

# Integrated Development Environment for C/C++

CS221  
Spring 2025

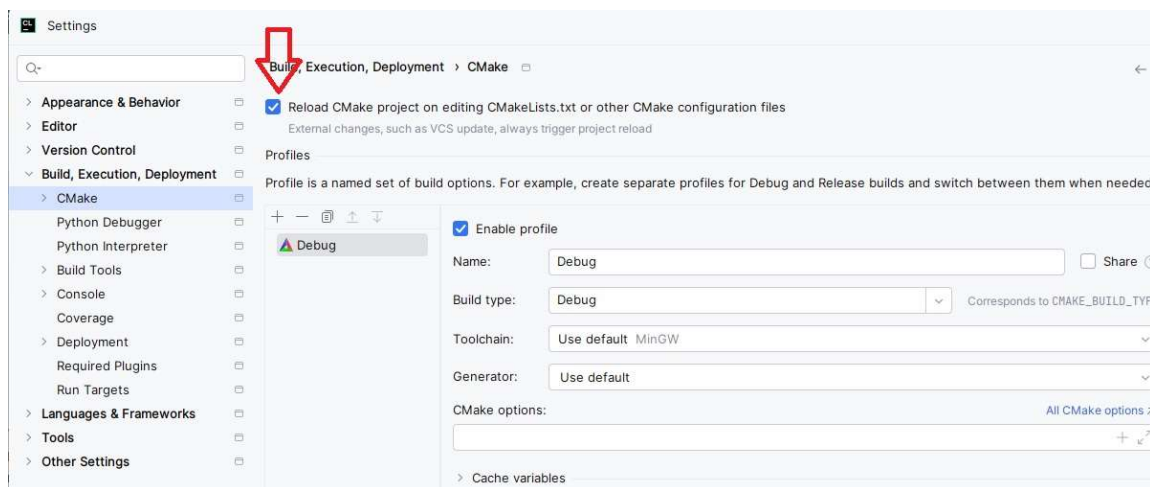
"Eclipse for C/C++" does not work properly on Mac computers. The only IDE I've found that works on both Mac and Windows is the "CLion" IDE, which comes from JetBrains, the same company as makes PyCharm and IntelliJ. The CLion look and feel is very similar to those two other IDEs.

You can install a free version from CLion as long as you are a student. Browse to <https://www.jetbrains.com/community/education/#students> and set up an account. Then browse to <https://www.jetbrains.com/ides/> to install CLion. You will need your "student license" to activate CLion.

Mac users must get the proper downloader for their CPU architecture—either "Apple Silicon" or "Intel". You can run the `uname -a` command in a terminal window if you are not sure which CPU type you have.

## CLion setup

- Go to the "File > New Project Setup > Settings for new projects" menu.
- Go to Build, Execution, Deployment > CMake. Check the 'Reload CMake project on editing...' checkbox and click 'OK'.



- You can set up the CLion appearance the way you like also

## How to set up CLion with our CS221 directory structure

- Your **CS221/MyWork** directory will be the top-level GitHub directory for your classwork.
- Each subdirectory inside MyWork e.g. **Lab15** will be a CLion "project".

In other words, the directory structure you already have for your **CS221/MyWork** directory should be fine. GitHub should work with no setup required.

## How does CLion build C projects?

By default, CLion uses **cmake** rather than **make** to build projects. **cmake** is simpler, less common, and less powerful than **make**. For CS221 I recommend you:

- Have CLion manage project building with **cmake**
- If you want to write a **Makefile** in the same project directory so you can build from the command line, that is fine and it does not interfere with CLion

Though it is possible to have CLion use **make**, it is a bit annoying and difficult.

## How to create a new project in CLion

Go to the "File > New > Project" menu.

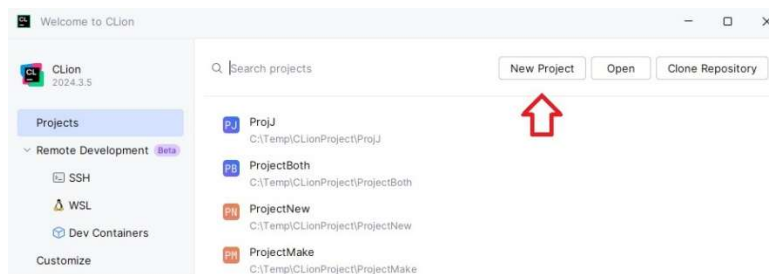
Select "C Executable", enter the proper Location for your project, click 'Create'.

You can add \*.c and \*.h files by right-clicking the Project > New > "C/C++ source file" or "C/C++ header file". Enter the filename and type ("\*.c" or "\*.h") and click 'OK'. If you are asked to "add to targets" your new file, that will add the new file to your existing executable when you build.

## How to open an existing project in CLion

If your project has a **Makefile**, CLion will attempt to set up a project managed by 'make', and it will mess things up. Overall the whole "project import" process is pretty flaky. This recipe worked for me:

- In a terminal window, temporarily rename your **Makefile** to something safe like **NotAMakefile**
- In CLion, close any existing project, via the "File>Close Project" menu
- In the resulting window, click 'New Project'. (Don't select 'Open'.)



- Select 'C Executable', enter the location of your existing project, and click 'Create'
- You now should have a project with your existing source files and also a **CMakeLists.txt** file (which is like the **Makefile** for 'make')

Now you can rename your **NotAMakefile** file to **Makefile**, for building in a terminal window.

## CMakeLists.txt files

You can edit your projects' **CMakeLists.txt** files, just like you can edit **Makefiles**. Here is a sample file (only the **red** is actually in the file, the **gray** is just my comments):

```
cmake_minimum_required(VERSION 3.30) ← don't change this
project(Project03 C) ← "Project03" should be the name of your project

set(CMAKE_C_STANDARD 11) ← don't change this
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -g -Wall") ← put this in if needed

add_executable(runfifo runfifo.c fifo.c) ← The "rule" to build runfifo.exe
add_executable(testfifo testfifo.c) ← The "rule" to build testfifo.exe
target_link_libraries(testfifo PRIVATE m) ← Add "-lm" to linking rule
```

You need an "add\_executable()" line for every executable in your project. The first name in parentheses is the name of the executable, and the rest of the names are the C source files that are compiled to make that executable.

You need a "target\_link\_libraries()" line to specify libraries to be linked, for every executable that needs libraries.

## Run/Debug configurations

CLion might have already defined several Run/Debug configurations corresponding to your executables. Sometimes, you may need to create your own configuration or to edit an existing one. To do this, on the titlebar's pulldown with configurations, pull down the menu and select 'Edit Configurations'

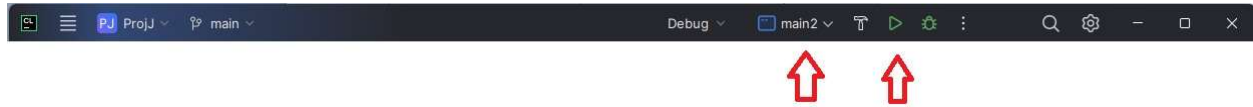


If you already have a configuration for the executable you care about, click the three dots after each configuration and then select 'Edit'. If not, click the '+' to create a new configuration, and then edit it. For the configuration, I recommend:

- Give it some useful name
- For the Target, select your executable
- For the Working directory, select your project directory
- For the Executable, browse and select your Executable if it exists already. If not, then just type in the name.
- Set any program command-line arguments you want
- Select 'Emulate Terminal in the output console'. This is needed so the Console window can send keyboard input to your program
- When debugging, you may need to select 'Run in external console'
- 'Before launch' should say 'Build'
- Now click 'OK'

## How to run a project

With the project open in CLion, select the desired executable to debug in the pulldown menu in the titlebar, and click the "run" icon.



## Command-line arguments in CLion

In the titlebar pull-down with the executable filenames, select 'Edit Configurations'. There you can select your executable and can enter command-line options.

## Keyboard Input in CLion

For CLion to pass keyboard input from the Console window to your program, you must:

- in the titlebar pull-down with the executable filenames, select 'Edit Configurations'. Select your executable. Click the 'Emulate terminal in the output console' checkbox and click 'OK'
- when debugging, you might have to select 'Run in external console'. This starts a separate window for keyboard input and terminal output.


## How to debug a project

With the project open in CLion, select the desired executable to debug in the pulldown menu in the titlebar, and click the "bug" icon.



This opens the "Debug" window, probably on the bottom left of the screen. Here are controls to "step-over", "step-into", "step-out" and "continue" your debugging. There is also a window to show you all variable values.

## GitHub and CLion

There is a "Commit" window you can open (look for the  icon). Here you can select files you want to commit, enter a "commit" message, and select "commit and push".

CLion generates a lot of configuration files. To keep these from messing up your GitHub repo, please get the latest **.gitignore** file from **CourseInfo/Day01** and copy it to your **CS221/MyWork** directory.