Lab 1

Outline

- **Background Info on Clion**
- **Install Clion**
- **Clion Basic Functionalities**
  - Create projects
  - Add files to a project
  - Compile and run C programs
  - In class Exercise
- **Debug with CLion**

Background Info on CLion

- Popular **Integrated Development Environment (IDE)**
  - Provides graphical tools for coding, building, running and debugging applications

Background Info on CLion

- Popular **Integrated Development Environment (IDE)**
  - Provides graphical tools for coding, building, running and debugging applications
- Free and Open Source cross-platform IDE that supports multiple compilers including GCC and CLang
  - It has been developed for Windows, Linux and MacOS

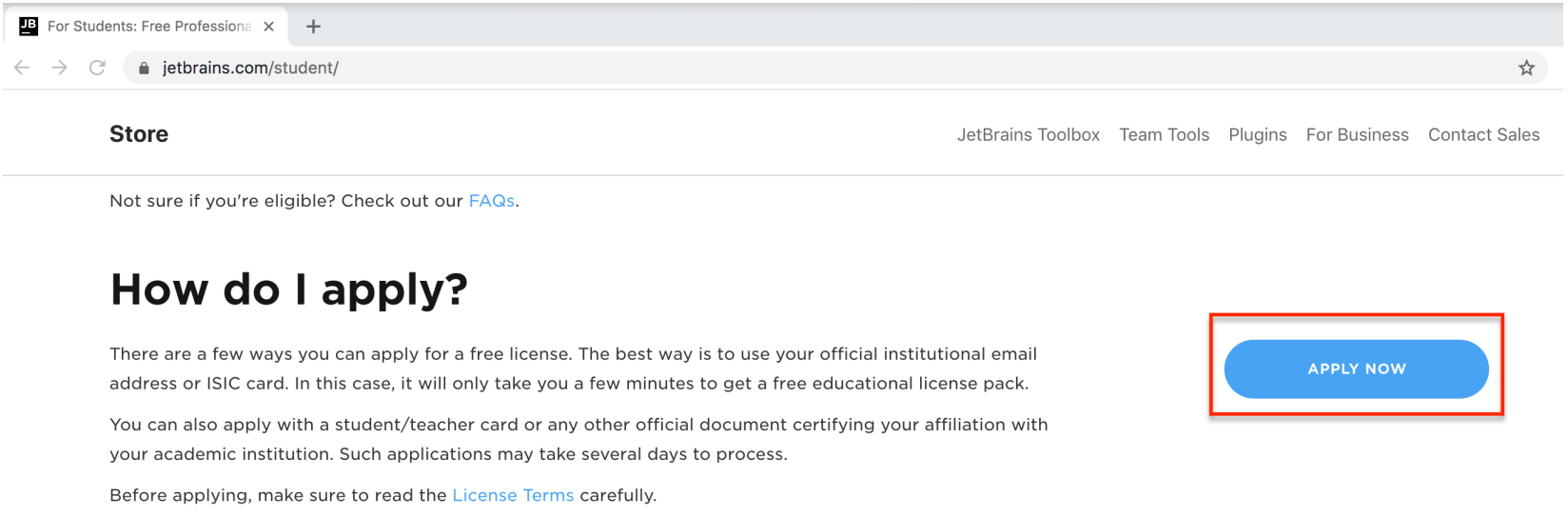
Background Info on CLion

- Popular **Integrated Development Environment (IDE)**
  - Provides graphical tools for coding, building, running and debugging applications
- Free and Open Source cross-platform IDE that supports multiple compilers including GCC and CLang:
  - It has been developed for Windows, Linux and MacOS
- It is a general-purpose open platform that facilitates and encourages the development of third-party plug-ins.

Install CLion

Obtain CLion

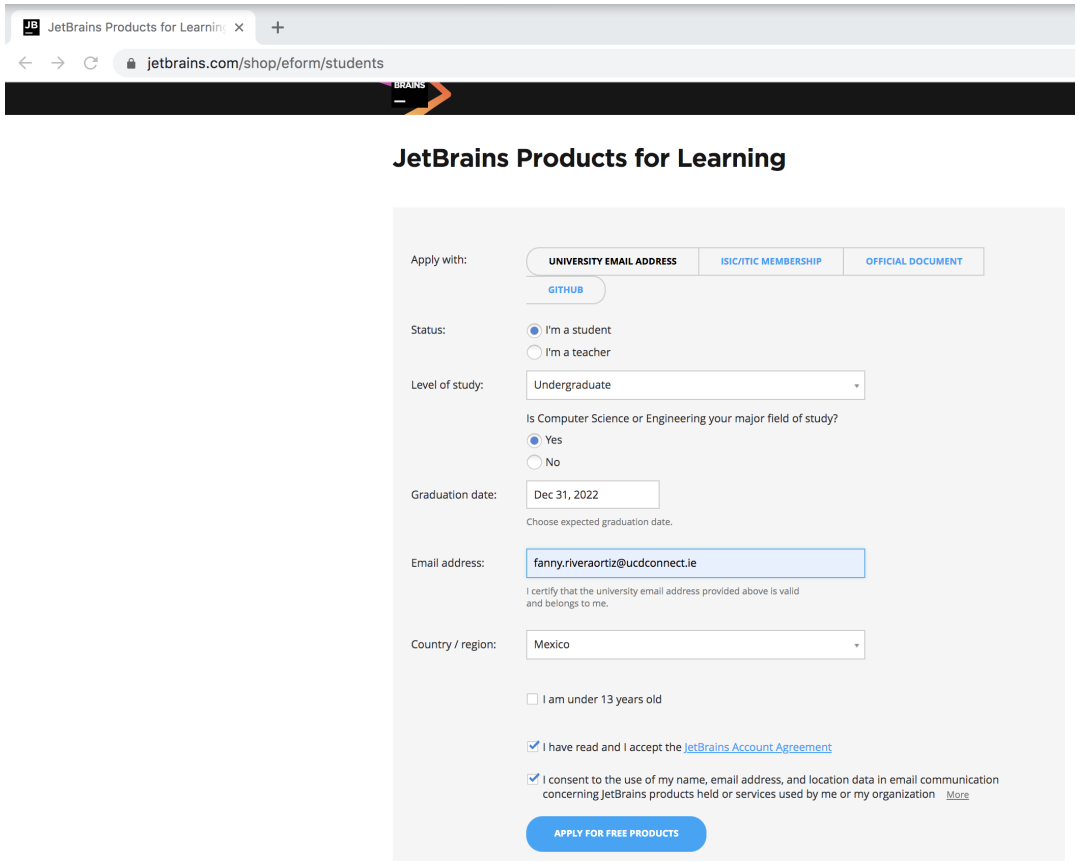
- Apply for a student pack using your institutional email at: https://www.jetbrains.com/student/



Obtain CLion

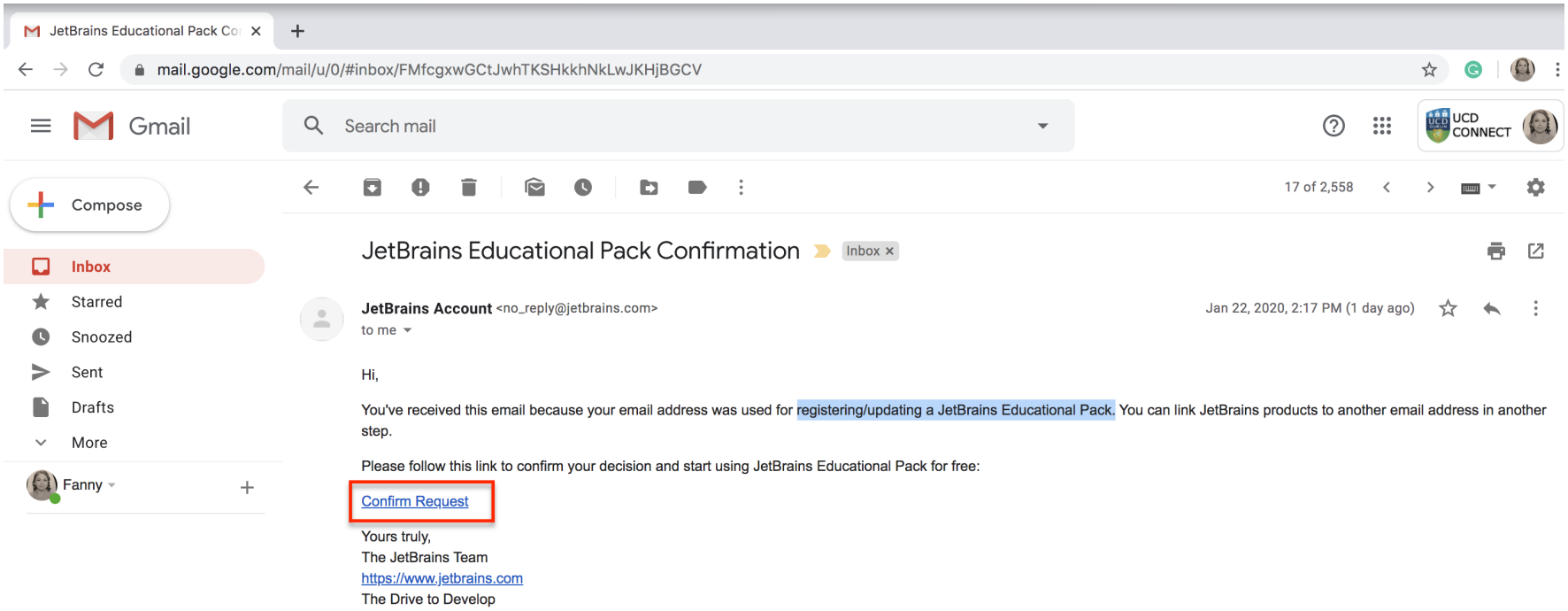
- Fill out the details using your Student UCD email address at:

https://www.jetbrains.com/shop/eform/students



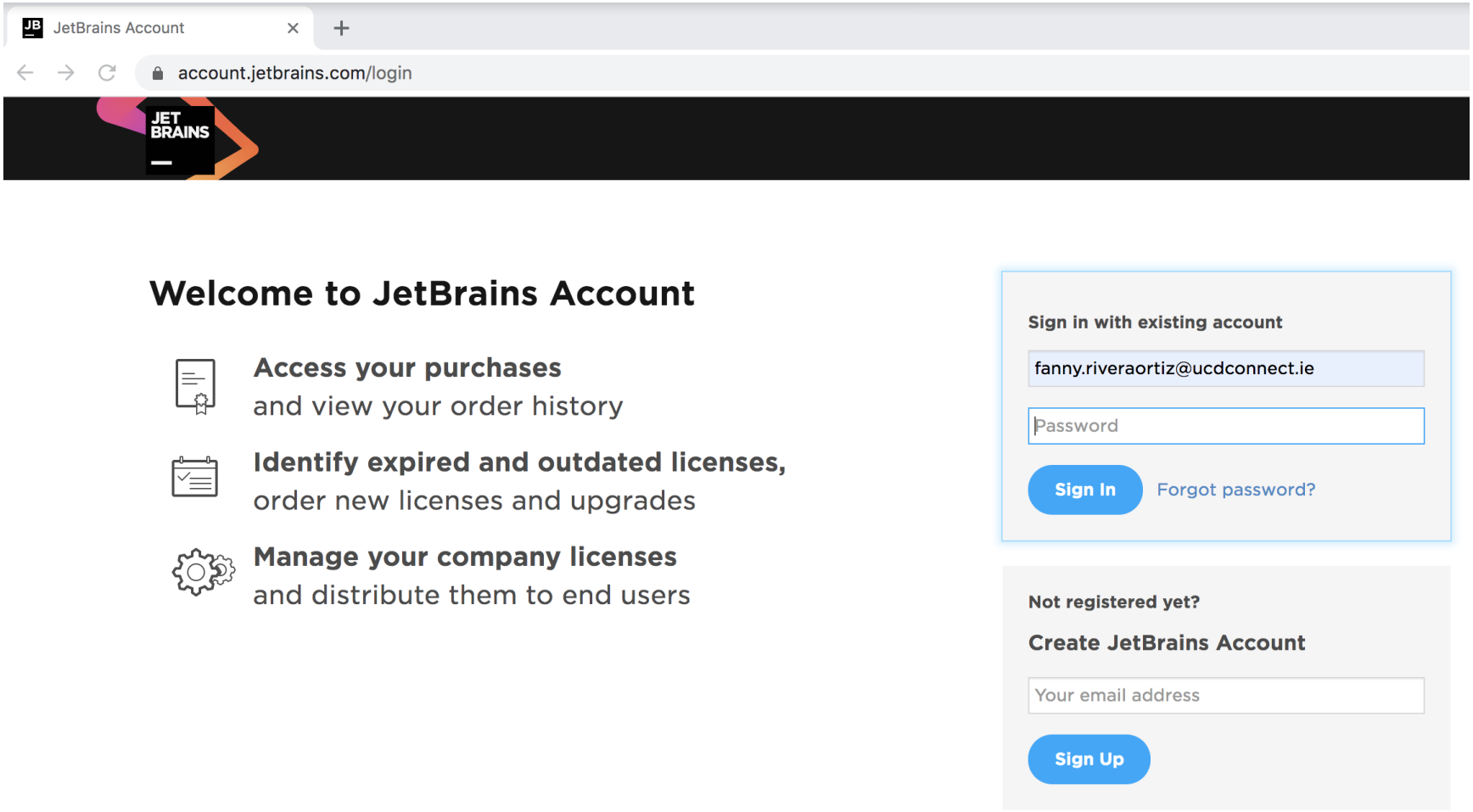
Obtain CLion

- You will get a confirmation email to register a JetBrains Educational Pack.
- Click on the Confirm Request Link.



Obtain CLion

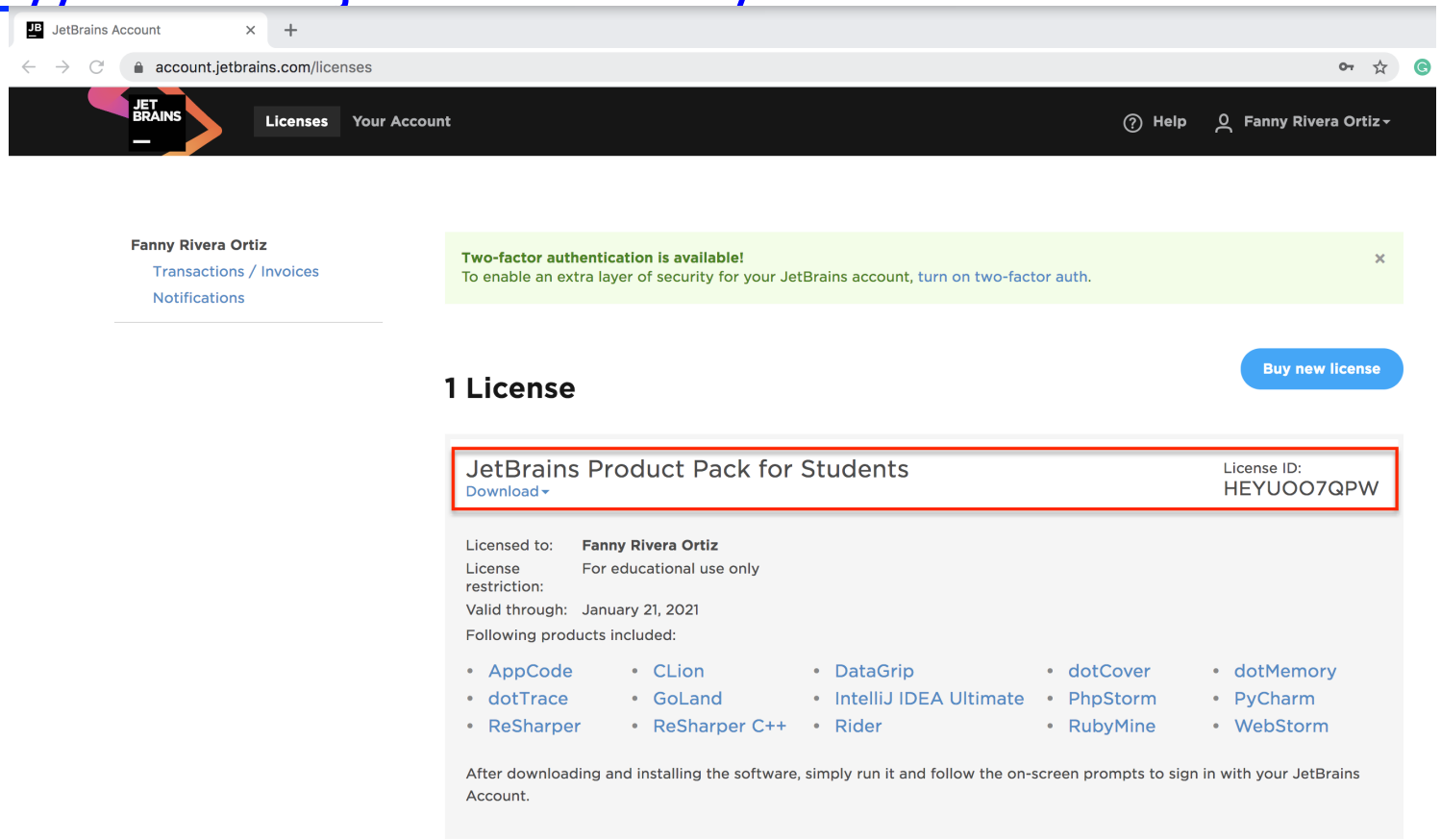
- Connect to your JetBrains (JB) account at: https://account.jetbrains.com/login



Obtain CLion

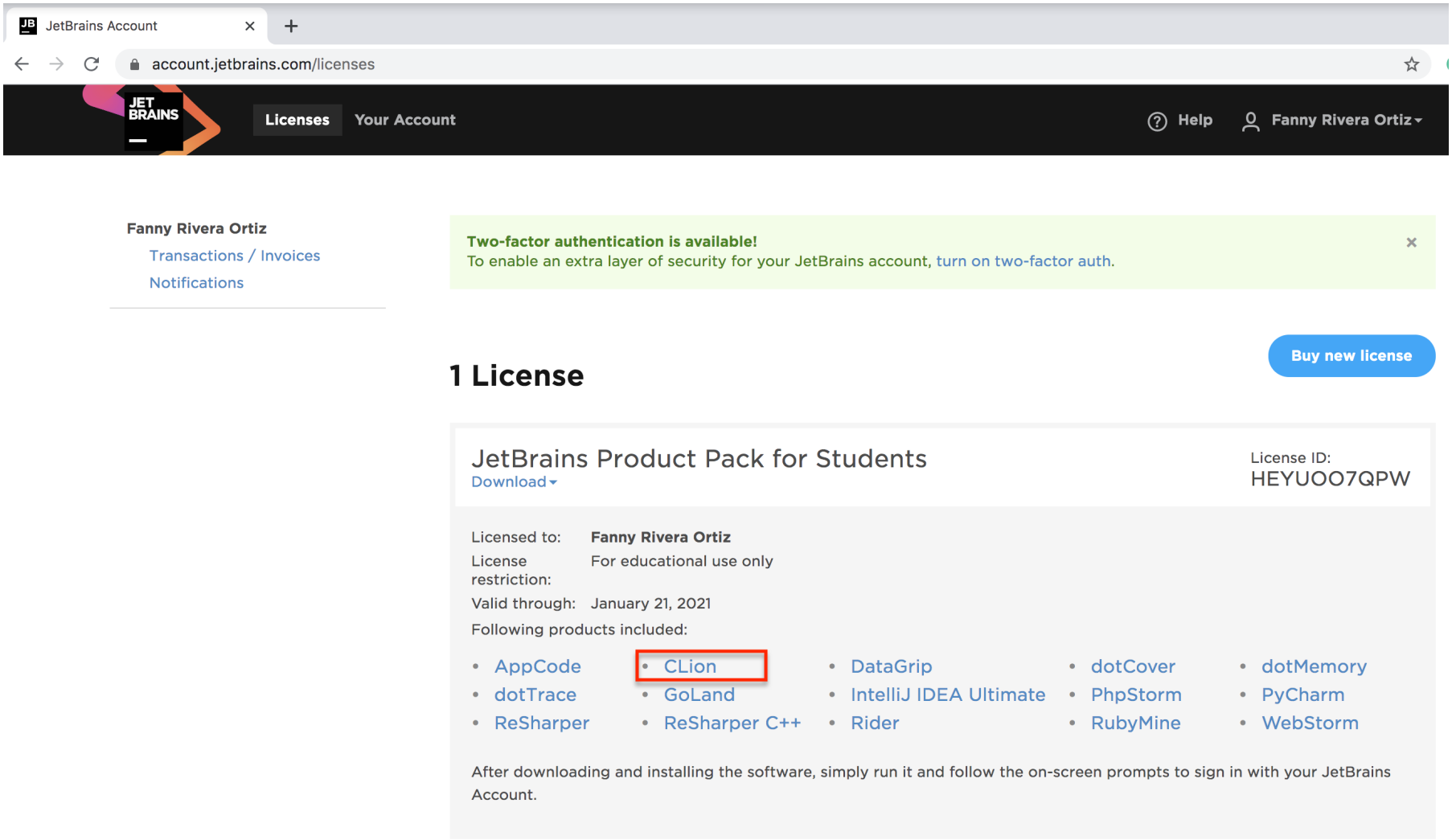
- Make sure you have a license Id for a JetBrains Product Pack for Students at:

https://account.jetbrains.com/licenses



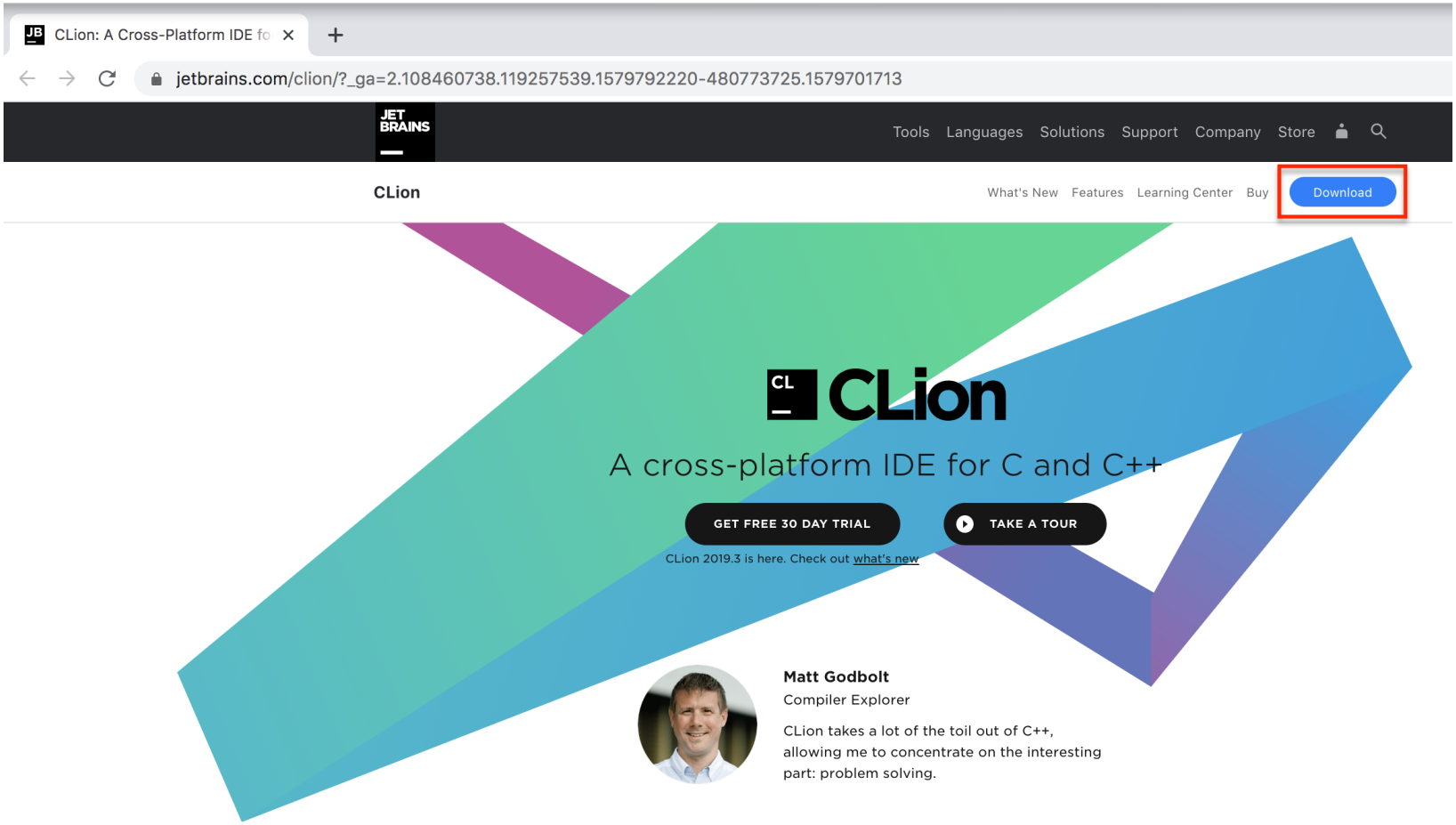
Obtain CLion

- Click on CLIon.



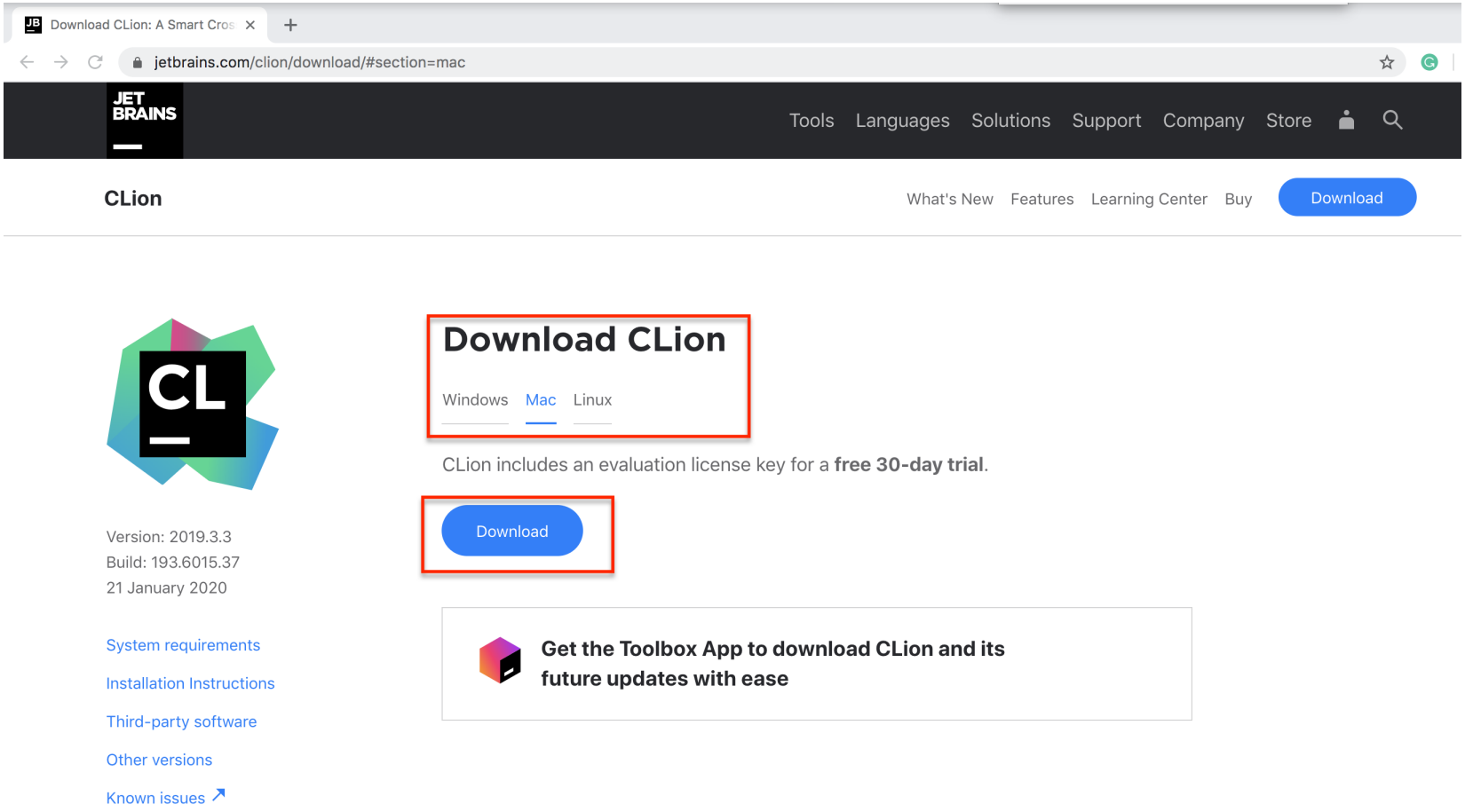
Install CLion

- Click on the Download button



Install CLion

- Run the installer (.exe on Windows, .dmg on MacOS) and follow the Wizard steps



Configure CLion

Toolchain Requirements

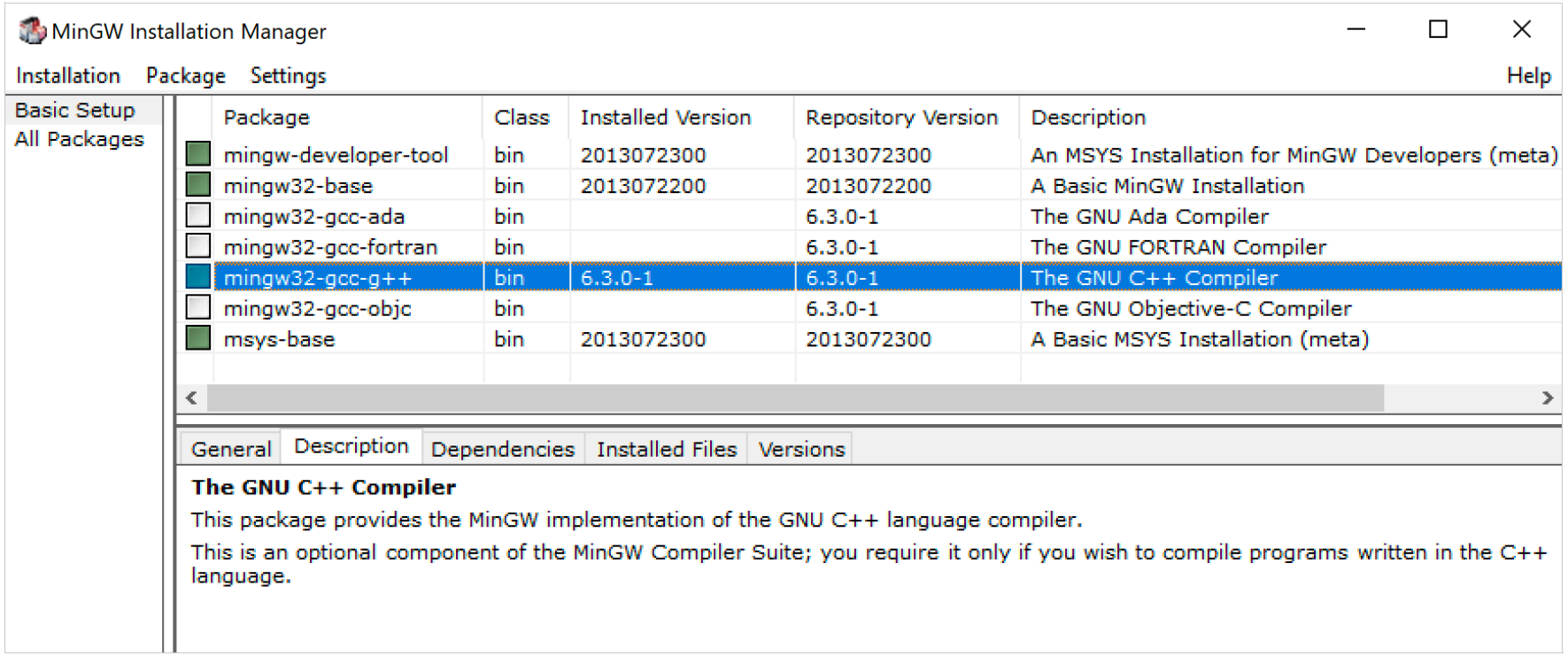
Ensure that you have one of these tools installed on your machine.

- **On Windows**
  - MinGW OR MinGW64 ➔(slides 17-19)
  - Cygwin > v2.8 ➔ (slides 20-22)
- **On MacOS**
  - XCode command line ➔ (slides 23-26)
- **On Linux**
  - GCC/G++ or Clang

Configure CLion on Windows using MinGW/MinGW64

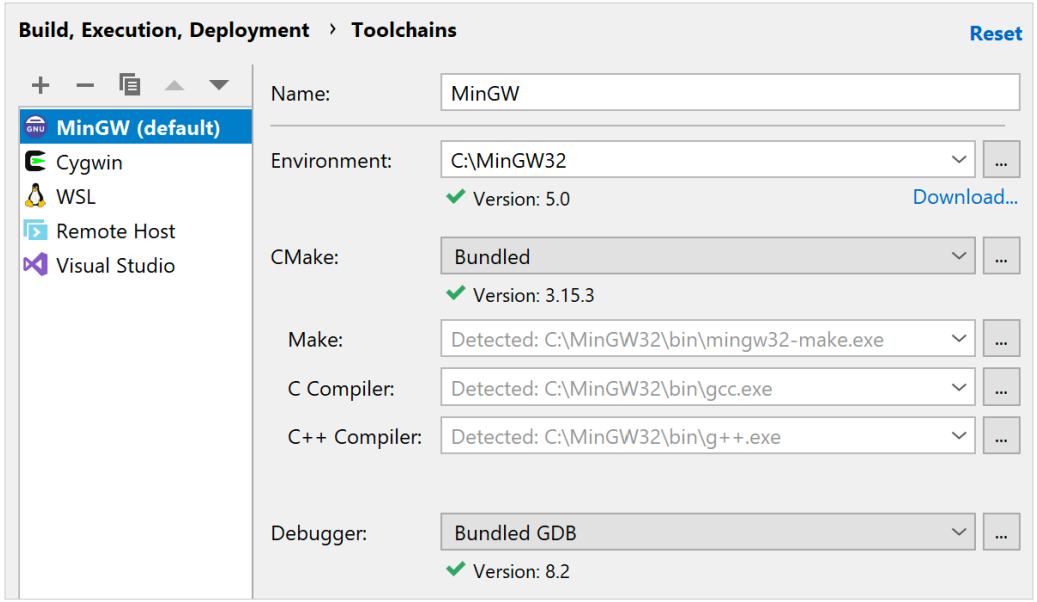
Installing MinGW/MinGW64

1. Download MinGW or MinGW64 installer
2. Run the installer and select the following packages in the **Basic Setup** list: `mingw-developer-tool, mingw32-base, mingw32-gcc-g++, mingw32-msys-base`



Configure CLion (MinGW/MinGW64)

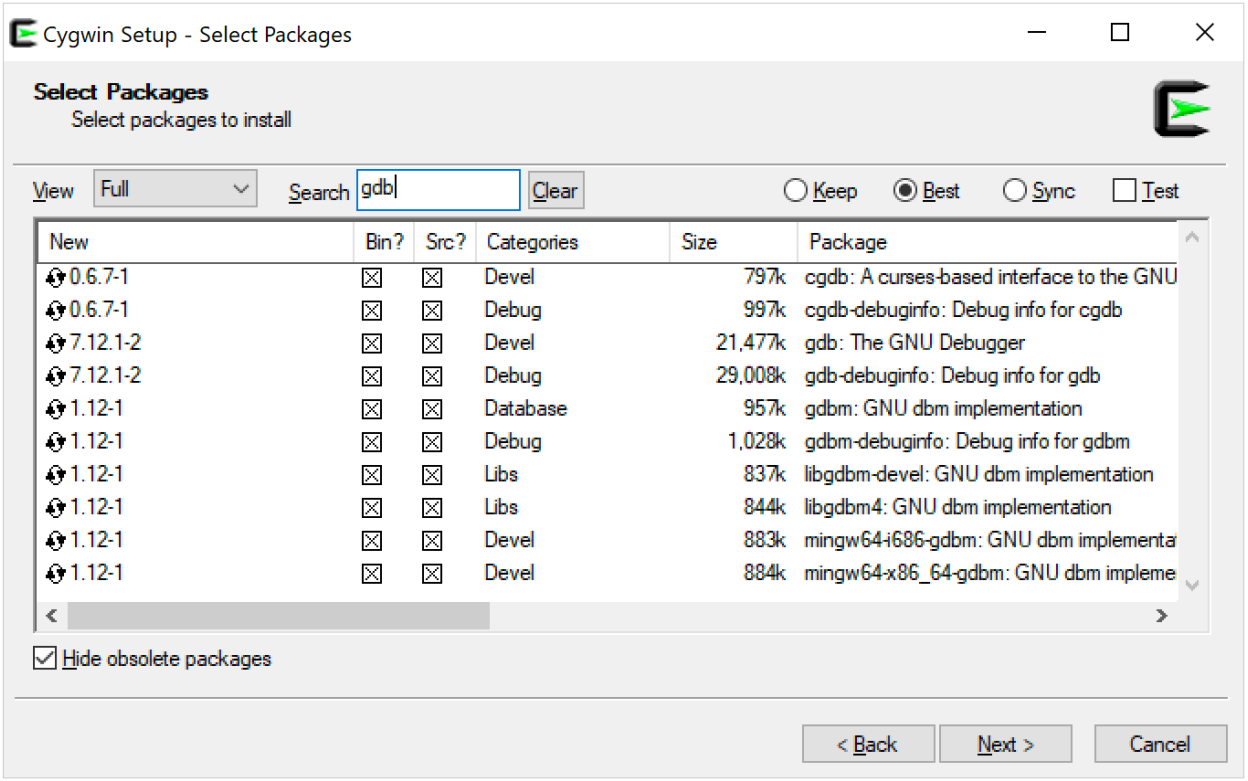
3. Once the installation is finished, open Clion and go to File | Settings | Build, Execution, Deployment | Toolchains.
4. Select MinGW from the Environment list. CLion attempts to detect the MinGW installation automatically. Check the detection result, and specify the path manually if required.
5. Wait until the tools detection finishes. If CLion cannot detect compilers or *make*, double-check the installed packages in MinGW Installation Manager. Press Apply when all the tools are set correctly.



Configure CLion on Windows using CygWin

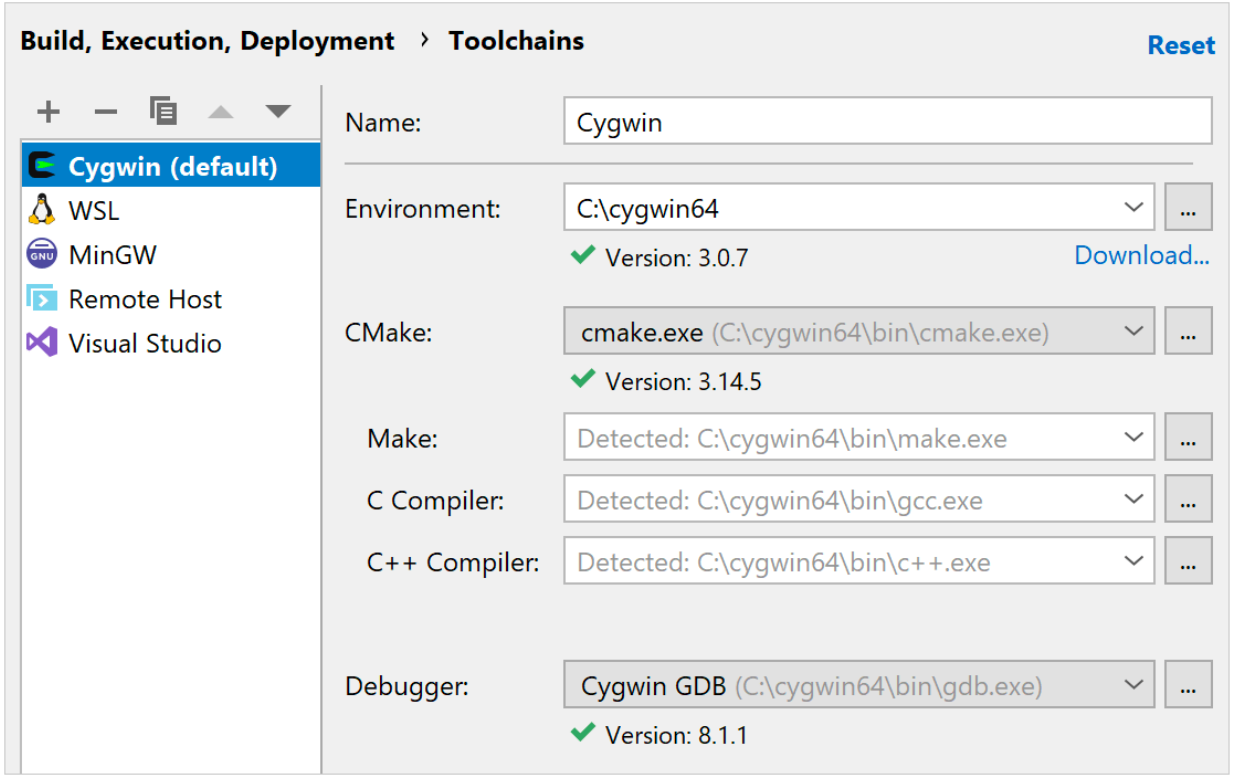
Install CygWin

1. Download the Cygwin installer v2.8 or later
2. Run the installer and select the following packages: `gcc-g++, make, gdb`
    1. To select a package, type its name in the **Search** field and then click it in the list until a tick mark appears in the **Bin?** column



Configure CLion (Cygwin)

3. Once the installation is finished, open Clion and go to **File | Settings | Build, Execution, Deployment | Toolchains**.
4. Select **Cygwin** from the **Environment** list. CLion attempts to detect the Cygwin installation automatically. Check the detection result, and specify the path manually if required.
5. Wait until detection finishes, and press **Apply**.



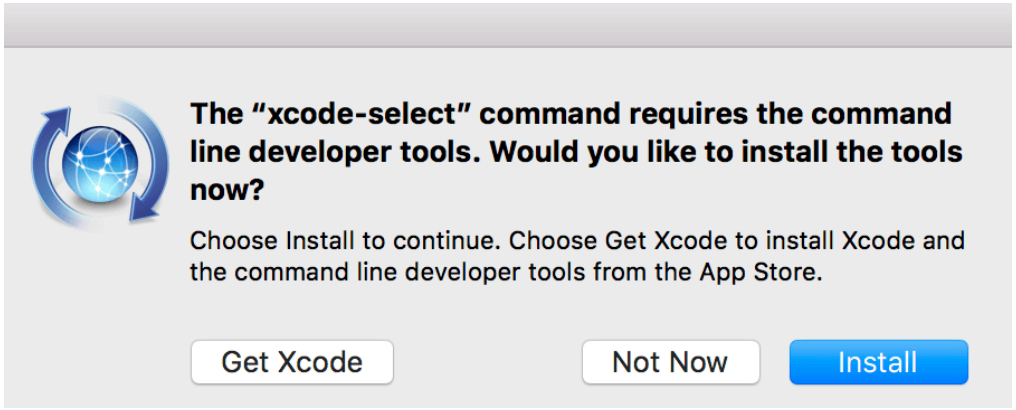
Configure CLion on MacOS using XCode

Install Xcode (1/3)

1. Open the terminal and run the following command:

   ```
   xcode-select –install
   ```

2. When prompted to install command line developer tools, click the **Install** button:



Install Xcode (2/3)

3. With Xcode command line tools, you get the Clang compiler installed by default. To check the compiler presence and its version, run on the terminal: `clang --version`

Install Xcode (3/3)

Command line tools may not update automatically along with the system or Xcode update.

This may cause error messages like invalid active developer path during project loading in CLion.

To fix this, run the `same xcode-select --install` command on the terminal, and the tools will be updated accordingly.

Create and Manage Projects

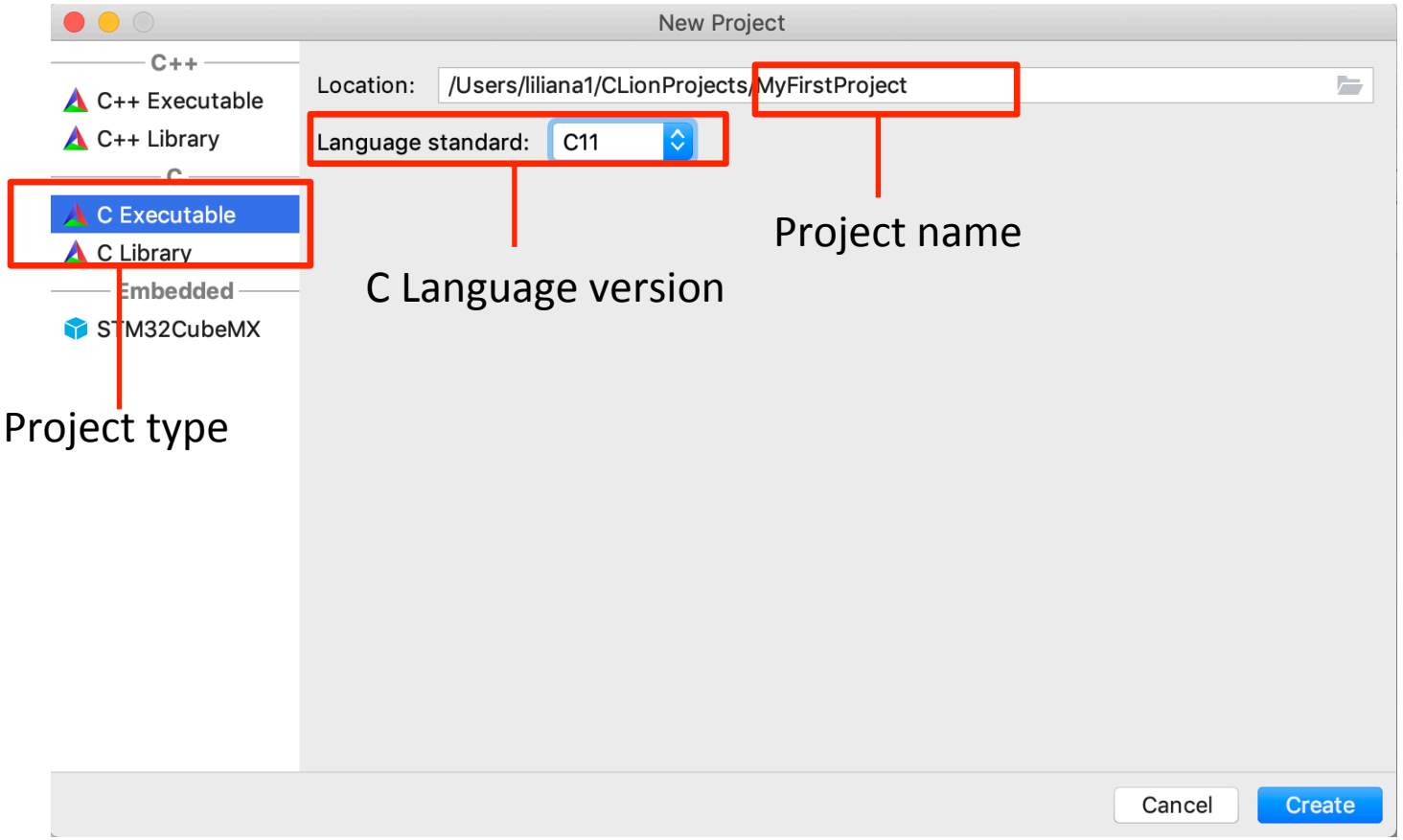
Projects

- A **project** groups a set of related source and library files as well as configuration files.
- Different types of projects depending on the objective
  - We will focus on C Executable projects
    - Mainly contain C source files created from scratch and compiled code
- Each project is contained in a subdirectory inside a project location in your filesystem.

Create a New C Project

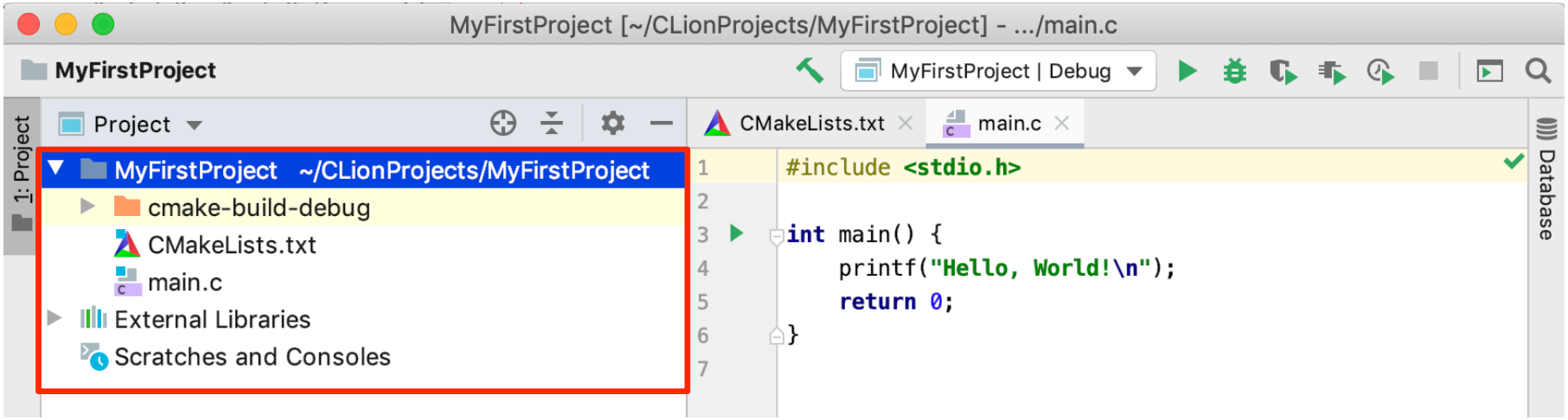
Create a New Project (1/2)

- On the left pane, select the Project Type (C Executable)
- Input a **Location**, i.e. directory where you want your project to be contained



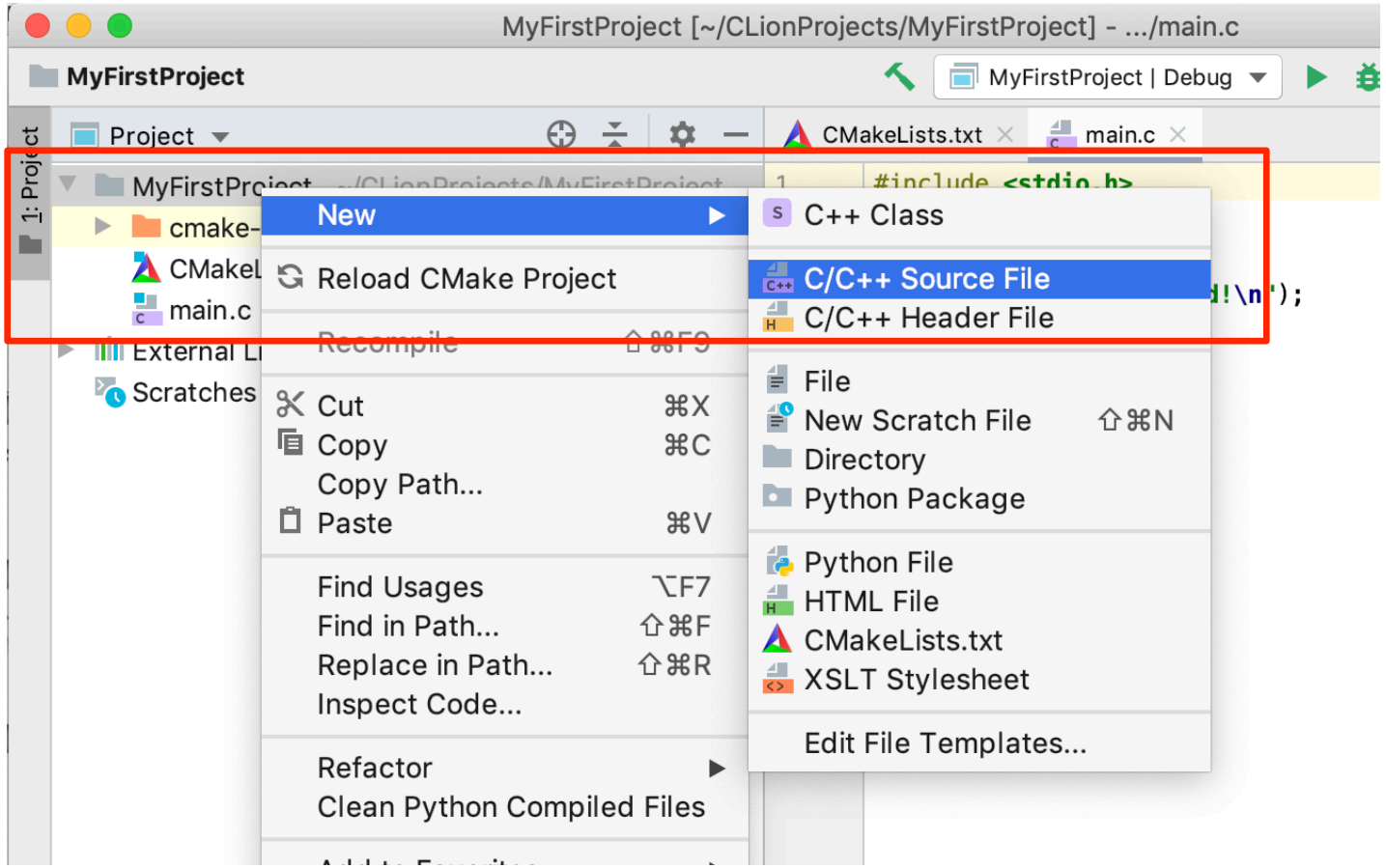
Create a New Project (2/2)

- You can now see your project inside the workspace and the default source file (*main.c)* contains



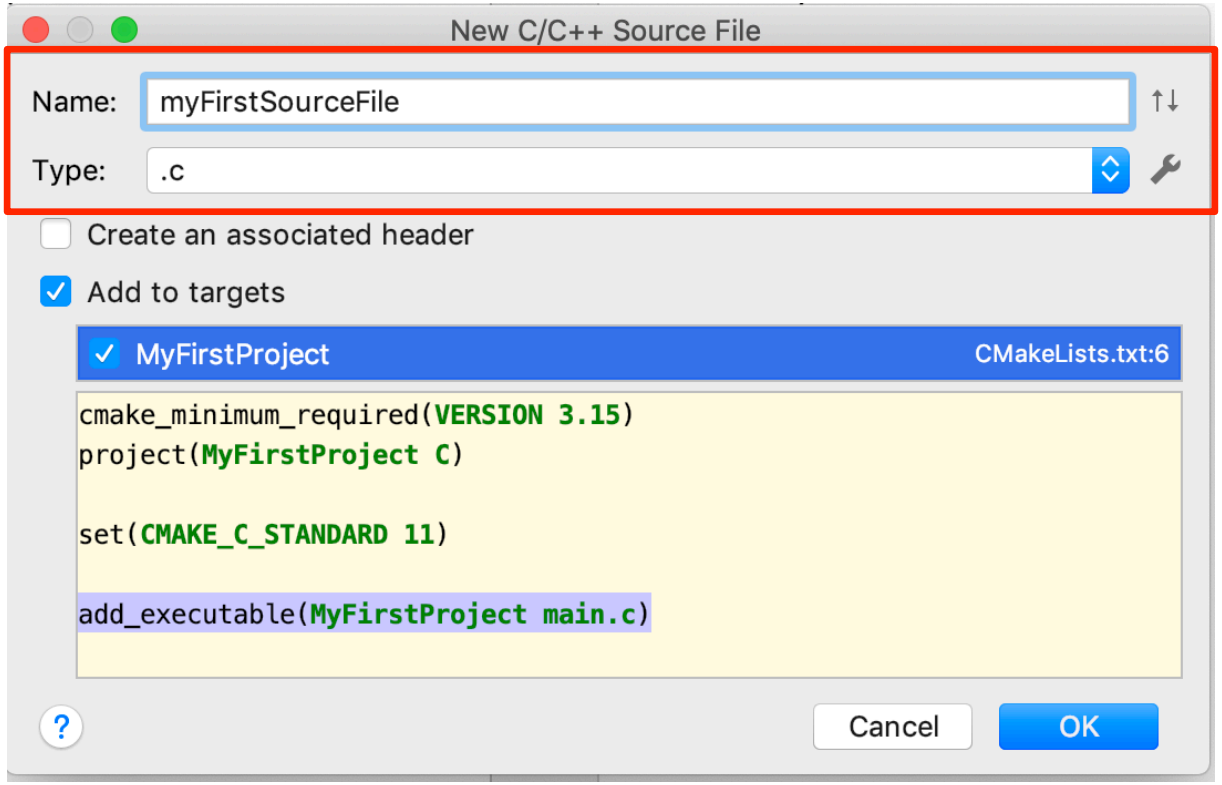
Add Files to Your Project

Add a new C Source File to the Project



- Right-click on the Project and select *New C/C++ Source File*

Creating a New C Source File



- Provide the name of the file (e.g., *myFirstSourceFile*) and the type (*.c*)
- Select *OK*

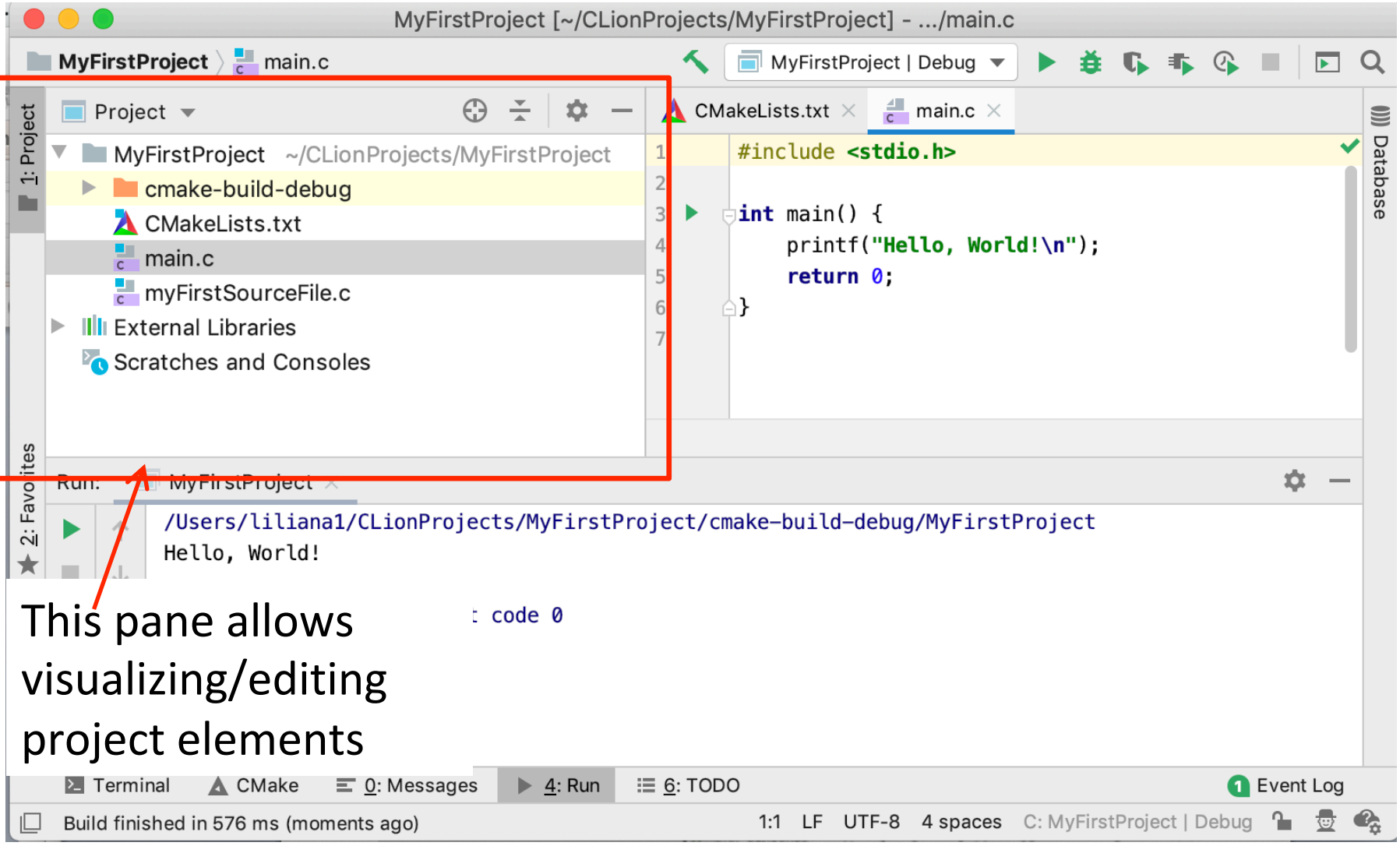
BrightSpace

Download Files

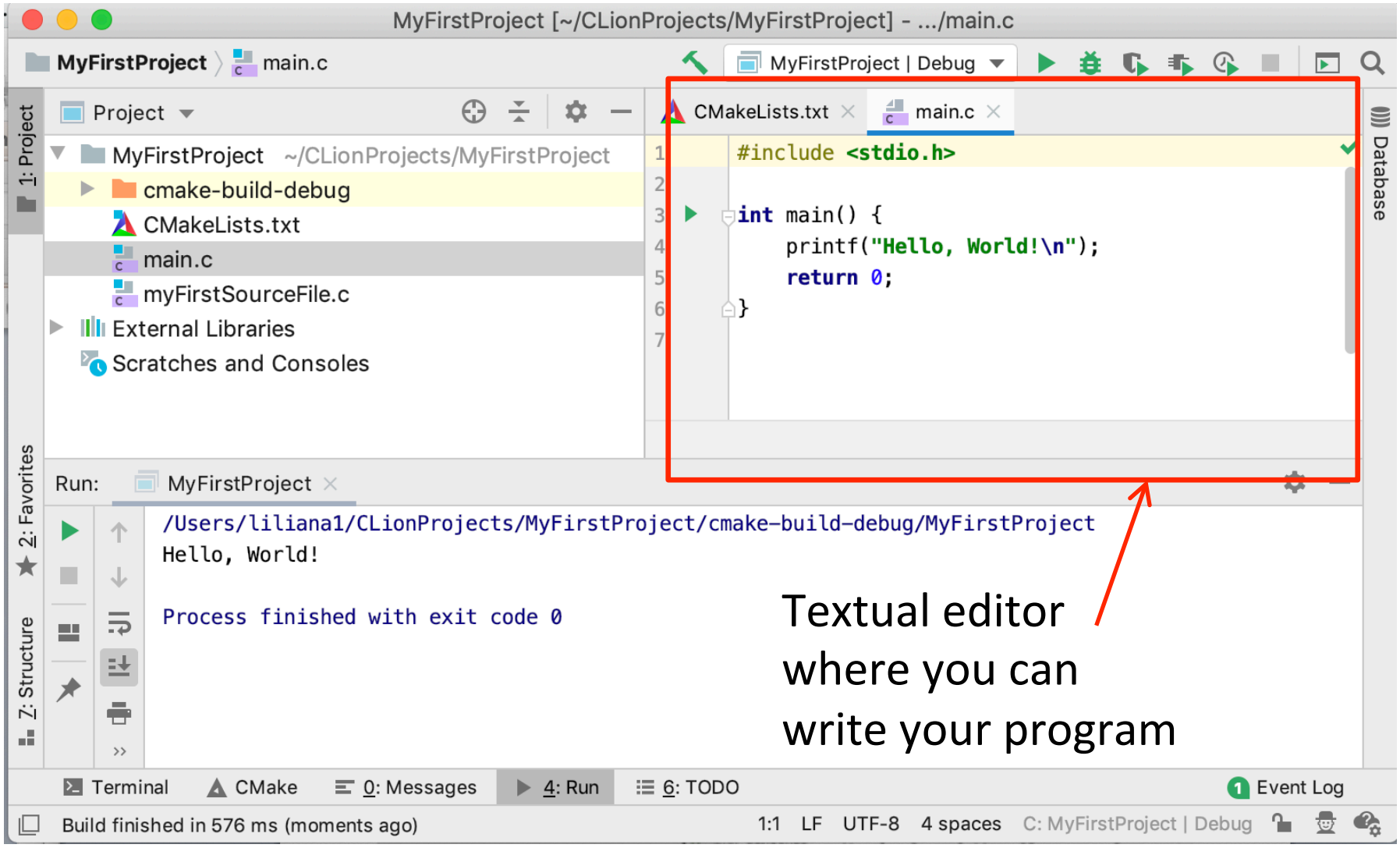
- Connect to UCD BrightSpace at https://brightspace.ucd.ie/d2l/home/54656
- Download from the Week 1 >> Lab 1 the following files:
  - factorial.c
  - example1.c

CLion IDE Components

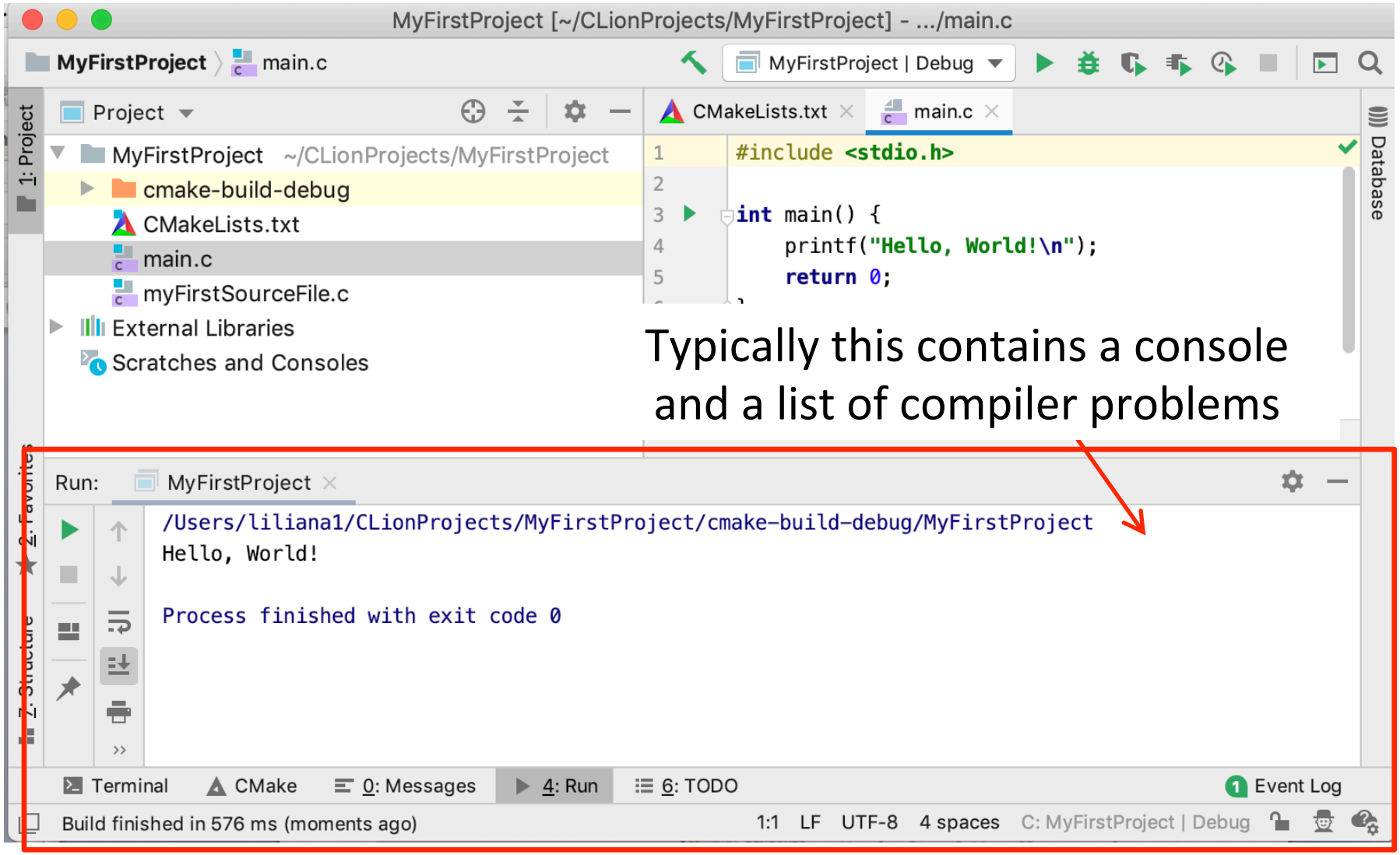
IDE Components



IDE Components

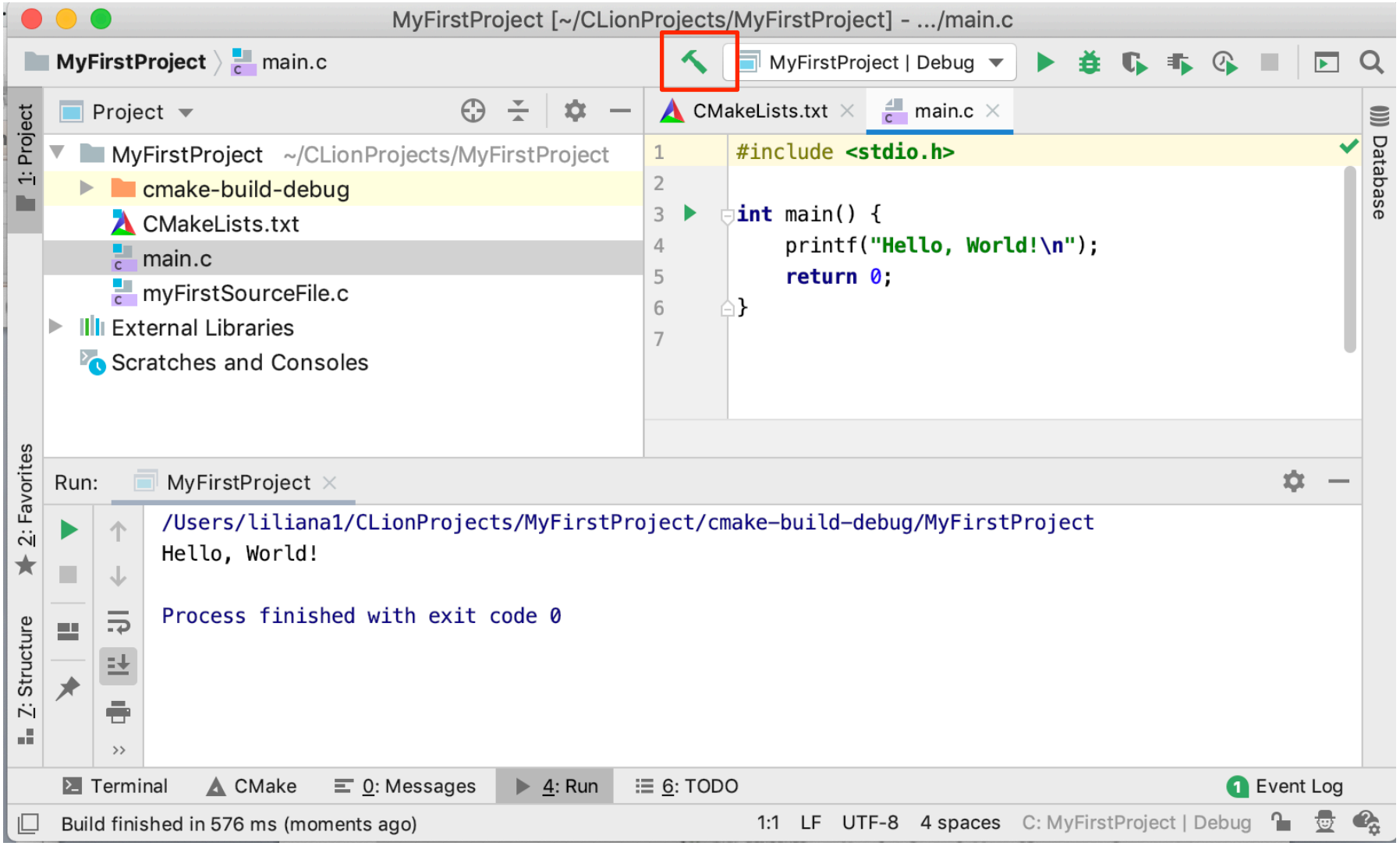


IDE Components



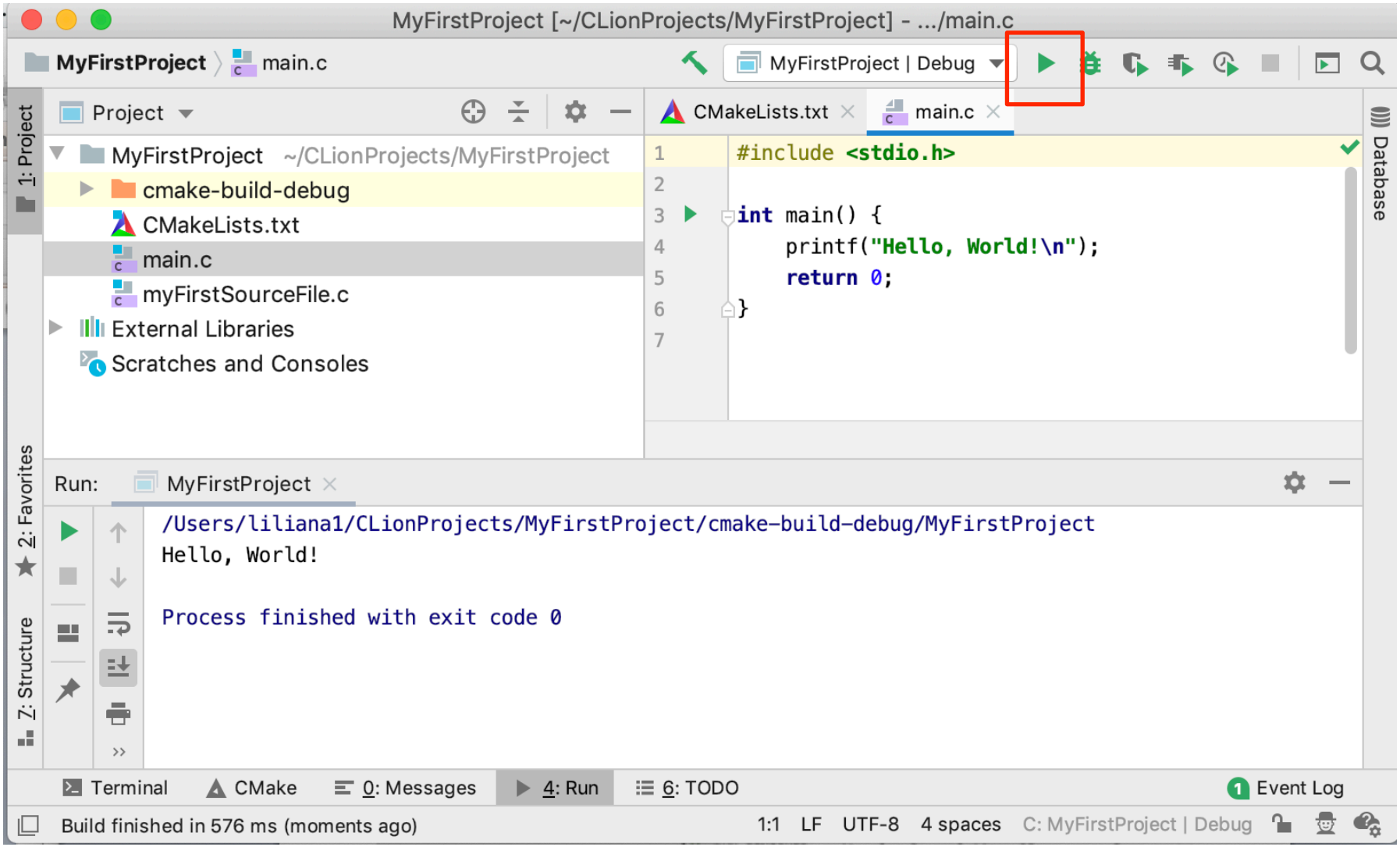
Compile file main.c

- Select the main file and press the *Build Button*



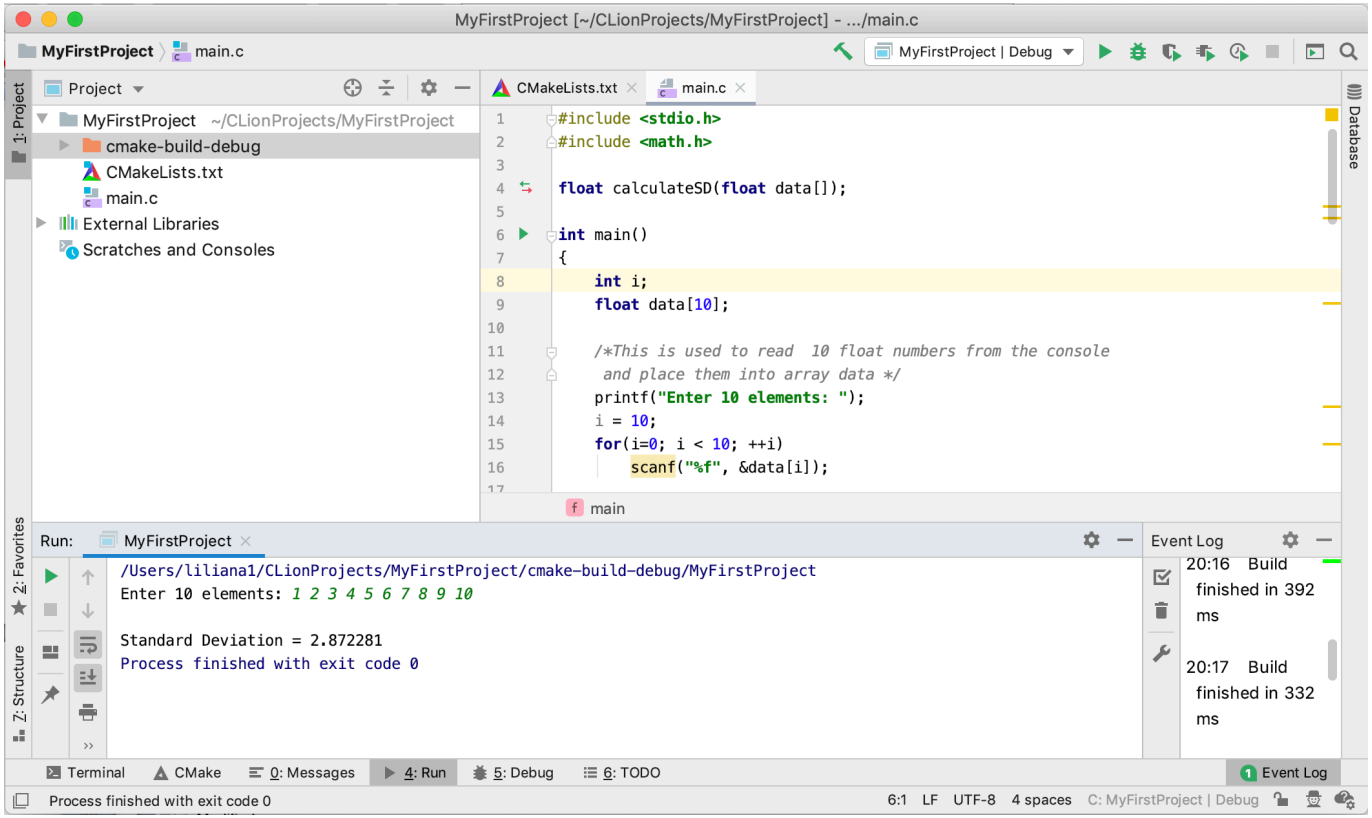
Execute it

- Click on the Run button



Try it yourself!

- From BrightSpace, from the laboratory 1, download the example1.c file
- Copy the content of file *example1.c* in your *main.c* file
- Compile it and execute it



Debugging

Debugging

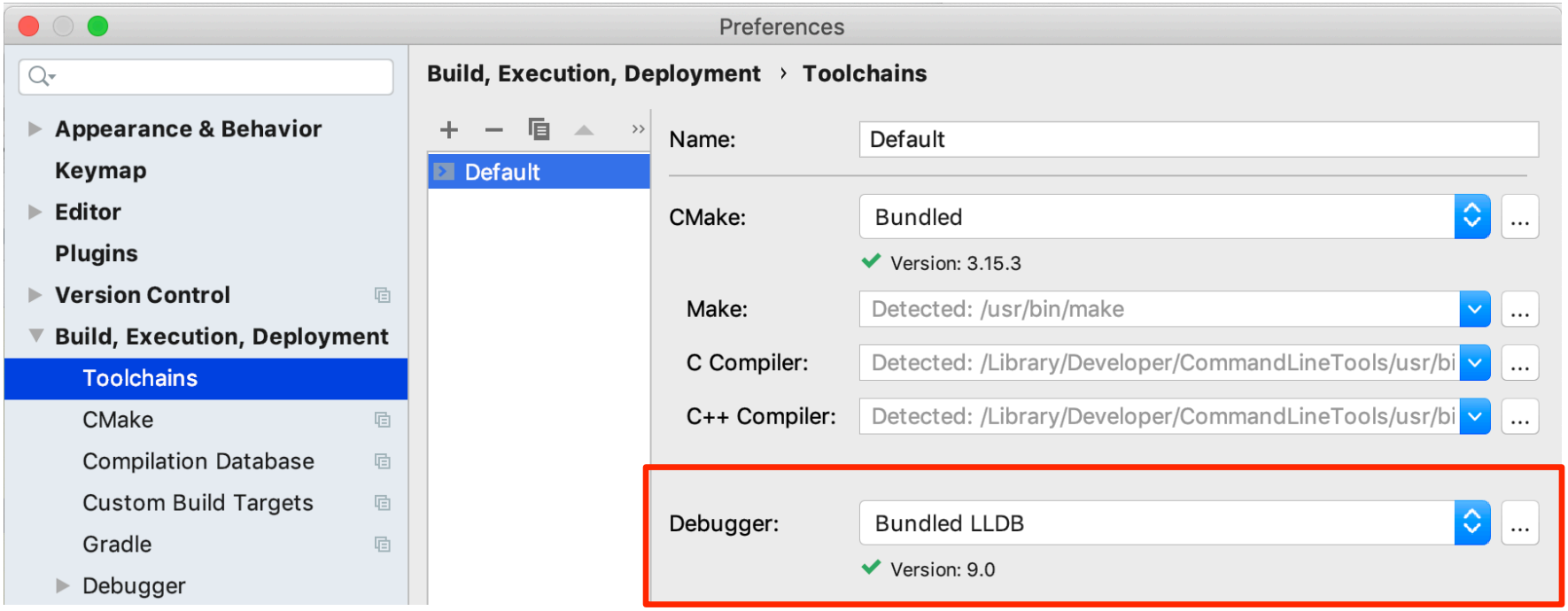
As your programs become more complicated, there will be a need to trace the program execution step by step or place *break points* where you wish the program to pause.

- A debugger can pause your program and you can watch the values of the variables that you have defined.

Prerequisites

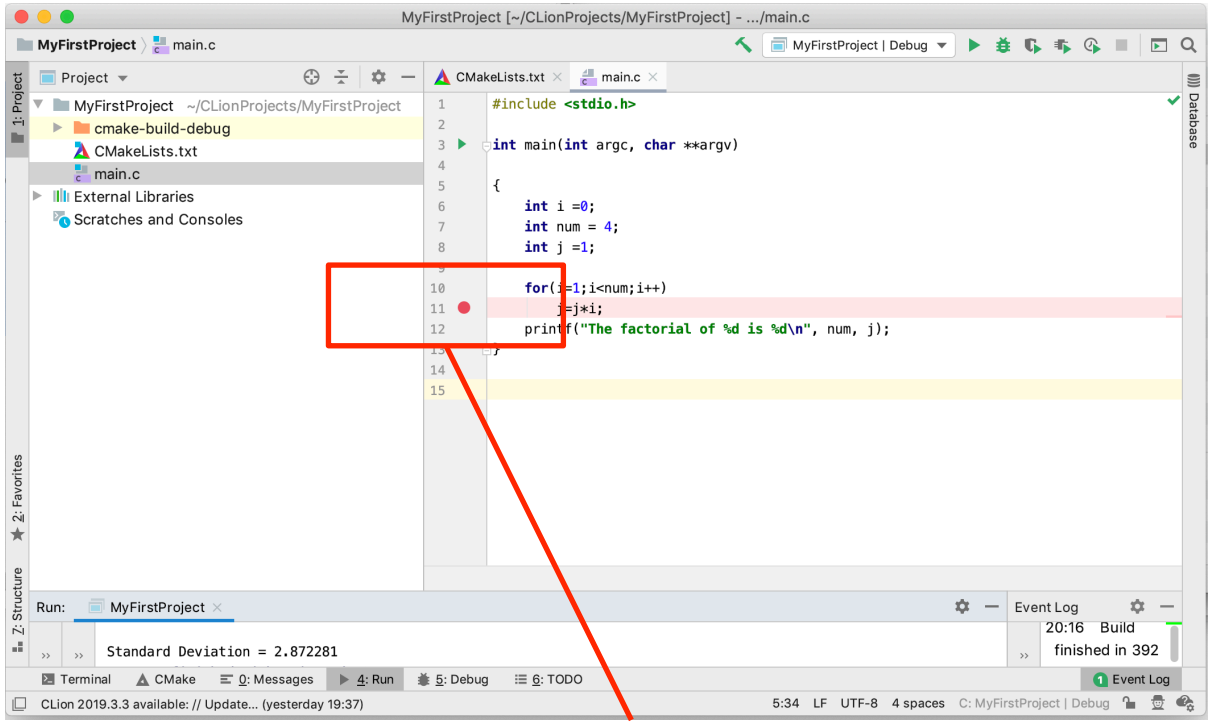
Ensure that a debugger is configured in your toolchains.

Select Clion >> Preferences >> Build, Execution Deployment >> Toolchains >> Default



Debugging

- From BrightSpace, from the laboratory 1, download the *factorial.c* file
- Copy the content of file *factorial.c* in your main file.



- Click on the left side for the line containing instruction

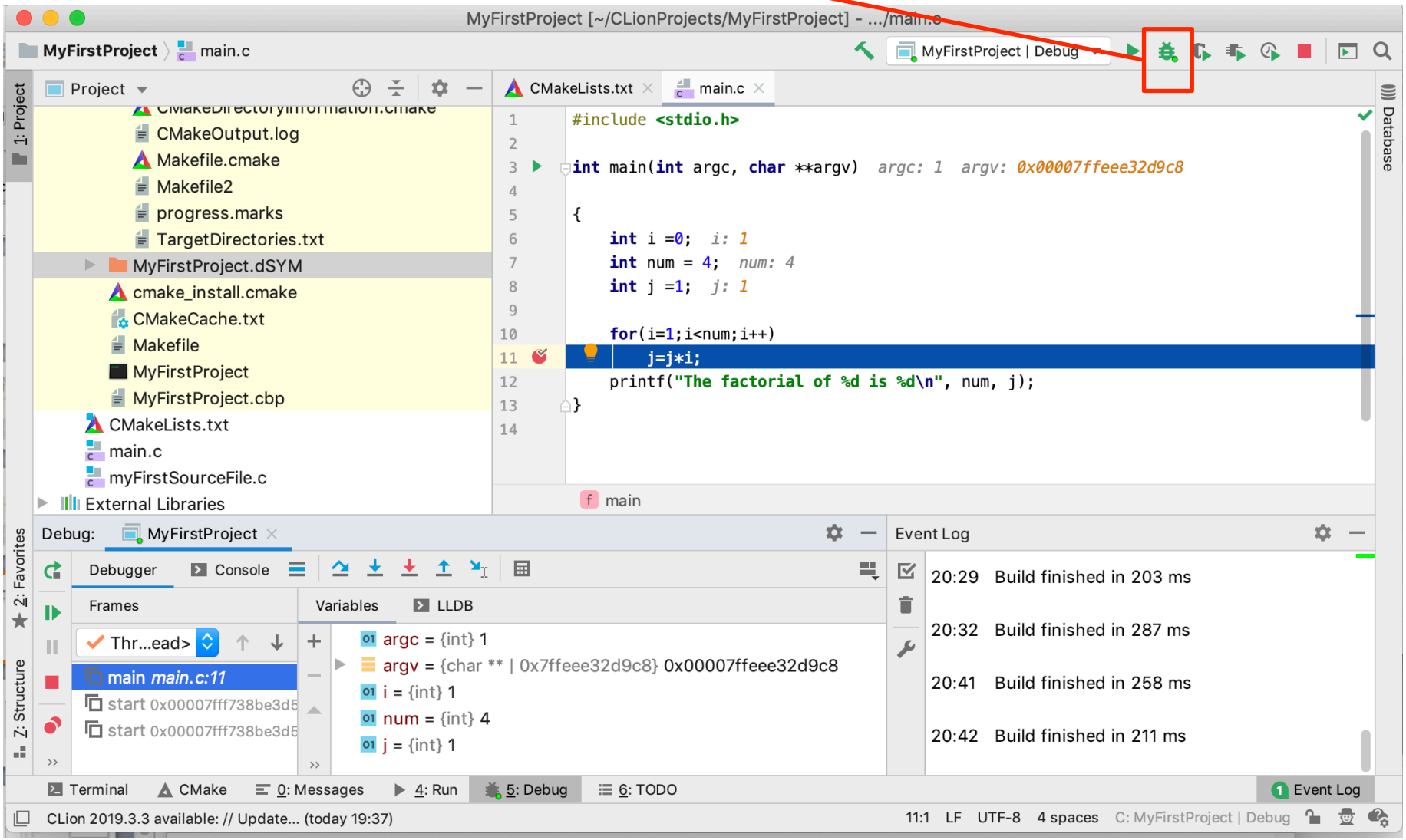
```
j = j*i;
```

- A red circle will appear. This is called **breakpoint.**

A breakpoint indicates where you want to stop the execution of your program.

Debugging

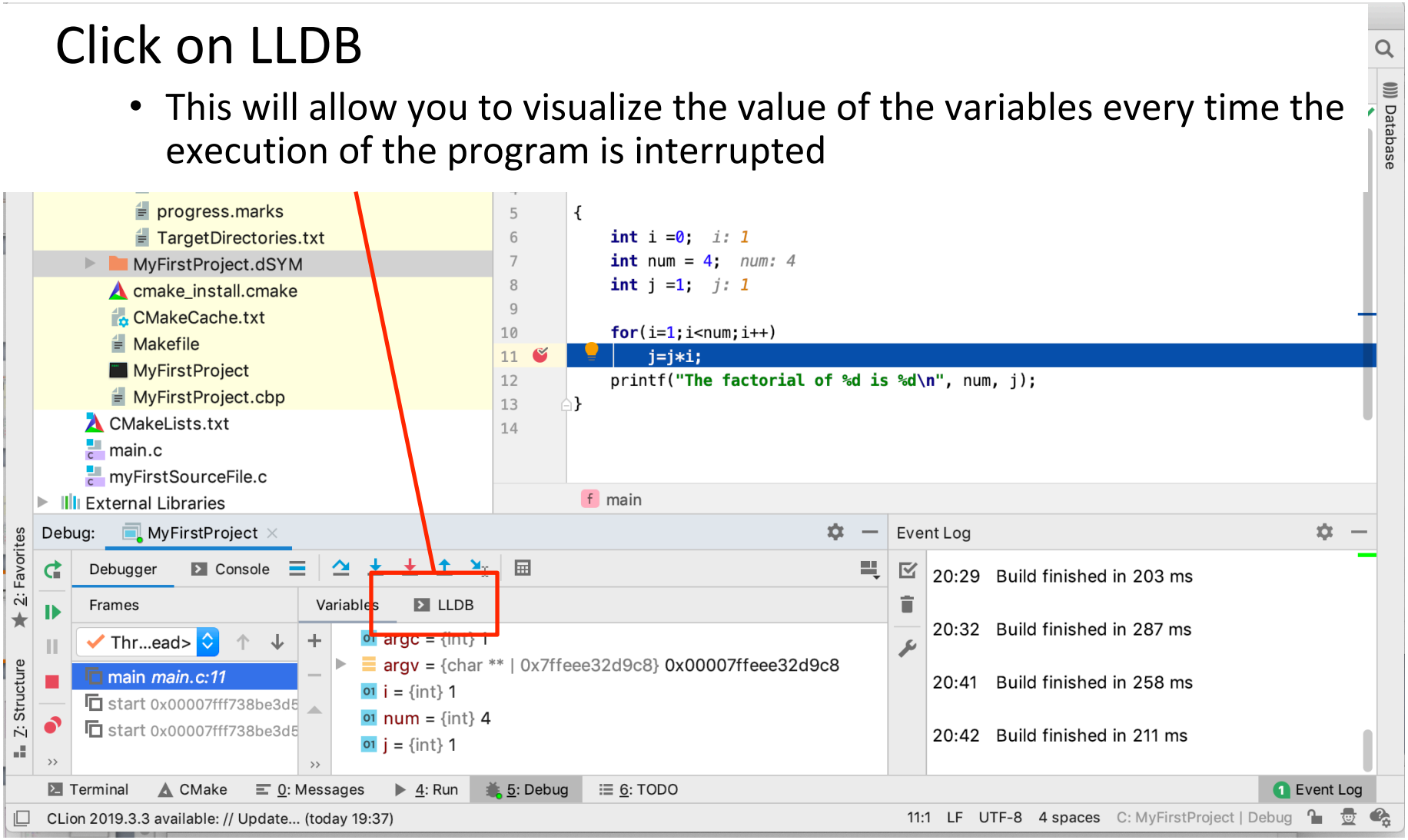
- Click on the debug button.



Debugging

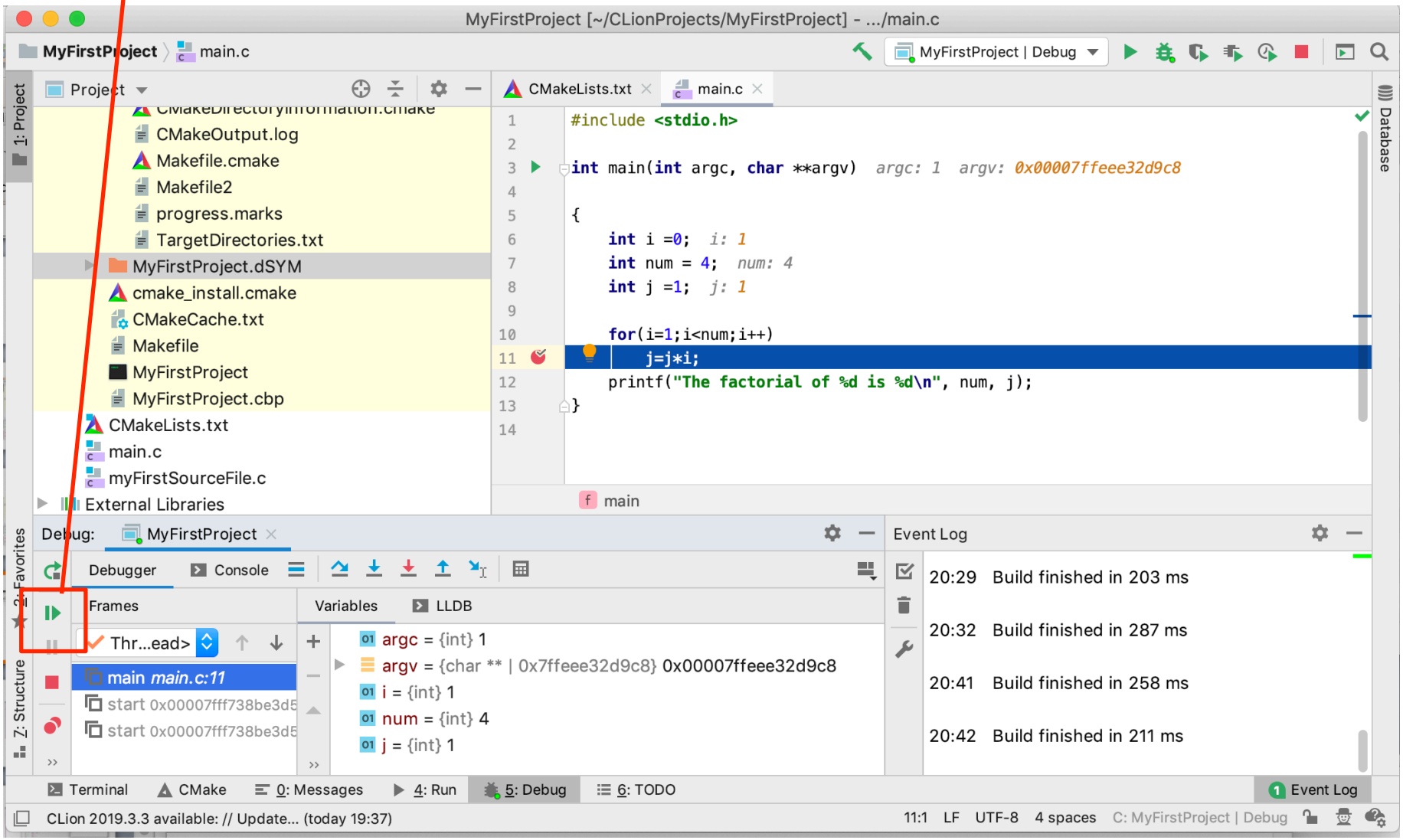
Click on LLDB

- This will allow you to visualize the value of the variables every time the execution of the program is interrupted

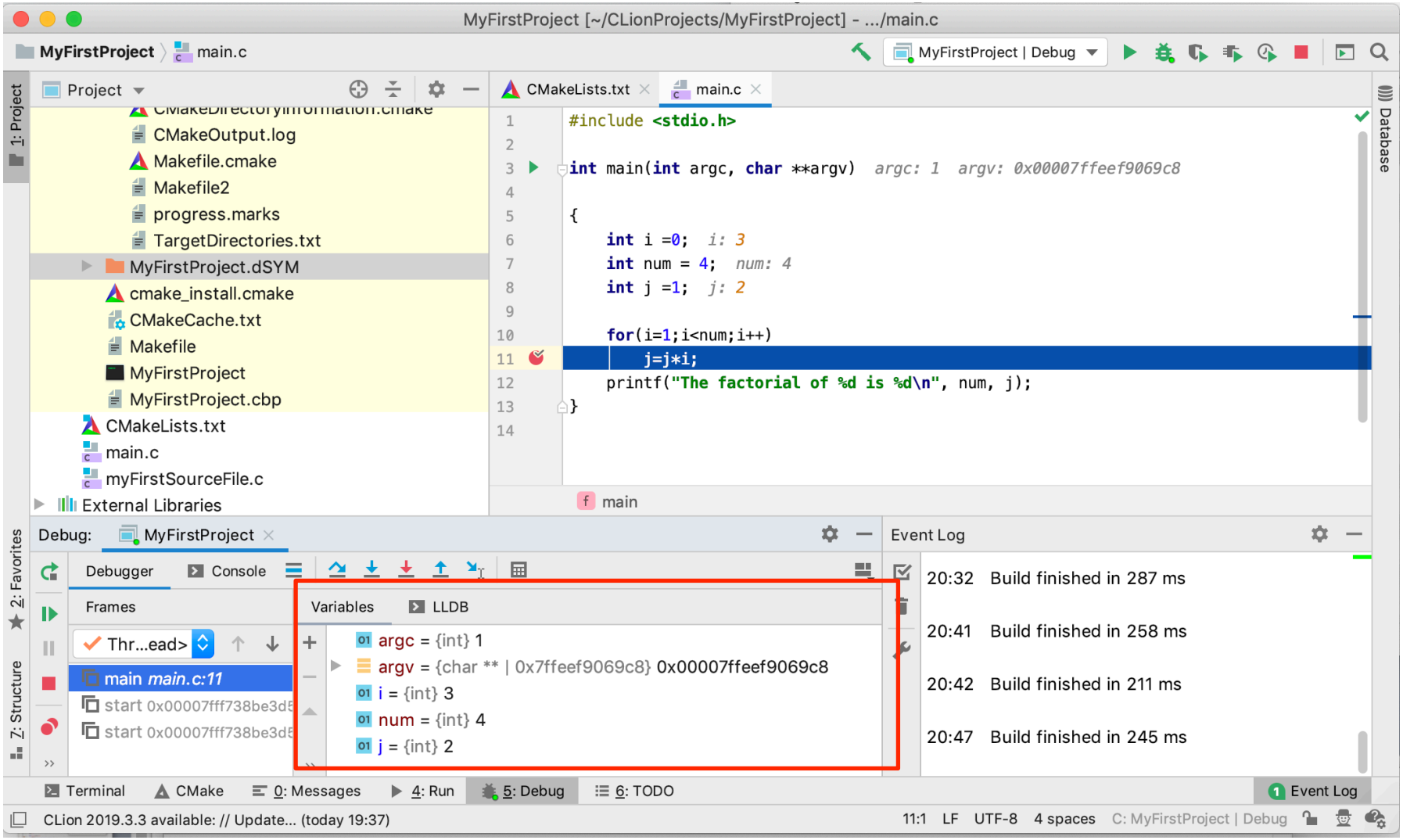


Debugging

- Press Resume Program

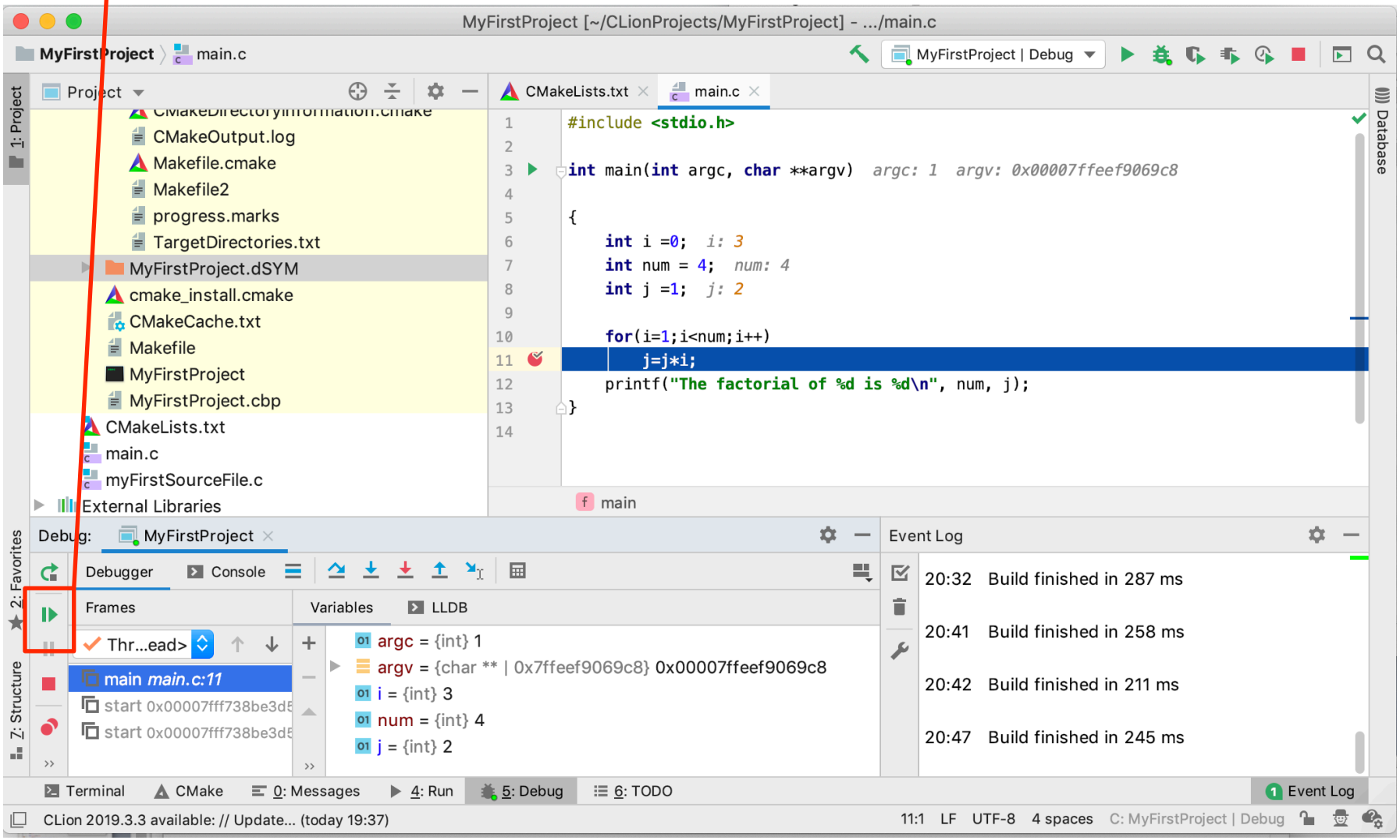


Debugging

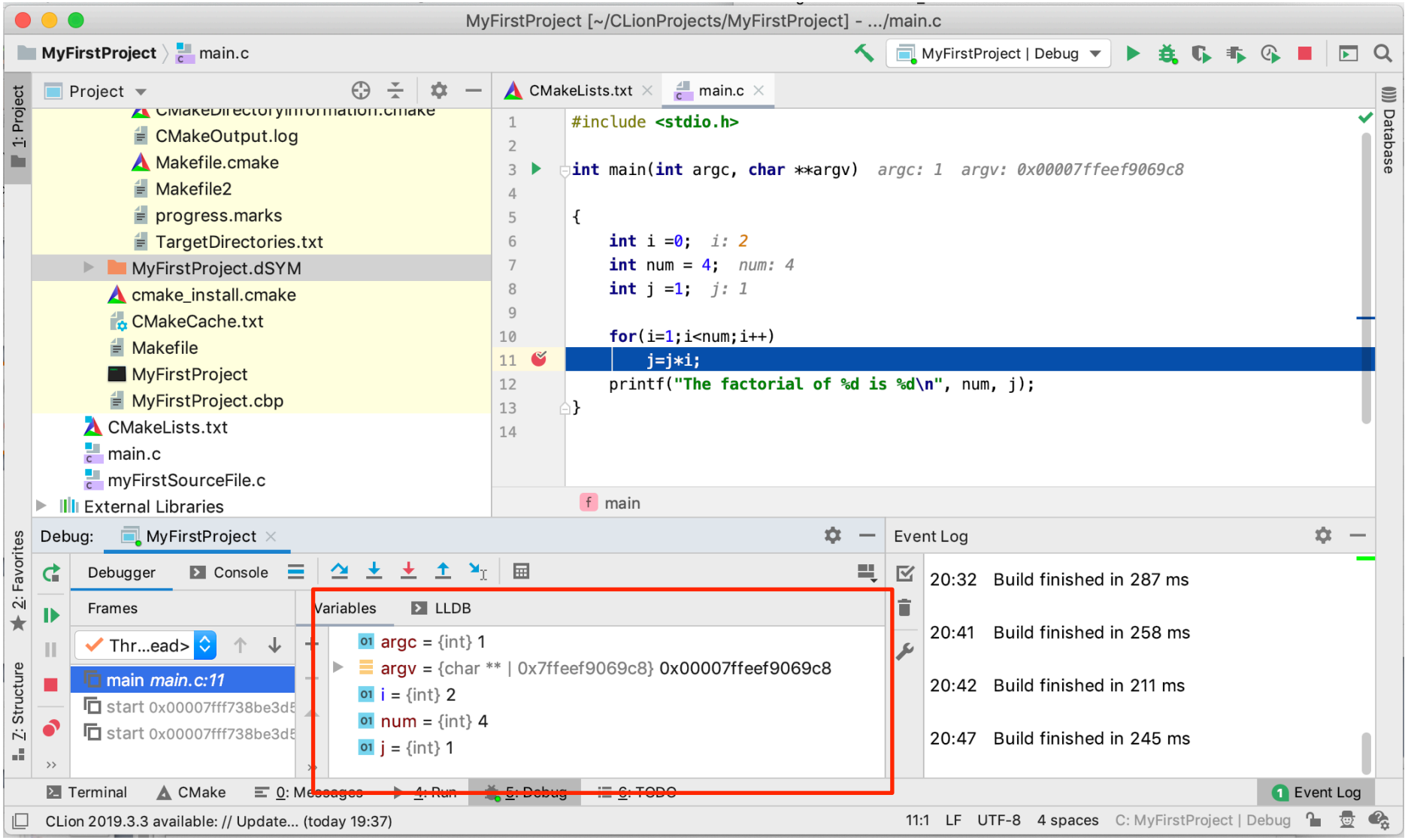

MyFirstProject [~/CLionProjects/MyFirstProject] - .../main.c
MyFirstProject | Debug
Project
CMakeOutput.log
Makefile.cmake
Makefile2
progress.marks
TargetDirectories.txt
MyFirstProject.dSYM
cmake_install.cmake
CMakeCache.txt
Makefile
MyFirstProject
MyFirstProject.cbp
CMakeLists.txt
main.c
myFirstSourceFile.c
External Libraries
CMakeLists.txt
main.c
#include <stdio.h>
int main(int argc, char **argv) argc: 1 argv: 0x00007ffeef9069c8
{
int i =0; i: 3
int num = 4; num: 4
int j =1; j: 2
for(i=1;i<num;i++)
j=j*i;
printf("The factorial of %d is %d\n", num, j);
}
main
Debug: MyFirstProject
Debugger
Console
Frames
Variables
LLDB
Thr...ead>
main main.c:11
start 0x00007fff738be3d5
start 0x00007fff738be3d5
argc = {int} 1
argv = {char ** | 0x7ffeef9069c8} 0x00007ffeef9069c8
i = {int} 3
num = {int} 4
j = {int} 2
Event Log
20:32 Build finished in 287 ms
20:41 Build finished in 258 ms
20:42 Build finished in 211 ms
20:47 Build finished in 245 ms
Terminal
CMake
0: Messages
4: Run
5: Debug
6: TODO
Event Log
CLion 2019.3.3 available: // Update... (today 19:37)
11:1 LF UTF-8 4 spaces C: MyFirstProject | Debug

Debugging

- Press Resume Program

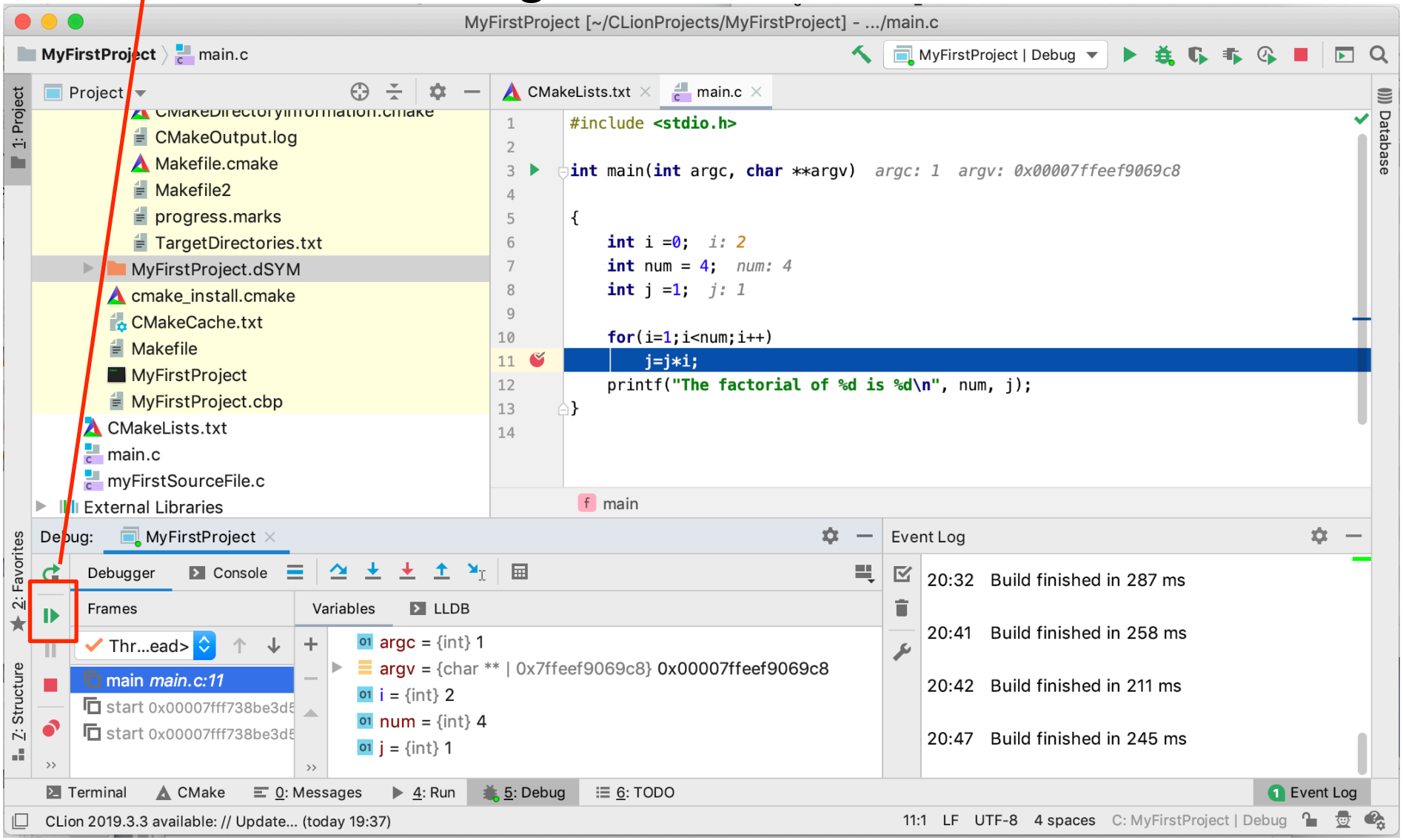


Debugging


MyFirstProject [~/CLionProjects/MyFirstProject] - .../main.c
MyFirstProject | Debug
Project
CMakeOutput.log
Makefile.cmake
Makefile2
progress.marks
TargetDirectories.txt
MyFirstProject.dSYM
cmake_install.cmake
CMakeCache.txt
Makefile
MyFirstProject
MyFirstProject.cbp
CMakeLists.txt
main.c
myFirstSourceFile.c
External Libraries
CMakeLists.txt
main.c
#include <stdio.h>
int main(int argc, char **argv) argc: 1 argv: 0x00007ffeef9069c8
{
int i =0; i: 2
int num = 4; num: 4
int j =1; j: 1
for(i=1;i<num;i++)
j=j*i;
printf("The factorial of %d is %d\n", num, j);
}
main
Debug: MyFirstProject
Debugger
Console
Frames
Variables
LLDB
Thr...ead>
main main.c:11
start 0x00007fff738be3d5
start 0x00007fff738be3d5
argc = {int} 1
argv = {char ** | 0x7ffeef9069c8} 0x00007ffeef9069c8
i = {int} 2
num = {int} 4
j = {int} 1
Event Log
20:32 Build finished in 287 ms
20:41 Build finished in 258 ms
20:42 Build finished in 211 ms
20:47 Build finished in 245 ms
Terminal
CMake
0: Messages
4: Run
5: Debug
6: TODO
Event Log
CLion 2019.3.3 available: // Update... (today 19:37)
11:1 LF UTF-8 4 spaces C: MyFirstProject | Debug
1: Project
2: Favorites
7: Structure
Database

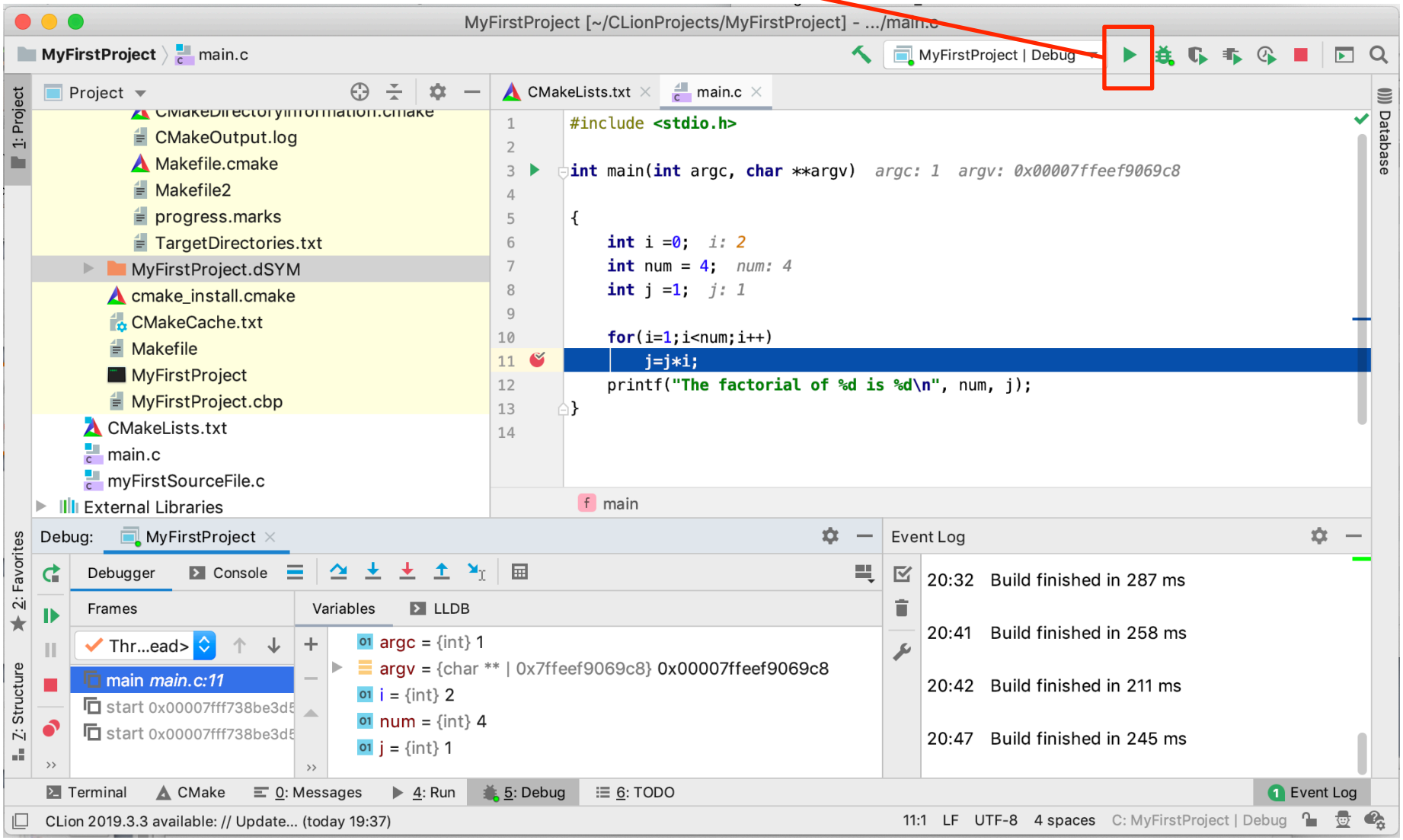
Debugging

- Press Resume Program



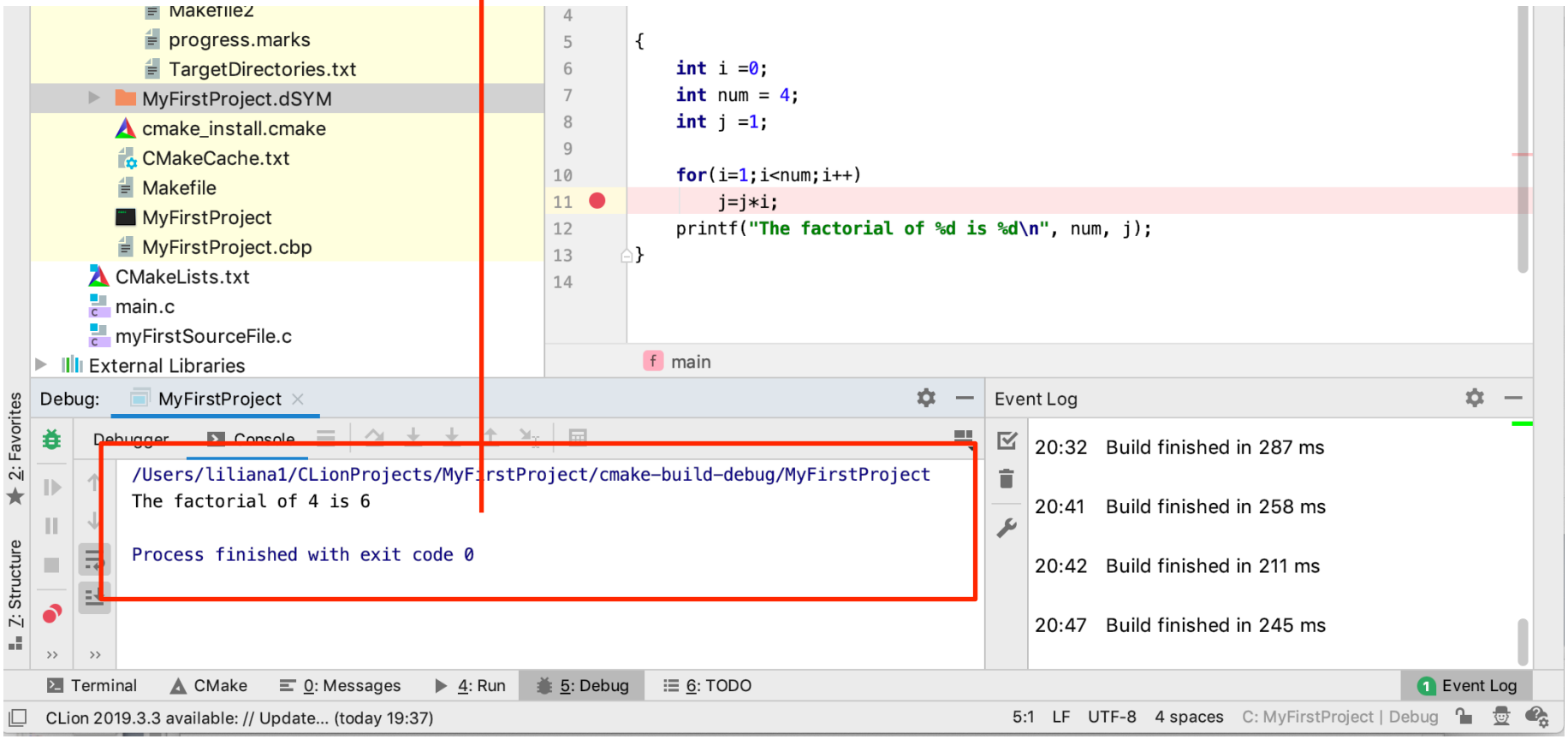
Debugging

- Press Run Program



Debugging

- This result is wrong.
- The program returns the factorial of 3 (the number preceding 4) instead of the factorial of 4!



Correct Program

- Correct the line of code inside the red square to calculate the factorial of a number correctly.

```c
#include <stdio.h>

int main(int argc, char **argv)

{
    int i =0;
    int num = 4;
    int j =1;

    for(i=1;i<=num;i++)
        j=j*i;
    printf("The factorial of %d is %d\n", num, j);
}
```