

# 1 Introduction

This is simply to get you up and running with C++. You may use any IDE you wish in this class, I would suggest either CodeLite, Eclipse, or Code::Blocks. There are instructions on the MyClasses page for this class on installing and configuring these for your computer. All three of these are available on the Windows machines in HS 150. Each had advantages and functionality that the other do not. I would suggest trying one of them and if you do not like the interface give another one a try. This lab will go through a couple processes you will be doing often during the course.

Every IDE uses a different folder/directory structure on where the code and data files are expected to be and where to find them. When you submit your work you will be handing in a zip file that contains all of the code and data files needed to run your program. If any of these are missing I will be deducting points. So for whichever IDE you use make sure that you know where these are. I would suggest bring the files up in a text editor (Notepad, Notepad++, ...) before you compress them and make sure they are correct.

You do not need to do these exercises for each of the three IDEs just select one of them. If you wish to experiment with another one later you can go through the process for it on your own.

As always, back up your work in several places. You can use your P drive, cloud storage, thumb drive, .... You may bring your laptop to lab and use it instead of the lab computers.

## 2 Creating a New Project

In this section we are just going to create the standard Hello World project, compile it, and run it.

### 2.1 CodeLite

1. Start up CodeLite.
2. Select New Workspace, if a workspace has already been loaded select File > Close Workspace to close it. A workspace in CodeLite is like a workspace in Eclipse, it contains all of your projects you create. You can create as many workspaces as you would like and can easily load in different ones.
3. Select a C++ type.
4. Select a path and workspace name. The path will probably default to the local hard drive. If you store the workspace on the local drive it will not be accessible from any other place than that particular machine. I would suggest that if you are using the HS 150 machines to do your work and not your own laptop that you change this to a location on your P drive.

5. At this point the workspace will be loaded, and empty. On the left there should be a Workspace tab that is selected, the name of the workspace will be listed below, and there will be nothing under it. When you add in projects, they will be listed below the workspace name.
6. Now we will create a new project. In CodeLite you should create a new project for each new program. Select File > New > New Project from the main menu. You can also right-click on the workspace name and select New > New Project from the popup menu.
7. Here you fill in some info that the IDE needs to compile your program.
  - Path is the path to the workspace and that should not be changed.
  - Name is the name of the project. This can be whatever you want but needs to be unique, that is, different from other project names. I would suggest that you use a single name with no spaces or special characters, it is better for your OS. HelloWorld would work well for this program.
  - Category will always be Console for this course.
  - Type will always be Single executable (g++) on Windows and Linux, and Single executable (clang) on the Mac.
  - Compiler is MinGW (C:\MinGW\bin) on Windows, GCC on Linux, and either GCC or clang on the Mac.
  - Debugger is GNU gdb debugger.
  - Build System is CodeLite Makefile Generator
8. Click OK. Note that for the next project you create these options will be loaded in automatically so all you will need to do is give the project a name.
9. At this point the new project will appear under the workspace at the left. You will see that it is also bold and italics. When a project is bold and italics that means that it is selected, so when you compile it and run it that is the program that will be run. You can have different files from different workspaces in the editor at one time but it is the selected project that is run. You can select a different project by double-clicking the project name.
10. Double-click the project name and you will see a subfolder named src (for source) and then double-click the src folder. This will then show the file main.cpp. For a little while in this class all of our code will be in a single file. Later we will have several separate files that work together, we will discuss that when it is time. If the file does not get loaded into the editing area automatically, double click the main.cpp. At this point you should see the file's contents. Note also that in the tab it displays both the project name and the filename. This comes in handy if you have multiple projects open.

11. The program that is automatically generated is the usual Hello World program. CodeLite actually writes this in C and not C++. If you compile a C program in C++ it will work fine but we want to use C++ functionality. Edit the code to look like the following.

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     cout << "Hello world!" << endl;
8     return 0;
9 }
```

12. Now we are ready to compile. First from the main menu select View and make sure both the navigation bar and tool bar are selected. If not click on those to make them visible. Both of these are handy to have. Compile the program by clicking the hammer icon (or from the main menu Build > Build Project, or use the F7 shortcut). The Build window at the bottom will show the details of the build and list any errors and warnings you have.
13. Run the program by selecting the run icon on the toolbar, or Build > Run from the main menu. At this point the console window will appear with the output of the program. You can shut the console down by pressing any key on most systems, some systems require the enter key to be pressed.

Some notes about the CodeLite IDE:

- You can adjust the shortcuts to be whatever you want to make the interface easier to use. Select Settings > Keyboard Shortcuts to make any changes there.
- You can also do some customization to the toolbar by clicking the small down arrow on the far right of the toolbar and selecting Customize. I would suggest that you add in the Format Source tool at the bottom of the list. This icon will look like an indent icon and it will automatically format your code indentation to match AStyle formatting, which is a requirement for all of the code you submit to me.

## 2.2 Eclipse

The C++ version of the Eclipse IDE is very similar to the Java version of the Eclipse IDE, but there are some differences.

1. Start up Eclipse for C++.
2. As with Java, the workspace needs to be selected at startup. In the lab this is (should be) set to a generic workspace on your P drive, good place for it since it will store your work on the network drive instead of the local drive. If you do get the Select a directory as workspace prompt navigate to your P drive and create a folder for the workspace.

3. The generic start window may appear which you can close. Then you will see the IDE layout, which is very similar to the Java one.
4. Now we will create a new project. As with Java you will want to create a new project for each new program. From the main menu select File > New > C/C++ Project. You can also select the drop down arrow on the new tool bar icon and then C/C++ Project.
5. Select C++ Managed Build, click Next.
6. Give the project a name, it must be different than any other project name in the workspace and should not contain spaces or special characters. For this example, HelloWorld would be good. Select Hello World C++ Project in the Project type and in the Toolchains window select MinGW GCC on Windows, GCC on Linux, and clang on the Mac. Click Finish.
7. At this point the new project will appear in the Project Explorer on the left. As with Java the icon is an open folder, also as with Java the project that is highlighted is the current project that will be compiled and run. The HelloWorld.cpp file is created and loaded automatically. If you double-click the HelloWorld in the project explorer you will see Includes and src (for source) and then double-click src to see the files in the project, just HelloWorld.cpp for now. For a little while in this class all of our code will be in a single file. Later we will have several separate files that work together, we will discuss that when it is time. If the file does not get loaded into the editing area automatically, double click the HelloWorld.cpp. At this point you should see the file's contents.
8. The program that is automatically generated is the usual Hello World program. We will make some minor alterations to this. Edit the program to look like the following.

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     cout << "Hello world!" << endl;
8     return 0;
9 }
```

9. Now we are ready to compile. To compile you use the hammer icon, or Project > Build Project from the main menu. You will note that there are two hammer icons. The one on the far left is part of the launch bar. I usually hide the launch bar since it is the other icon you want. We will hide the launch bar at the end of this. Click the hammer icon that is further to the right. The console window at the bottom will display the details of the compile as well as errors and warnings if you have any.
10. Run the program by selecting the run icon on the toolbar, or Run > Run from the main menu. The run icon looks like the run icon from the Java IDE, green circle with white arrow. Again, there are two of them, the one on the far left is part of the launch

bar and not the best one to use, use the one further to the right. At this point it will ask you if this is a C/C++ Container Application or a Local C/C++ Application, we will always be selecting Local C/C++ Application here. Select it and proceed. At this point the program will be run and the output will be in the Console window at the bottom of the IDE.

Some notes about the Eclipse IDE:

- You can adjust the shortcuts to be whatever you want to make the interface easier to use. Select Window > Preferences > General > Keys to make any changes there.
- You can also do some customization to the toolbars if you would like.
- To get rid of the launch bar, which I would suggest you do, Select Window > Preferences, in the search in the top left type in launch, select Launch Bar, on the right deselect both the Enable the Launch Bar and Enable the Build Button. Then click Apply and Close.
- As with the Java IDE, Shift+Ctrl+F will automatically format your code indentation to match AStyle formatting, which is a requirement for all of the code you submit to me.
- The build and run process in Eclipse is a little strange. The first time you compile a program you must use the hammer icon. After that, the run icon will both compile and run the program.

## 2.3 Code::Blocks

This is for Windows and Linux users. There is a version for the Mac but it is nearly 10 years old and has not been maintained, I do not recommend that you use it.

For Code::Blocks, you might get a welcome screen, if so you can close it. You should see the general layout for Code::Blocks at this point. Management window on the left and other tabbed windows at the bottom. The way workspaces work in Code::Blocks is a bit different than it is in CodeLite or Eclipse. Here there is one generic workspace that you load projects into. You can have multiple workspaces and set this up more like CodeLite and Eclipse but frankly it works better if you simply load a project or two in during the run of the program instead of using multiple workspaces. Feel free to experiment.

Also, if you installed this on Windows (using my instructions) it automatically installed MinGW in a different location, as a subfolder to the Code::Blocks system.

1. Create a new project. In general, every program should have its own project. Select File > New > Project from the main menu or click on the new icon in the toolbar and select Project from the popup menu that appears.
2. In the list of icons select Console application (we will always use this one). Click Go.

3. There will be an information wizard that can be skipped.
4. Select C++ and click Next.
5. Give the project a title, HelloWorld would work here. You can change the folder for the project to be anywhere on the computer or network drive, might want to use your P drive if you are working on a lab machine. Keep this folder in mind since you will need to go there to get the code files when you submit them. Click Next.
6. The compiler should be defaulted to GNU GCC Compiler, this is MinGW, leave it as it is. Keep the options below that at the defaults that are listed. Click Finish.
7. At this point you will see HelloWorld in the workspace. The title is bold which means that it is the currently selected project. Below that is a Sources folder. Double-click the Sources and you will see main.cpp below this. Double-click the main.cpp file to open it up in the editing area. It should look something like the following.

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     cout << "Hello world!" << endl;
8     return 0;
9 }
```

8. To compile the program click the compile toolbar button, yellow gear icon. You can also select Build > Build from the main menu. The details of the build will be output in the Build log at the bottom. Errors and warnings will show up there.
9. To run the program click the run tool, green arrow, or Build > Run from the main menu. At this point the console window will appear with the output of the program. Hit any key to close the window.
10. There is a tool that does both compile and run, the yellow gear and green arrow combined.

Some notes about the Code::Blocks IDE:

- You can adjust the shortcuts to be whatever you want to make the interface easier to use. Select Settings > Editor > Keyboard Shortcuts to make any changes there.
- You can also do some customization to the toolbars if you would like. At the start there are a lot of toolbars that are loaded. You can turn some of these off by View > Toolbars > and then unchecking the ones you do not need. Personally, all I really use most of the time is Compiler and Main.
- There is a source code formatter to automatically format your code into AStyle format, a requirement for all submitted code. It is under Plugins > Source code formatter (AStyle) in the main menu. You will probably want to create a shortcut to it.

## 3 Submitting Your Work for Labs & Homeworks

All code for homeworks and labs will be submitted through MyClasses. since each IDE stores files in different locations it is important to know where they are.

### 3.1 CodeLite

In CodeLite if you enabled the Navigation Bar it is easy to find your files since there is a menu option for that.

1. In the Navigation Bar you should see a box with the project name and main.cpp. Click in the box and a small submenu will appear. At the bottom is the option to Open Containing Folder, select it.
2. At this point your file manager/explorer will open in the correct folder.
3. All the .cpp (and later in the class .h) files are those created by CodeLite. The other files are not code files and I do not need them when grading your work. So if the MyClasses submission is to be just code files then you can drag the .cpp and .h files to MyClasses. If I ask for a single zip file submission then you will copy all the .cpp and .h files to another folder and zip up the entire folder to be uploaded to MyClasses.

### 3.2 Eclipse

Eclipse has a very nice feature that I wish both CodeLite and Code::Blocks had, the Project Explorer also works like a file manager. So you can just click and drag files to and from the Project Explorer.

1. Open your file manager/explorer.
2. If I ask for submissions to be just code files you can click and drag the .cpp and later .h files from the Project Explorer to MyClasses. If I ask for submissions to be a single zip file then create a new folder named something like Lab04 or Homework05, etc. then click and drag all the files from the Project Explorer to that new folder, finally zip the new folder and upload it to MyClasses.

### 3.3 Code::Blocks

1. Open your file manager/explorer.
2. Create a new folder named something like Lab04 or Homework05, etc. to store the code files. As with other IDEs there are files we do not need that are stored with our code files.

3. In the Projects tab in Code::Blocks right click on the project name, then select Open Project Folder in File Browser. This is not the system file browser but one that is semi-functional inside of Code::Blocks. Here you will see the .cpp and .h files for the project. Select all the .cpp and all the .h files, do not select the others. Right-click and select Copy to from the menu. At this point the file dialog will appear, navigate to the new folder you created and click the Select Folder button. This will copy the selected files to the new folder.
4. If I ask for submissions to be just code files you can click and drag the .cpp and later .h files from this new folder to MyClasses. If I ask for submissions to be a single zip file then you would zip the new folder and upload it to MyClasses.

**Note:** Do not try to drag the code files from the Code::Blocks File Browser to the new folder, the program crashes.

## 4 Running the Example Code

On the MyClasses site there is a zip file of example code. The example code will be the code examples I use and manipulate in class. You will want to run and experiment with these during the semester. To do so, the basic idea is to create a new project in whichever IDE you are using and copy the code files to the project so that it can be compiled and run. When we have multiple code files and/or data files this is a little more involved and we will discuss the processes when the time comes. For now here is the easiest way to do this which works on each IDE.

1. Create a new project in your IDE.
2. Open the main code file produced by the project creation. Hello World of some type. Delete the contents of the file.
3. Open the example code file in an external text editor, for example, NotePad, NotePad++, Kate, Text Wrangler, etc.
4. Select all the code in this external text editor. Usually Ctrl+A or something like it will do the job.
5. Copy this to the clipboard, Ctrl+C.
6. Go back to the empty file in the IDE.
7. Paste the code in, Ctrl+V.
8. Save the IDE file, compile and run.

**Note:** Do not try to drag the code into the IDE, this might not work the way you think it should.



## 5 Submit Your Work

If you have not already done this use the instructions above to submit just the main .cpp file to the MyClasses system.