it in, try it out. Are you sure it works? How do you know?

If there are things you're wondering about and I'm busy with another student, feel free to talk to the other students about it yourself, or else write down a note to yourself—we'll talk about this in class tomorrow.

...and then break it

Once you've got Fragment 2.2 typed in and working correctly (and saved!), I want you to try *intentionally breaking it* in different ways. Try a little change, see if it gets a compiler error. Or makes the wrong output.

Each time you break it, write down (in a notebook, probably, or possibly a text file) a) what you did to break it, and b) what the error message was (maybe just a few salient words from the error).

How many distinct error messages can you trigger with relatively minor changes to the code?

Handing in

Other than clicking submit on the Hello World program, there is no further handin this week. (Most labs will have continuing work after the 75 minute lab period ends, but we're not ready for that yet.)

This document was written and prepared without the use of generative AI.