

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/>

The screenshot shows a web browser window with the JetBrains logo and the text "For Students: Free Professional". The address bar contains "jetbrains.com/student/". Below the header, there's a navigation menu with "Store" selected. To the right are links for "JetBrains Toolbox", "Team Tools", "Plugins", "For Business", and "Contact Sales". A red box highlights the "APPLY NOW" button in a blue rounded rectangle.

JB For Students: Free Professional X +

← → C 🔒 jetbrains.com/student/ ☆

Store JetBrains Toolbox Team Tools Plugins For Business Contact Sales

Not sure if you're eligible? Check out our [FAQs](#).

How do I apply?

There are a few ways you can apply for a free license. The best way is to use your official institutional email address or ISIC card. In this case, it will only take you a few minutes to get a free educational license pack.

You can also apply with a student/teacher card or any other official document certifying your affiliation with your academic institution. Such applications may take several days to process.

Before applying, make sure to read the [License Terms](#) carefully.

APPLY NOW

Obtain CLion

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

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

The screenshot shows a web browser window with the URL [jetbrains.com/shop/eform/students](https://www.jetbrains.com/shop/eform/students). The page title is "JetBrains Products for Learning". The form is titled "Apply with:" and includes tabs for "UNIVERSITY EMAIL ADDRESS" (which is selected), "ISIC/TIC MEMBERSHIP", "OFFICIAL DOCUMENT", and "GITHUB".
The "Status" section has two radio buttons: "I'm a student" (selected) and "I'm a teacher".
The "Level of study" dropdown is set to "Undergraduate".
The "Is Computer Science or Engineering your major field of study?" section has two radio buttons: "Yes" (selected) and "No".
The "Graduation date" input field contains "Dec 31, 2022".
The "Email address" input field contains "fanny.riveraortiz@ucdconnect.ie". A small note below it states: "I certify that the university email address provided above is valid and belongs to me."
The "Country / region" dropdown is set to "Mexico".
At the bottom, there are two checkboxes:

- I am under 13 years old
- I have read and I accept the [JetBrains Account Agreement](#)
- I consent to the use of my name, email address, and location data in email communication concerning JetBrains products held or services used by me or my organization [More](#)

Obtain CLion

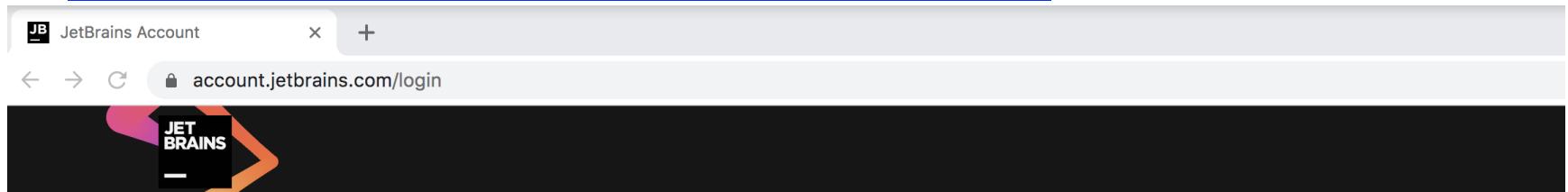
- You will get a confirmation email to register a JetBrains Educational Pack.
- Click on the Confirm Request Link.

The screenshot shows a Gmail inbox with the following details:

- Subject:** JetBrains Educational Pack Confirmation
- From:** JetBrains Account <no_reply@jetbrains.com>
- Date:** Jan 22, 2020, 2:17 PM (1 day ago)
- Message Preview:** Hi,
You've received this email because your email address was used for registering/updating a JetBrains Educational Pack. You can link JetBrains products to another email address in another step.
Please follow this link to confirm your decision and start using JetBrains Educational Pack for free:
[Confirm Request](#)
- Signature:** Yours truly,
The JetBrains Team
<https://www.jetbrains.com>
The Drive to Develop

Obtain CLion

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



Welcome to JetBrains Account



Access your purchases
and view your order history



Identify expired and outdated licenses,
order new licenses and upgrades



Manage your company licenses
and distribute them to end users

Sign in with existing account

Sign In

[Forgot password?](#)

Not registered yet?

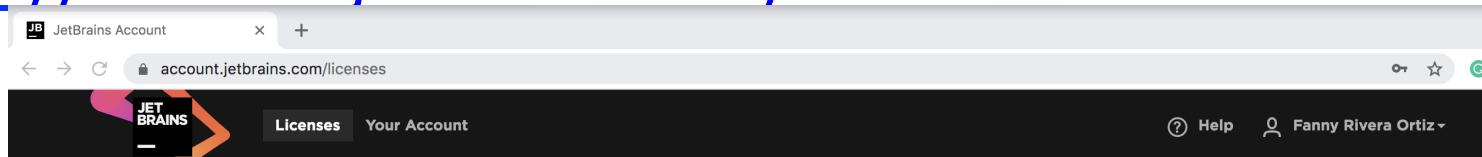
Create JetBrains Account

Sign Up

Obtain CLion

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

<https://account.jetbrains.com/licenses>



The screenshot shows a '1 License' summary with a 'Buy new license' button. Below it is a detailed product card for the 'JetBrains Product Pack for Students' with a red border. The card displays the 'License ID: HEYUOO7QPW'. It lists the following details:
Licensed to: Fanny Rivera Ortiz
License: For educational use only
restriction:
Valid through: January 21, 2021
Following products included:

- AppCode
- CLion
- DataGrip
- dotCover
- dotMemory
- dotTrace
- GoLand
- IntelliJ IDEA Ultimate
- PhpStorm
- PyCharm
- ReSharper
- ReSharper C++
- Rider
- RubyMine
- WebStorm

After downloading and installing the software, simply run it and follow the on-screen prompts to sign in with your JetBrains Account.

Obtain CLion

- Click on CLion.

The screenshot shows a web browser window for the JetBrains Account at account.jetbrains.com/licenses. The user is Fanny Rivera Ortiz. A green notification bar at the top right says "Two-factor authentication is available! To enable an extra layer of security for your JetBrains account, turn on two-factor auth." Below the navigation bar, there's a sidebar with "Fanny Rivera Ortiz" and links for "Transactions / Invoices" and "Notifications". The main content area has a title "1 License" and a "Buy new license" button. It displays the "JetBrains Product Pack for Students" with a "Download" link and a "License ID: HEYUOO7QPW". The license details show it's licensed to Fanny Rivera Ortiz for educational use only, valid through January 21, 2021. The included products list includes AppCode, CLion (which is highlighted with a red box), DataGrip, dotCover, dotMemory, dotTrace, GoLand, IntelliJ IDEA Ultimate, PhpStorm, ReShaper, ReSharper C++, Rider, RubyMine, and WebStorm. A note at the bottom says to run the software and sign in with the JetBrains Account.

JB JetBrains Account

account.jetbrains.com/licenses

Licenses Your Account Help Fanny Rivera Ortiz

Fanny Rivera Ortiz

Transactions / Invoices

Notifications

Two-factor authentication is available!
To enable an extra layer of security for your JetBrains account, turn on two-factor auth.

1 License

Buy new license

JetBrains Product Pack for Students

Download

License ID:
HEYUOO7QPW

Licensed to: Fanny Rivera Ortiz

License: For educational use only

restriction:

Valid through: January 21, 2021

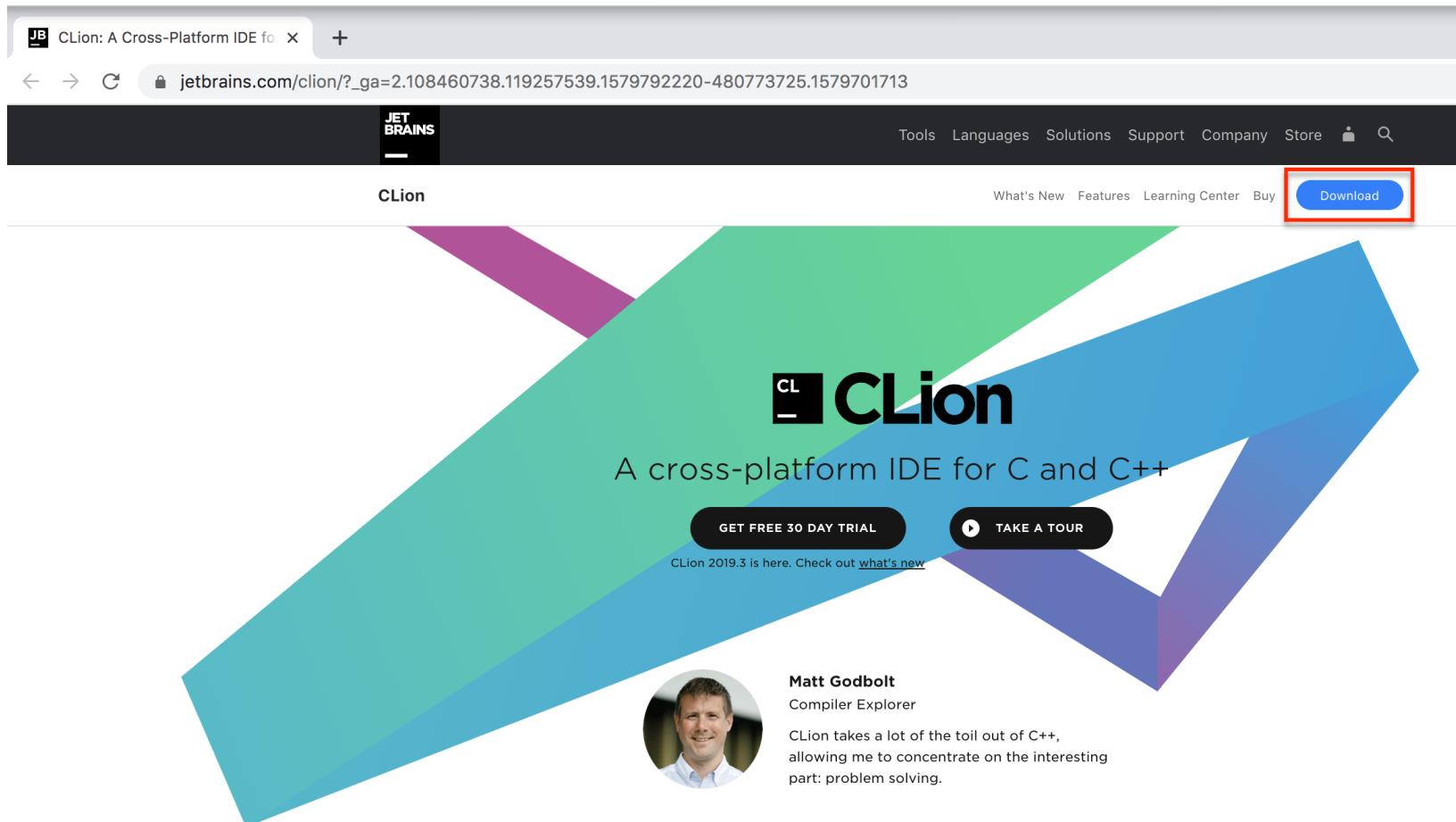
Following products included:

- AppCode
- CLion
- DataGrip
- dotCover
- dotMemory
- dotTrace
- GoLand
- IntelliJ IDEA Ultimate
- PhpStorm
- ReSharper
- ReSharper C++
- Rider
- RubyMine
- WebStorm

After downloading and installing the software, simply run it and follow the on-screen prompts to sign in with your JetBrains Account.

Install CLion

- Click on the Download button



Install CLion

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

The screenshot shows the CLion download page on the JetBrains website. At the top, there's a navigation bar with links for Tools, Languages, Solutions, Support, Company, Store, and a search icon. Below the navigation bar, there's a main menu with links for What's New, Features, Learning Center, and Buy, along with a prominent blue 'Download' button.

On the left side of the page, there's a large CLion logo icon and some version information: Version: 2019.3.3, Build: 193.6015.37, and Date: 21 January 2020. Below this, there are links for System requirements, Installation Instructions, Third-party software, Other versions, and Known issues.

The central part of the page features a 'Download CLion' section with three tabs: Windows, Mac (which is selected), and Linux. A red box highlights the 'Mac' tab. Below the tabs, it says 'CLion includes an evaluation license key for a **free 30-day trial**'. A red box highlights the blue 'Download' button. At the bottom, there's a callout for the Toolbox App with a red box highlighting the 'Get the Toolbox App to download CLion and its future updates with ease' text and icon.

Configure CLion

Toolchain Requirements

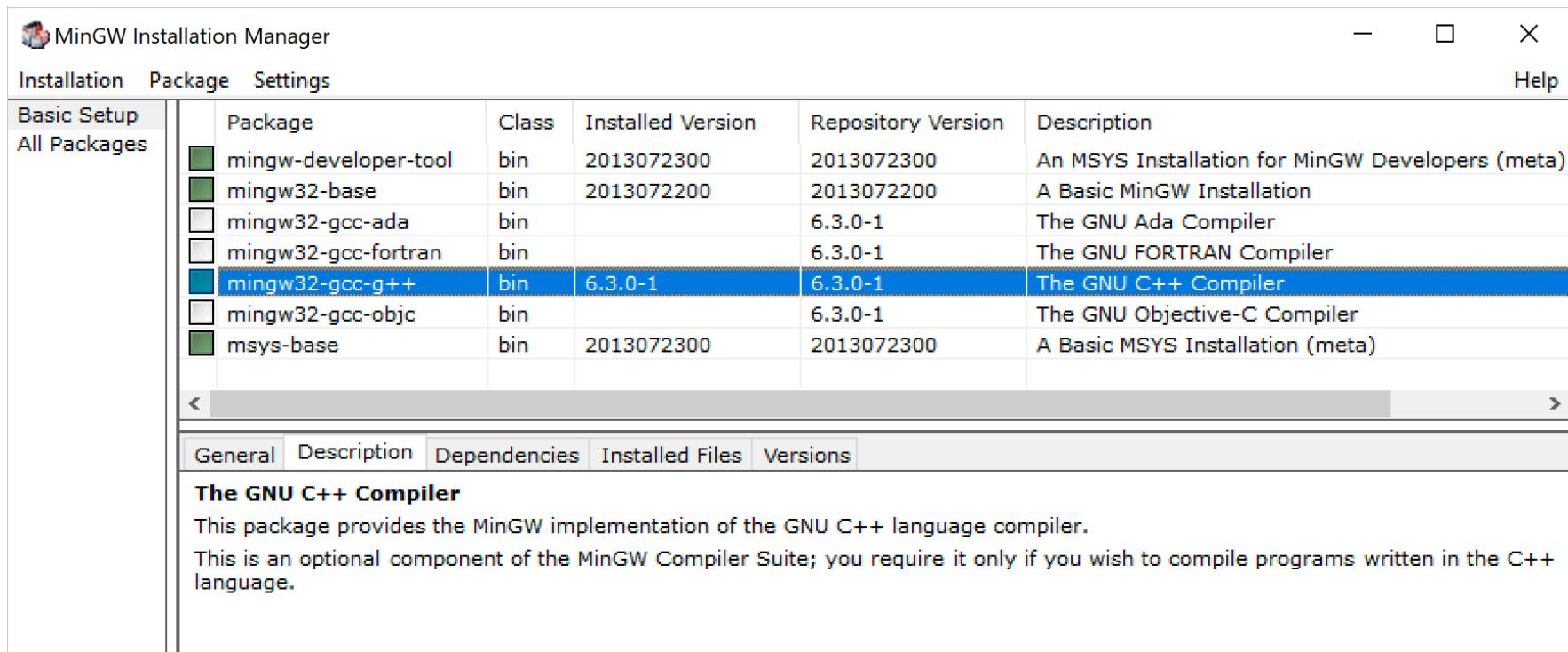
Ensure that you have one of these tools installed on your machine.

- **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

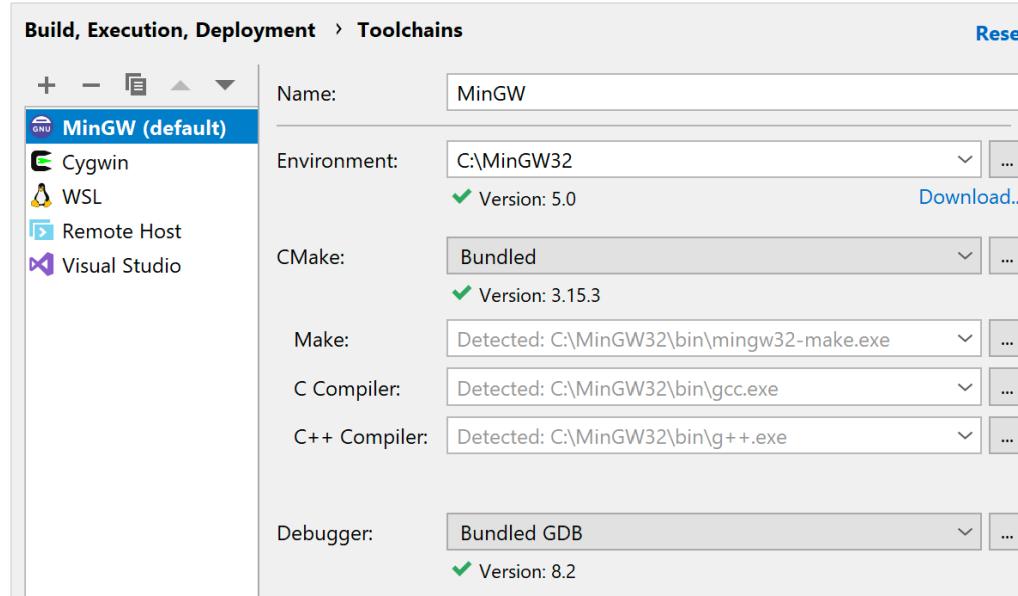
Installing MinGW/MingW64

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)

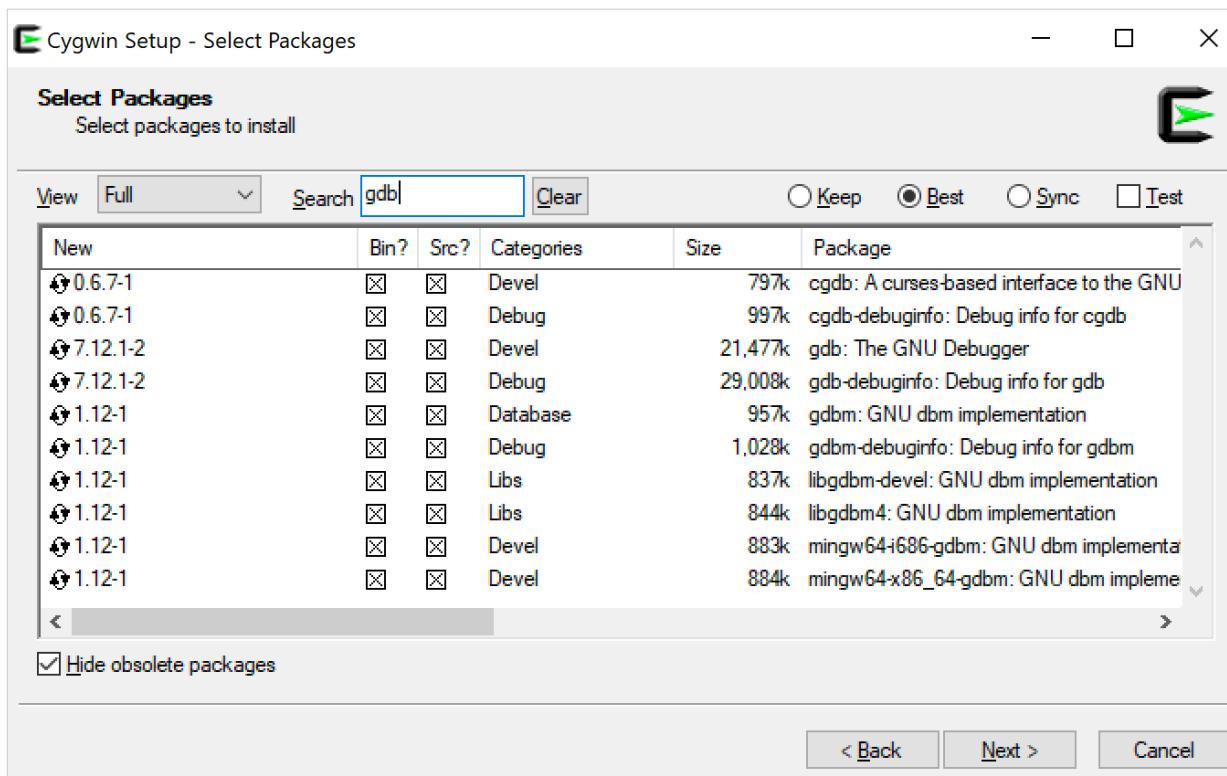
- Once the installation is finished, open CLion and go to File | Settings | Build, Execution, Deployment | Toolchains.
- Select MinGW from the Environment list. CLion attempts to detect the MinGW installation automatically. Check the detection result, and specify the path manually if required.
- Wait 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

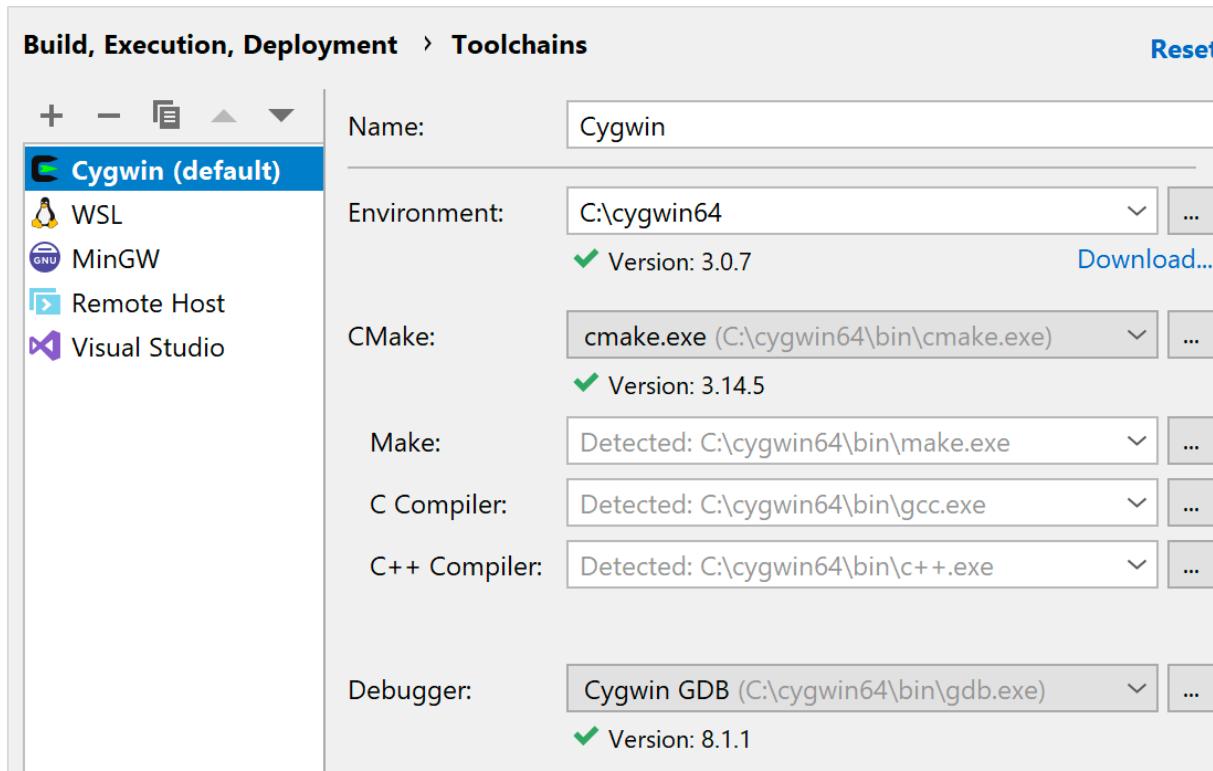
Install CygWin

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)

- Once the installation is finished, open CLion and go to **File | Settings | Build, Execution, Deployment | Toolchains**.
- 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.
- Wait until detection finishes, and press **Apply**.



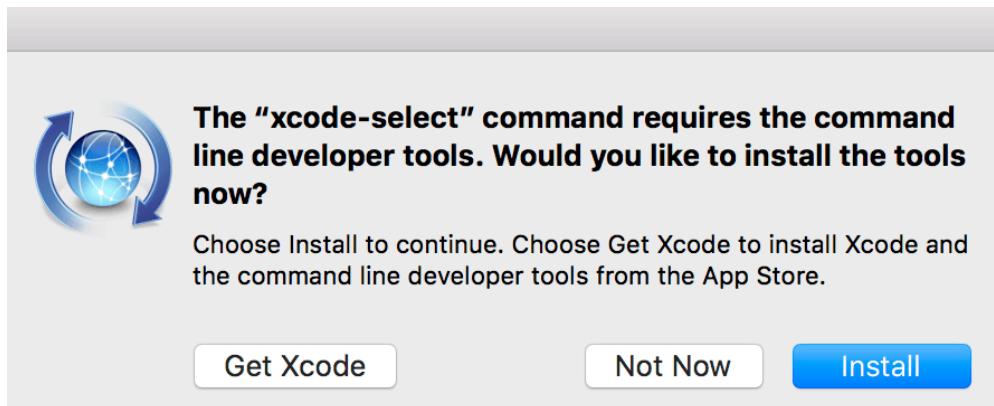
Configure CLion on MacOS using XCode

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)

- 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`



A screenshot of a macOS terminal window titled "fannyriveraortiz — bash — 108x24". The window shows the output of the command `clang --version`. The output includes the last login information, the command run, the Apple LLVM version (10.0.0), the target (x86_64-apple-darwin17.7.0), the thread model (posix), and the installed directory (/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin). The terminal has a dark theme with red, yellow, and green window controls.

```
Last login: Thu Jan 23 10:57:29 on console
[dhcp-c10182b8:~ fannyriveraortiz]$ clang --version
Apple LLVM version 10.0.0 (clang-1000.11.45.5)
Target: x86_64-apple-darwin17.7.0
Thread model: posix
InstalledDir: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin
dhcp-c10182b8:~ fannyriveraortiz$ 
```

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

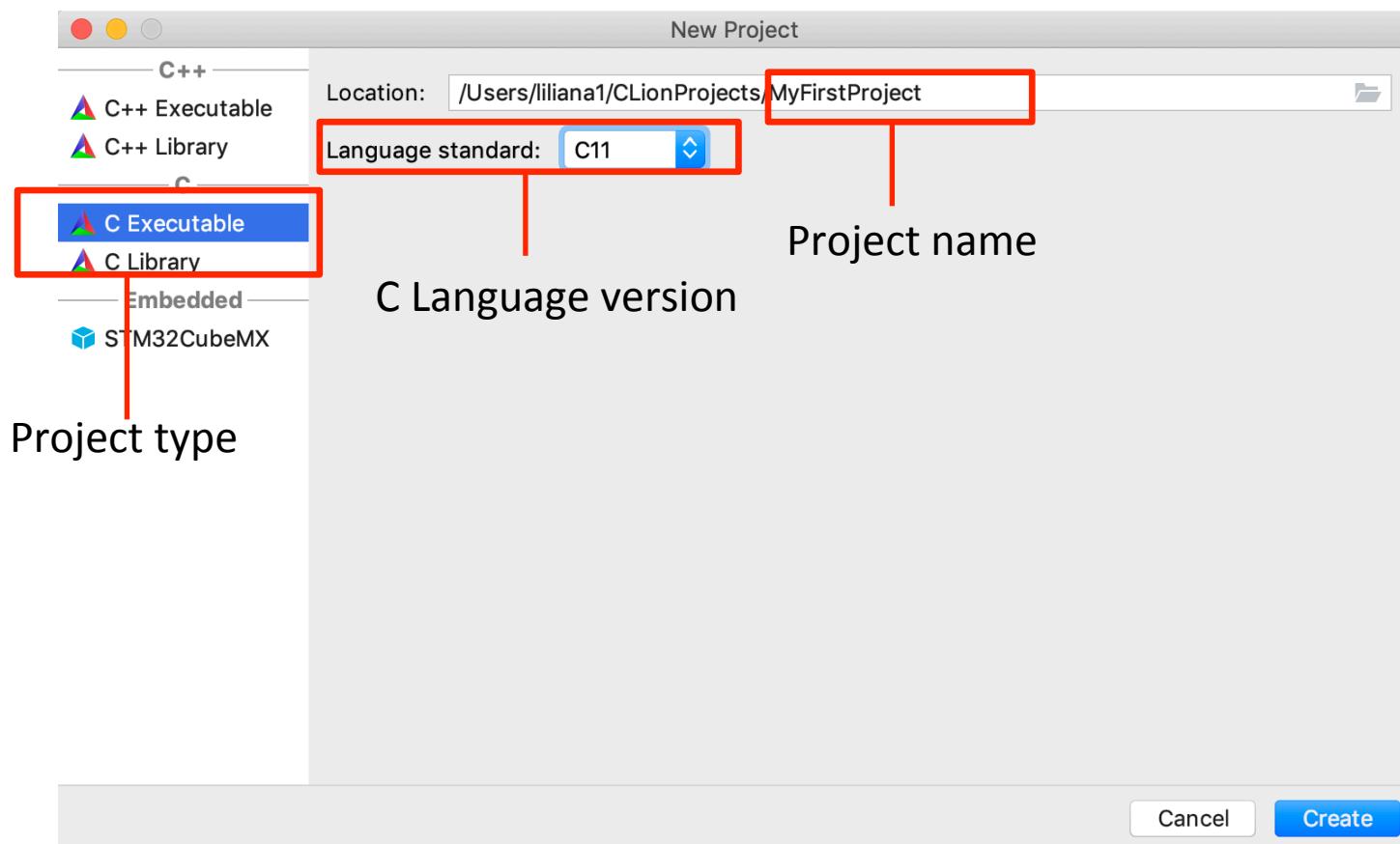
Projects

- 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

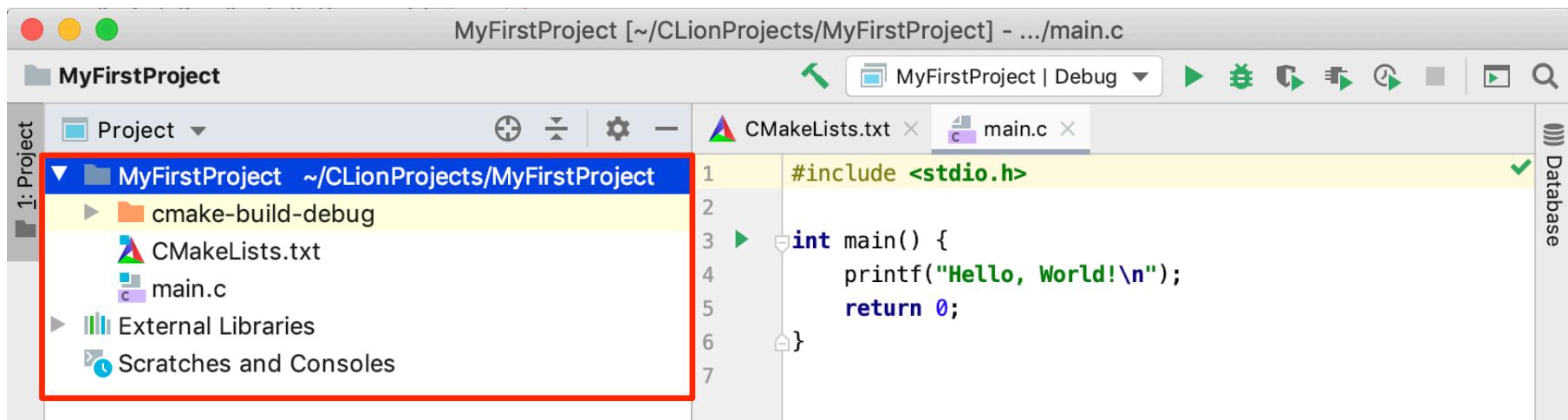
Create a New Project (1/2)

- 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

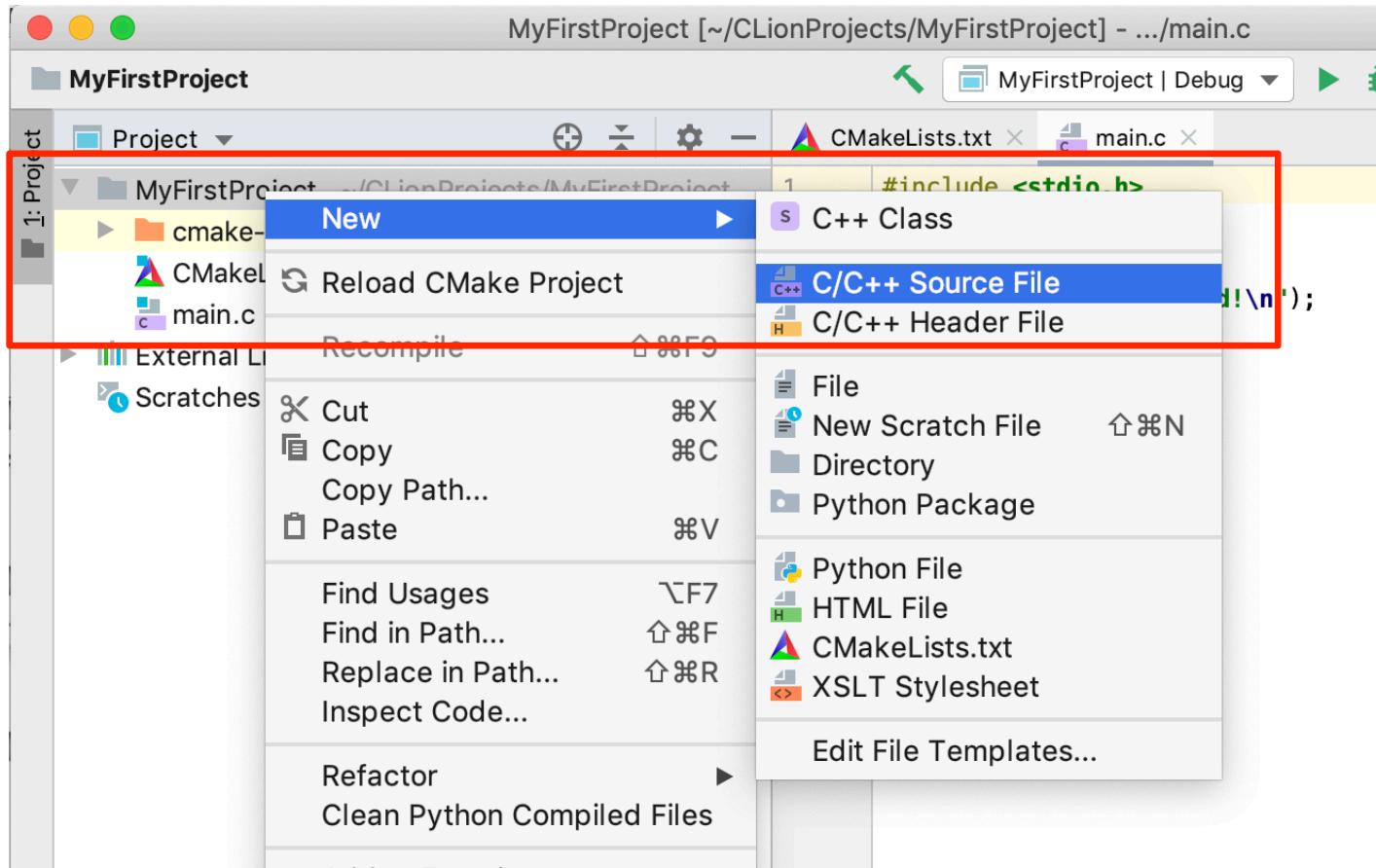


The screenshot shows the CLion IDE interface with the title bar "MyFirstProject [~/CLionProjects/MyFirstProject] - .../main.c". The left sidebar has a "Project" tab selected, showing the project structure under "1: Project". A red box highlights the "MyFirstProject" folder and its contents: "cmake-build-debug", "CMakeLists.txt", and "main.c". The main editor window shows the "main.c" file with the following code:

```
#include <stdio.h>
int main() {
    printf("Hello, World!\n");
    return 0;
}
```

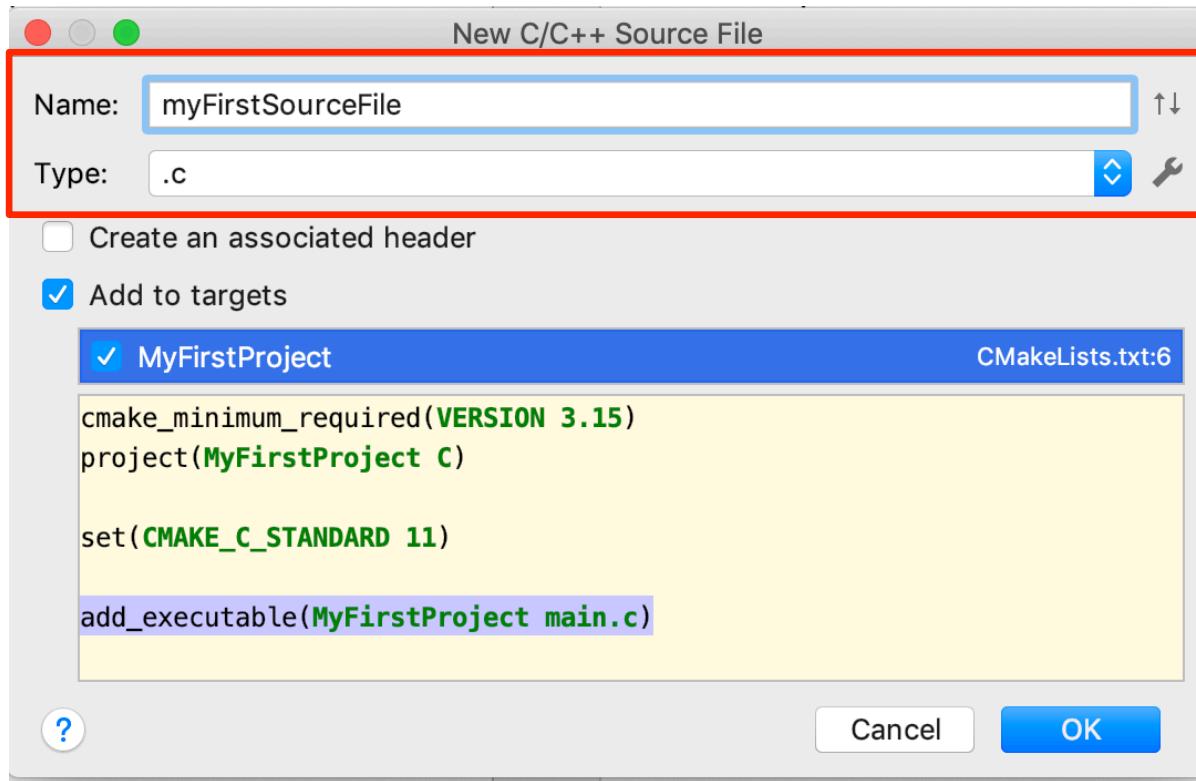
Add Files to Your Project

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

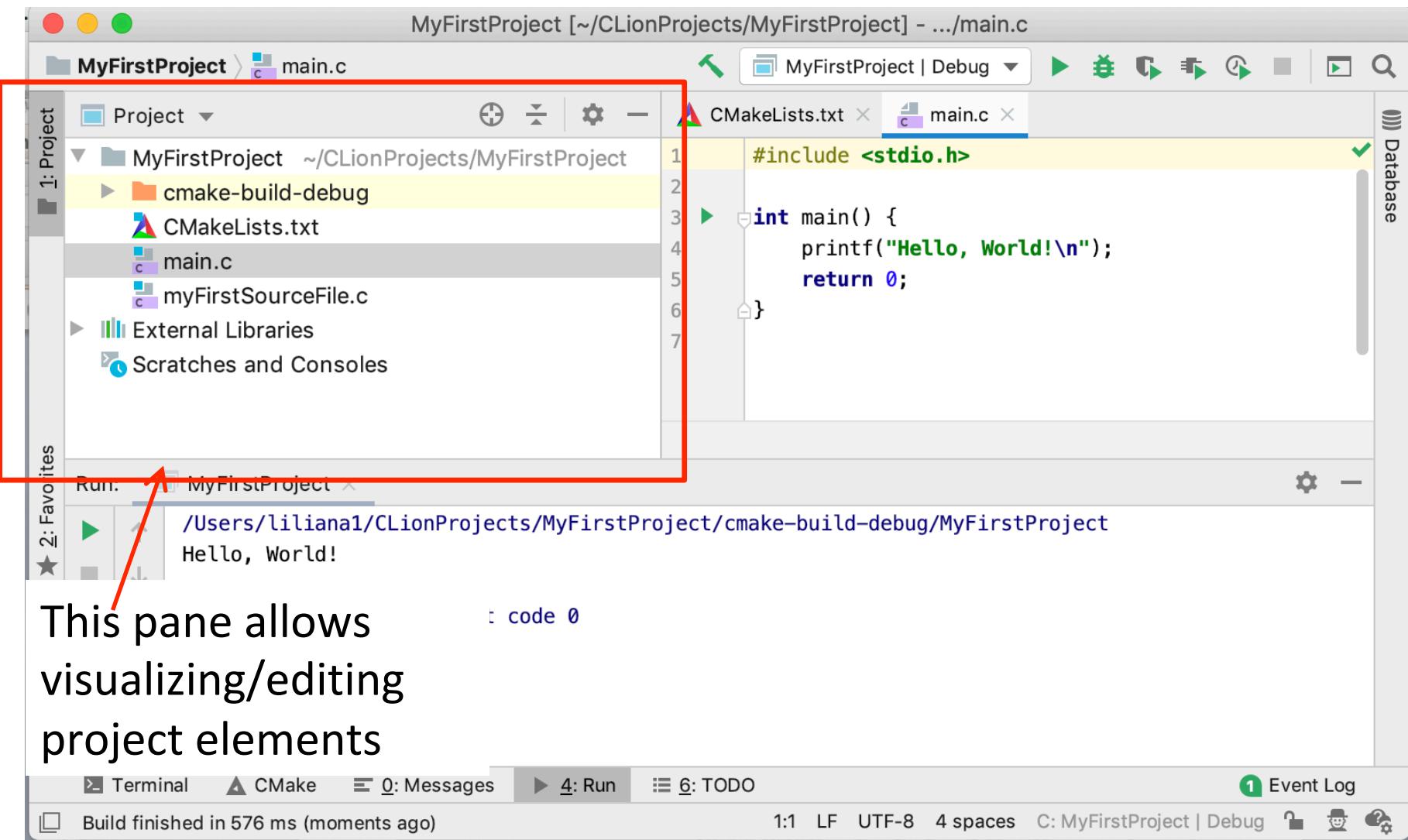
BrightSpace

Download Files

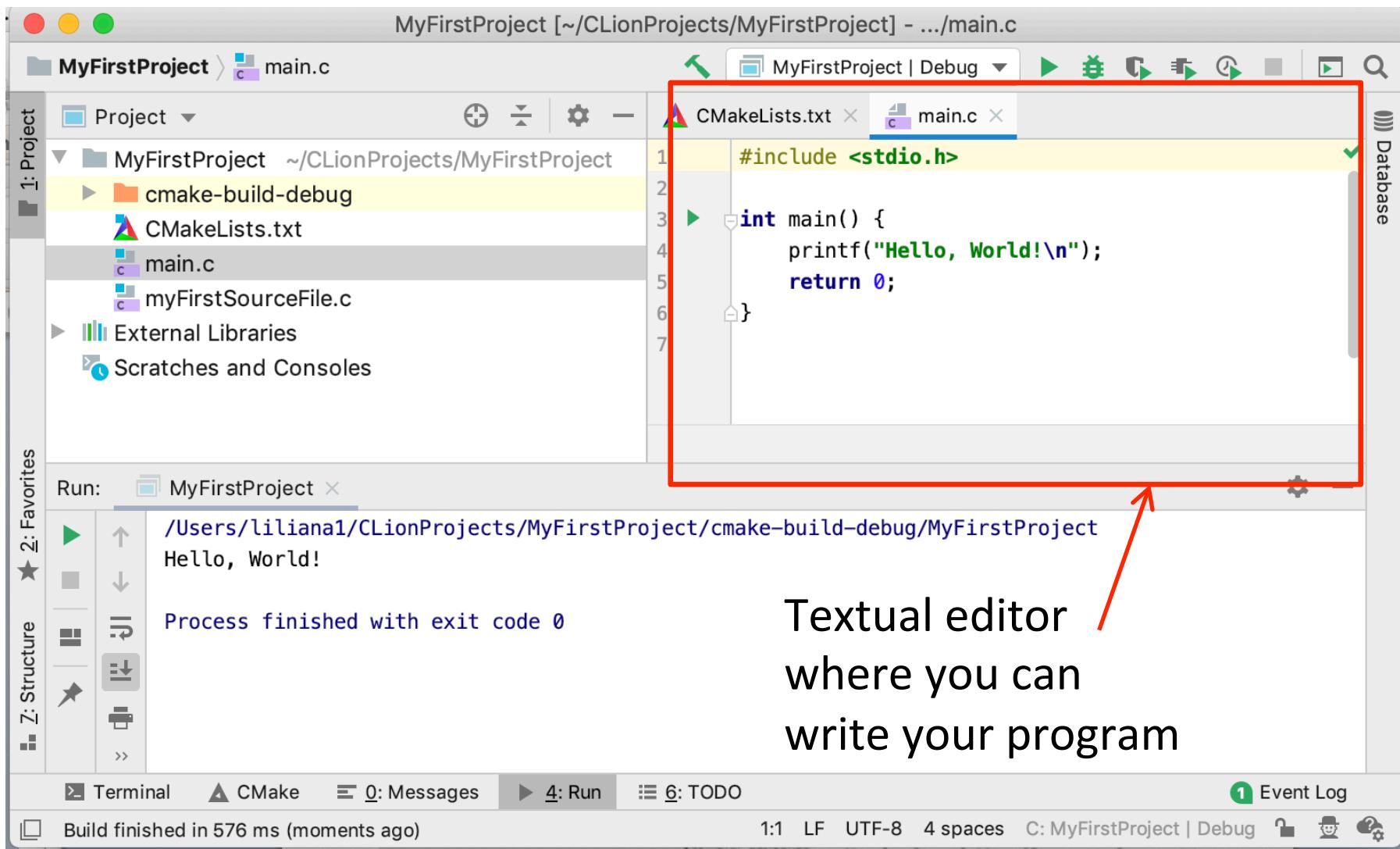
- Connect to UCD BrightSpace at
<https://brightspace.ucd.ie/d2l/home/54656>
- Download from the Week 1 >> Lab 1 the following files:
 - factorial.c
 - example1.c

CLion IDE Components

IDE Components



IDE Components



IDE Components

The screenshot shows the CLion IDE interface with the following components:

- Project Bar:** Displays the project name "MyFirstProject" and file "main.c".
- Code Editor:** Shows the main.c file content:#include <stdio.h>
int main() {
 printf("Hello, World!\n");
 return 0;
- Run Output:** Shows the terminal output for running the project "MyFirstProject". The output is:

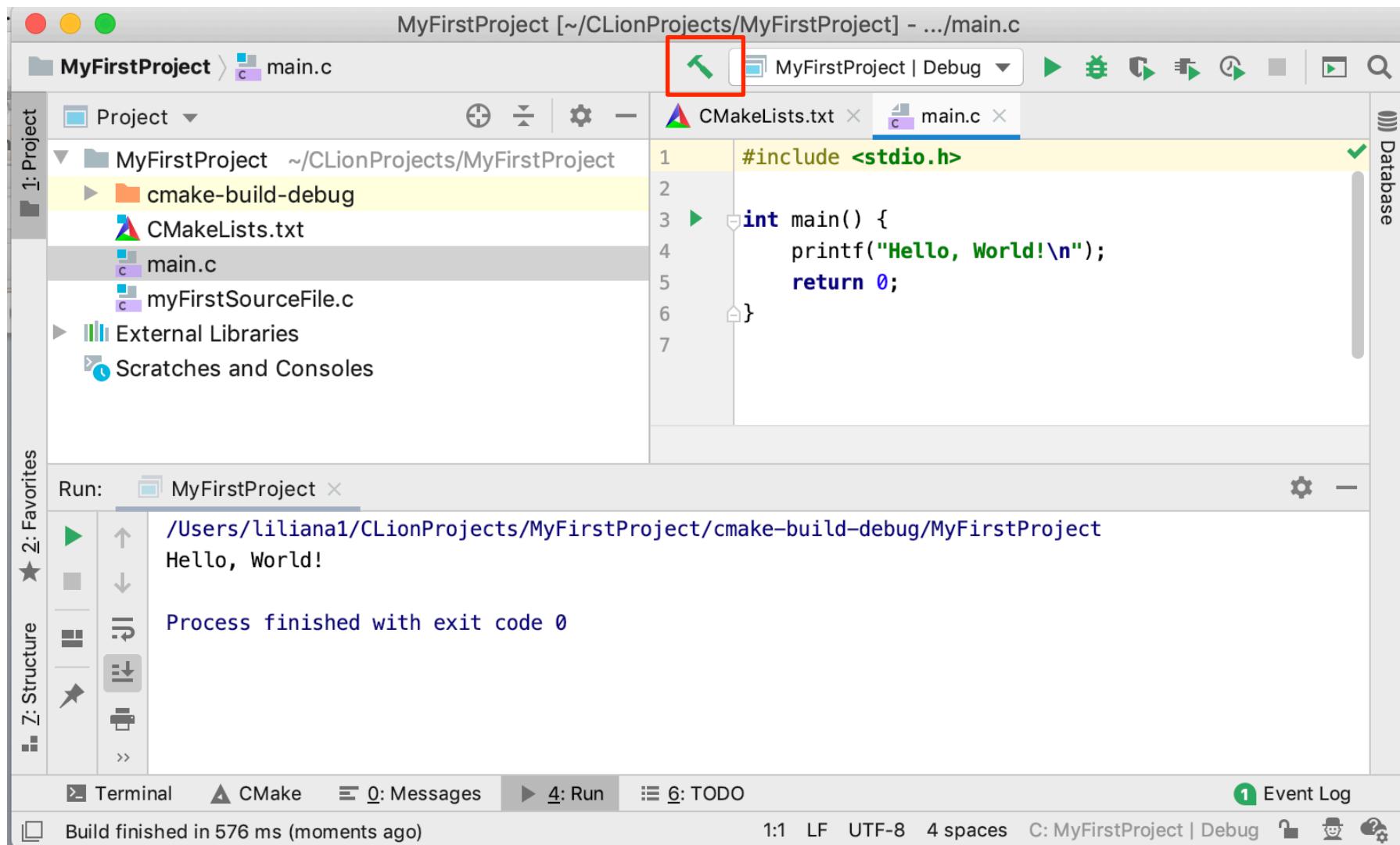
```
/Users/liliana1/CLionProjects/MyFirstProject/cmake-build-debug/MyFirstProject
Hello, World!
Process finished with exit code 0
```

A red box highlights this terminal area, and a red arrow points from the text "Typically this contains a console and a list of compiler problems" to the bottom of the red box.
- Bottom Status Bar:** Shows build status ("Build finished in 576 ms (moments ago)"), file encoding ("1:1 LF UTF-8 4 spaces"), and current directory ("C: MyFirstProject | Debug").

Typically this contains a console and a list of compiler problems

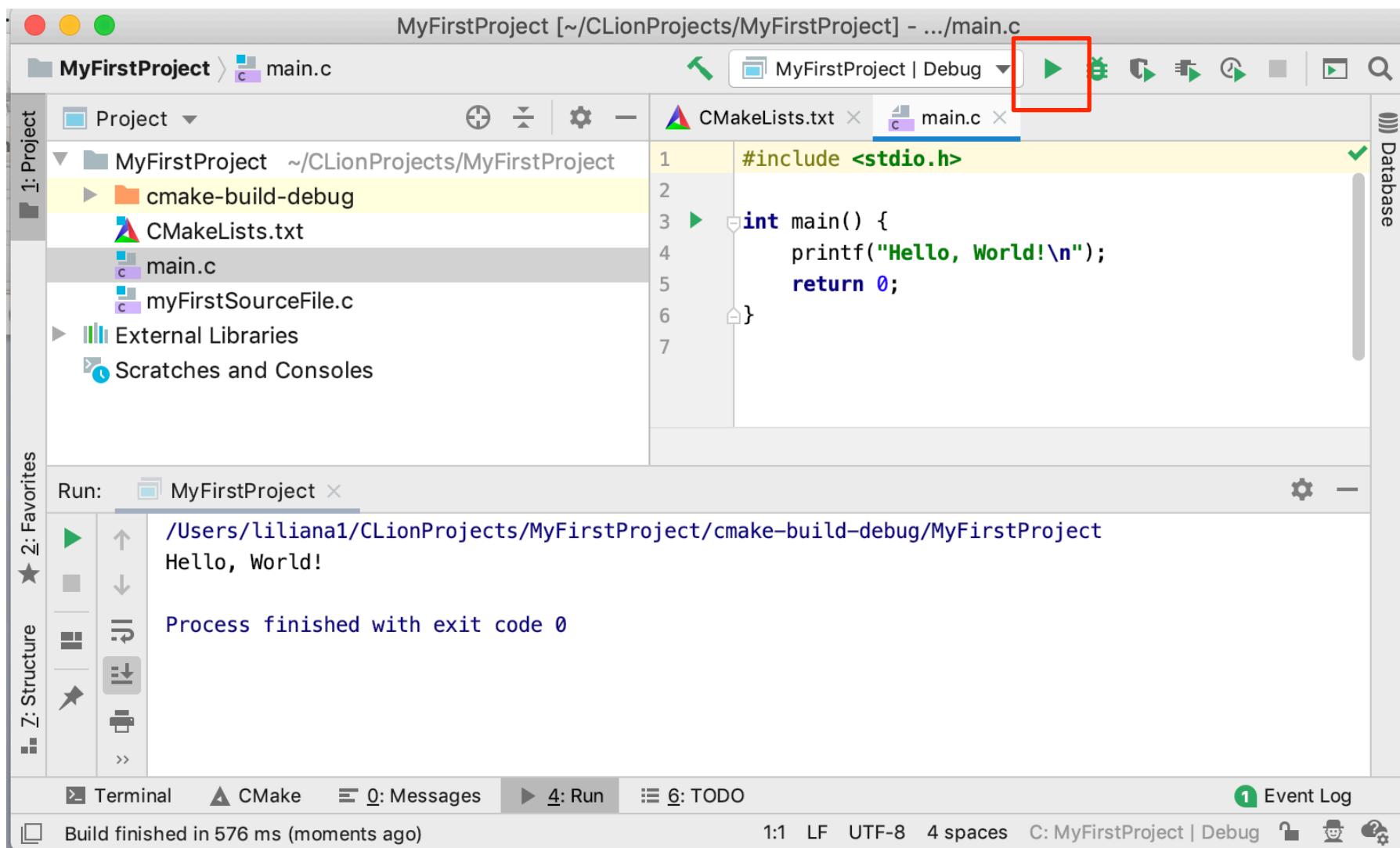
Compile file main.c

- 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

```
MyFirstProject [~/CLionProjects/MyFirstProject] - .../main.c
MyFirstProject > main.c
Project CMakeLists.txt main.c
MyFirstProject ~/CLionProjects/MyFirstProject
cmake-build-debug
CMakeLists.txt
main.c
External Libraries
Scratches and Consoles
Database
1:Project
2: Favorites
Run: MyFirstProject
/Users/liliana1/CLionProjects/MyFirstProject/cmake-build-debug/MyFirstProject
Enter 10 elements: 1 2 3 4 5 6 7 8 9 10
Standard Deviation = 2.872281
Process finished with exit code 0
Event Log
20:16 Build
finished in 392 ms
20:17 Build
finished in 332 ms
Terminal CMake Messages Run Debug TODO Event Log
Process finished with exit code 0
6:1 LF UTF-8 4 spaces C: MyFirstProject | Debug
```

The screenshot shows the CLion IDE interface with a project named "MyFirstProject". The "main.c" file is the active editor, containing the following code:

```
#include <stdio.h>
#include <math.h>

float calculateSD(float data[]);

int main()
{
    int i;
    float data[10];

    /*This is used to read 10 float numbers from the console
     and place them into array data */
    printf("Enter 10 elements: ");
    i = 10;
    for(i=0; i < 10; ++i)
        scanf("%f", &data[i]);
}
```

The "Run" tab at the bottom shows the command `/Users/liliana1/CLionProjects/MyFirstProject/cmake-build-debug/MyFirstProject` and the output `Standard Deviation = 2.872281`. The "Event Log" on the right shows build logs for two builds.

Debugging

Debugging

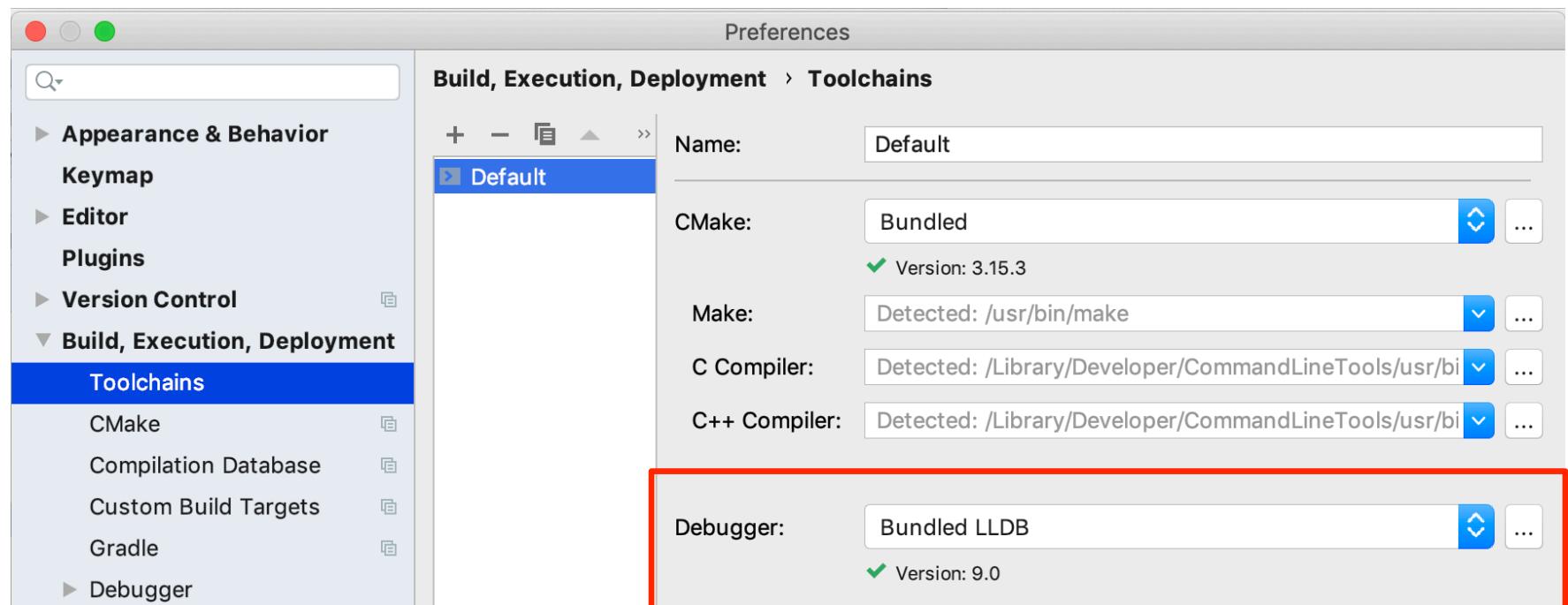
As your programs become more complicated, there will be a need to trace the program execution step by step or place *break points* where you wish the program to pause.

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

Prerequisites

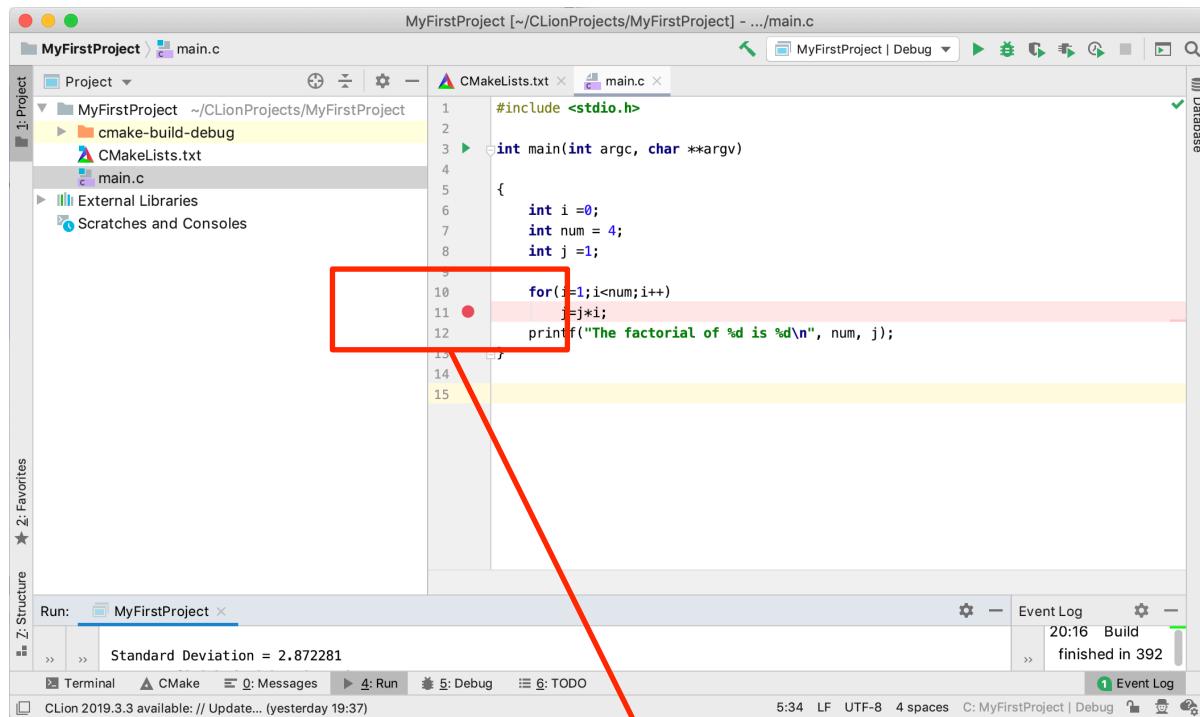
Ensure that a debugger is configured in your toolchains.

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.



The screenshot shows the CLion IDE interface with the following details:

- Project View:** Shows the project structure with "MyFirstProject" selected. Inside, there are files like "CMakeLists.txt", "main.c", and "cmake-build-debug".
- Code Editor:** Displays the "main.c" file content:

```
#include <stdio.h>
int main(int argc, char **argv)
{
    int i = 0;
    int num = 4;
    int j = 1;

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

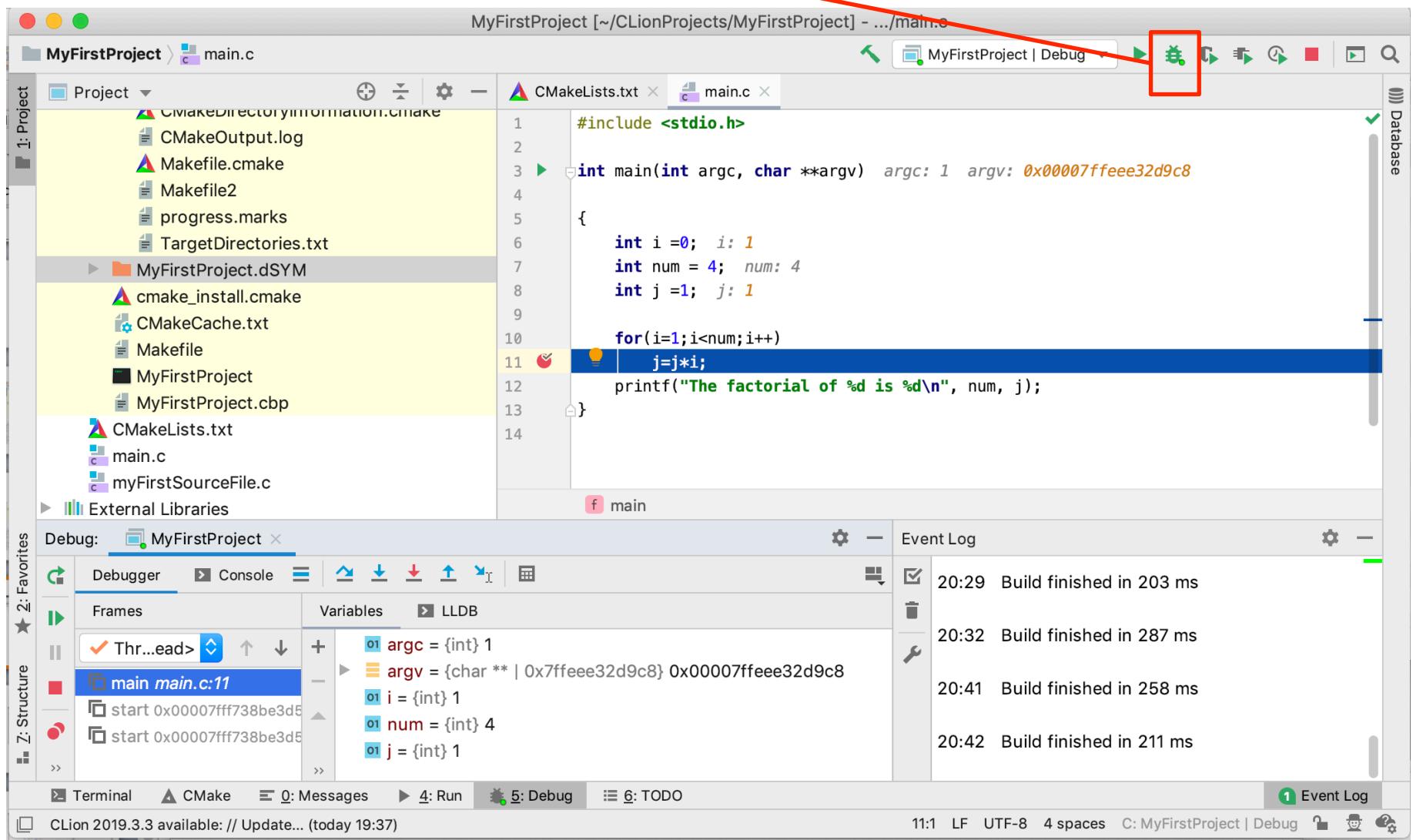
A red rectangle highlights the line "j=j*i;" (line 11). A red arrow points from this highlighted line to the left margin of the code editor, where a red circle marks the breakpoint location.
- Run Tab:** Shows the run configuration "MyFirstProject". It includes a message about standard deviation: "Standard Deviation = 2.872281".
- Event Log:** Shows the build status: "20:16 Build finished in 392".

- 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

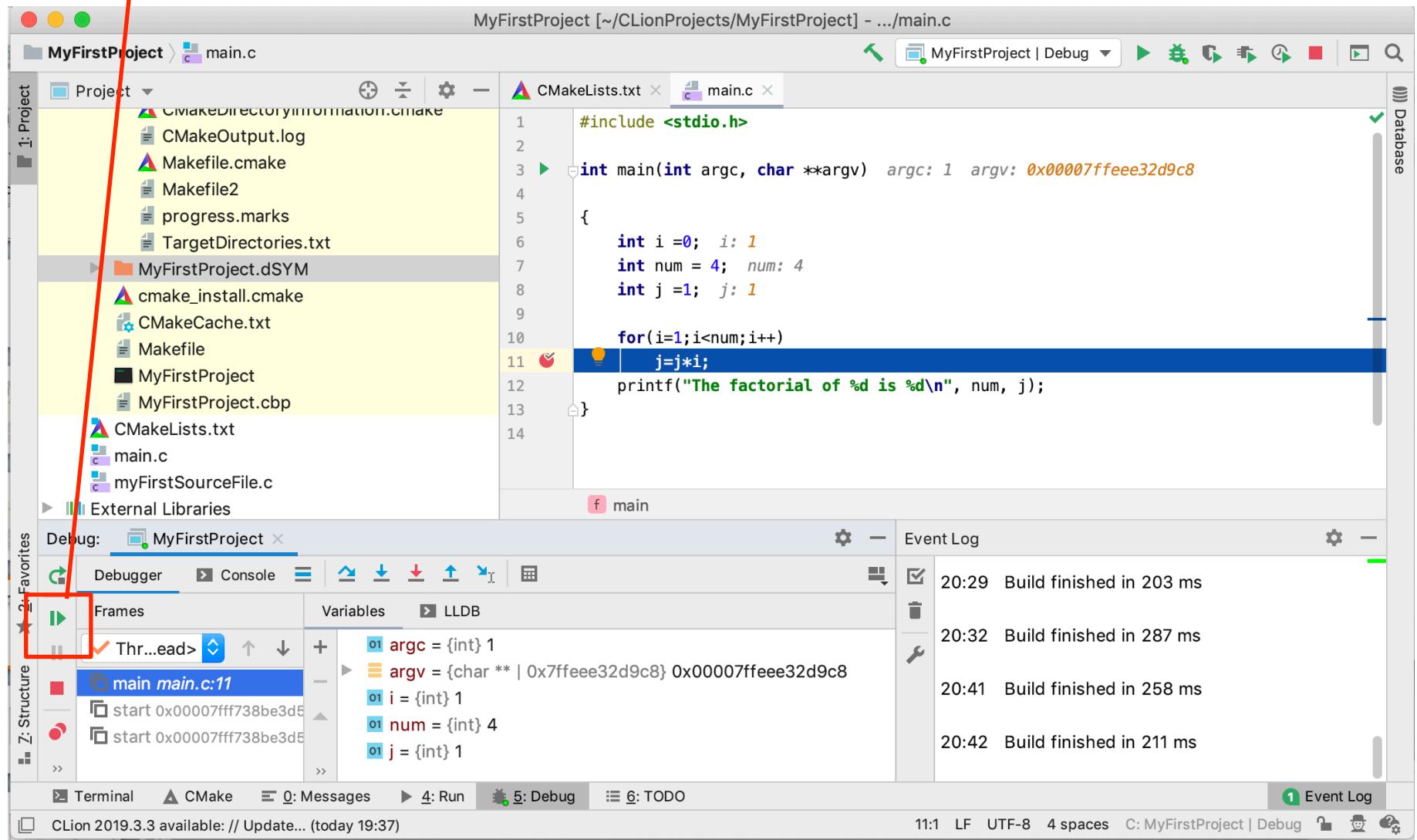
The screenshot shows the CLion IDE interface with the following details:

- Project View:** On the left, it shows the project structure with files like `progress.marks`, `TargetDirectories.txt`, and `MyFirstProject.dSYM`.
- Code Editor:** The main window displays a C code snippet:

```
5
6
7
8
9
10
11   int i = 0;  i: 1
12   int num = 4;  num: 4
13   int j = 1;  j: 1
14
15   for(i=1;i<num;i++)
16     j=j*i;
17   printf("The factorial of %d is %d\n", num, j);
18 }
```
- Debugger View:** At the bottom, the **Debugger** tab is selected. A red box highlights the **LLDB** tab in the bottom navigation bar.
- Variables View:** The Variables tab is active, showing local variables:
 - `argc = {int} 1`
 - `argv = {char **} 0x7ffeee32d9c8 0x00007ffeee32d9c8`
 - `i = {int} 1`
 - `num = {int} 4`
 - `j = {int} 1`
- Event Log:** On the right, the Event Log shows build logs:
 - 20:29 Build finished in 203 ms
 - 20:32 Build finished in 287 ms
 - 20:41 Build finished in 258 ms
 - 20:42 Build finished in 211 ms
- Bottom Navigation:** The navigation bar includes tabs for Terminal, CMake, Messages, Run, Debug (selected), TODO, and Event Log.

Debugging

- Press Resume Program



Debugging

The screenshot shows the CLion IDE interface during a debugging session of a C project named "MyFirstProject".

Project View: On the left, the project structure is shown with files like CMakeLists.txt, main.c, and myFirstSourceFile.c.

Code Editor: The main window displays the code for main.c. A red breakpoint is set on line 11, which contains the assignment `j=j*i;`. The code also includes a printf statement to output the factorial result.

Variables View: The Variables tool window (highlighted with a red box) shows the current values of variables:

- argc = {int} 1
- argv = {char **} 0x7ffef9069c8 0x00007ffef9069c8
- i = {int} 3
- num = {int} 4
- j = {int} 2

Event Log: The Event Log shows build logs with times and durations:

- 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

Bottom Navigation: The footer bar includes tabs for Terminal, CMake, Messages, Run, Debug (which is selected), TODO, and Event Log.

Debugging

- Press Resume Program

The screenshot shows the CLion IDE interface with a project named "MyFirstProject". The main editor window displays the file "main.c" containing C code for calculating factorials. A red arrow points from the text "Press Resume Program" to the "Resume" button in the debugger toolbar. The debugger toolbar also includes buttons for Stop, Step Into, Step Over, and Step Out. The "Frames" tab in the bottom-left shows the current stack frame "main main.c:11". The "Variables" tab lists local variables: argc (int) 1, argv (char**) 0x7ffef9069c8, i (int) 3, num (int) 4, and j (int) 2. The "Event Log" tab on the right shows build logs with times ranging from 20:32 to 20:47.

```
#include <stdio.h>
int main(int argc, char **argv) argc: 1 argv: 0x00007ffef9069c8
{
    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);
}
```

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

CLion 2019.3.3 available: // Update... (today 19:37)

Debugging

The screenshot shows the CLion IDE interface during a debugging session of a C project named "MyFirstProject".

Project View: On the left, the project structure is shown with files like CMakeLists.txt, main.c, and myFirstSourceFile.c. A file named "MyFirstProject.dSYM" is currently selected.

Code Editor: The main editor window displays the "main.c" source code. A breakpoint is set at line 11, which contains the assignment statement `j=j*i;`. The code includes a printf statement to output the factorial result.

Variables View: The bottom-left panel shows the current variable values in the LLDB debugger. A red box highlights this panel. The variables listed are `argc`, `argv`, `i`, `num`, and `j`.

Event Log: The bottom-right panel shows the build log with several entries indicating build completion times.

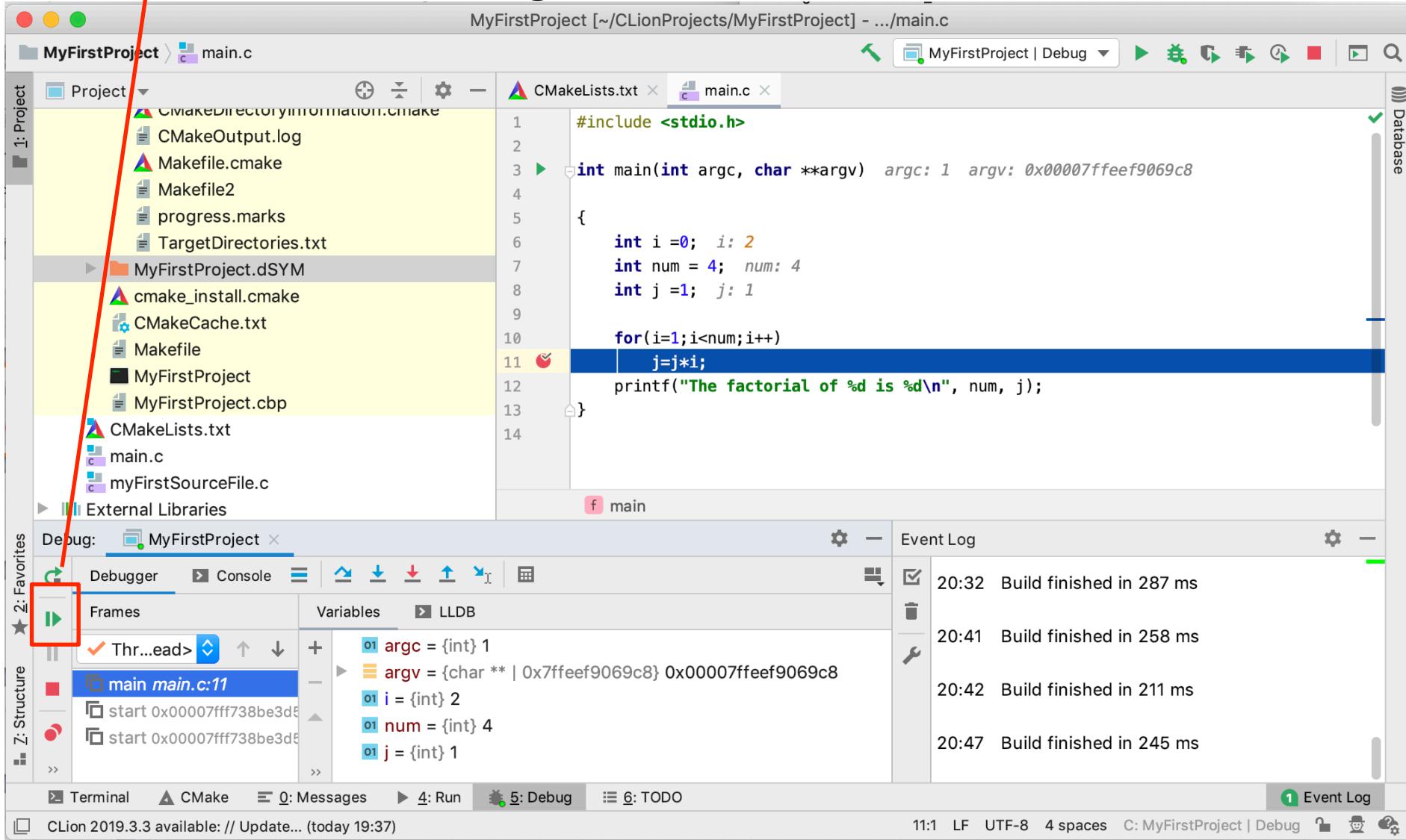
Bottom Navigation: The footer bar includes tabs for Terminal, CMake, Messages, Run, Debug (which is active), and TODO.

Status Bar: The status bar at the bottom right shows the current time (11:11), file encoding (LF), character width (4 spaces), and the current directory (C: MyFirstProject | Debug).

Time	Message
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

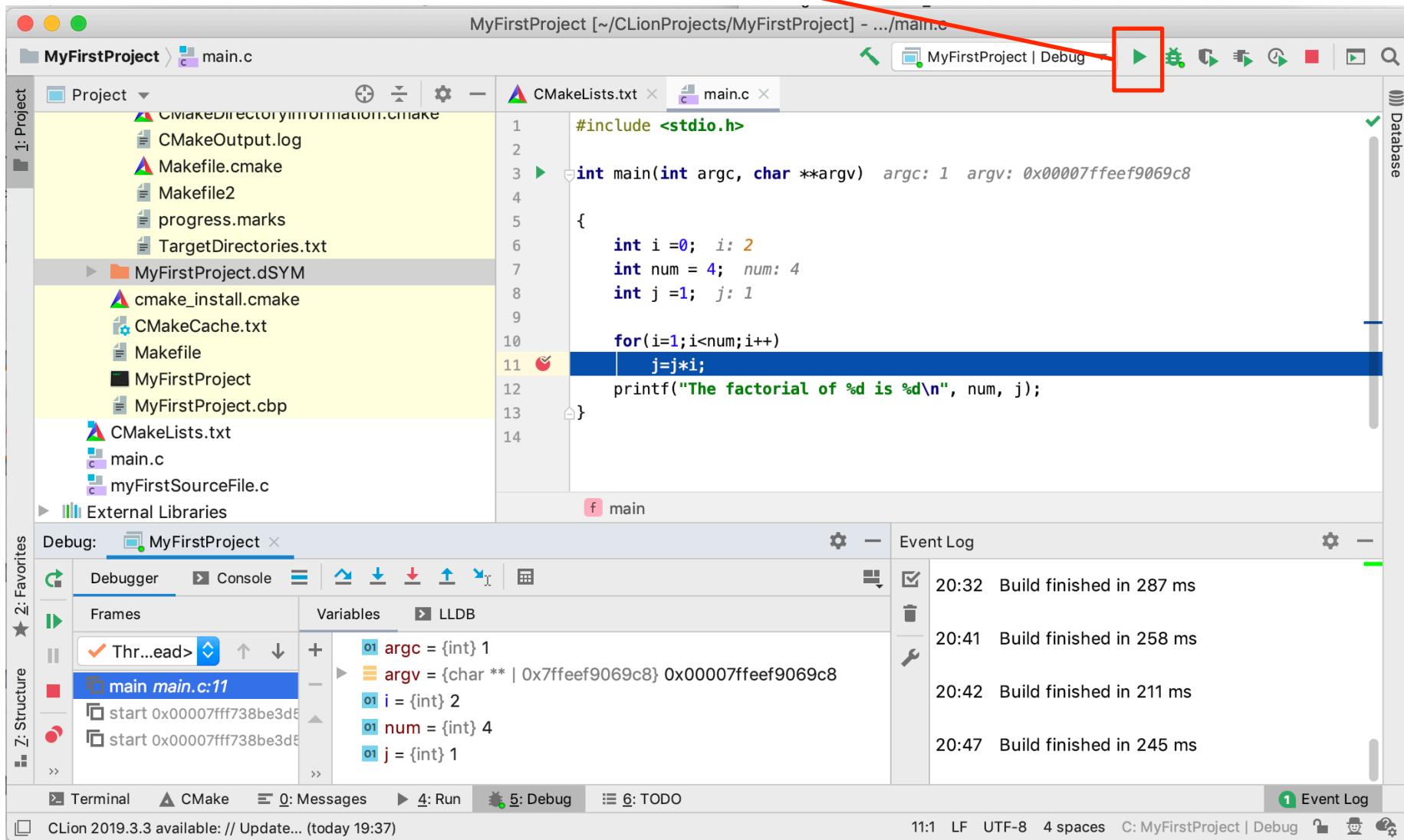
Debugging

- 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!

The screenshot shows the CLion IDE interface with the following details:

- Project Structure:** The left sidebar shows the project structure with files like `MyFirstProject.dSYM`, `cmake_install.cmake`, `CMakeCache.txt`, `Makefile`, `MyFirstProject`, `MyFirstProject.cbp`, `CMakeLists.txt`, `main.c`, and `myFirstSourceFile.c`.
- Code Editor:** The main editor window displays the `main.c` file with the following code:

```
4
5
6     int i =0;
7     int num = 4;
8     int j =1;
9
10    for(i=1;i<num;i++)
11        j=j*i;
12    printf("The factorial of %d is %d\n", num, j);
13
14 }
```

A red dot marks the current line at `j=j*i;`.
- Terminal:** The bottom-left terminal window shows the output of the debug session:

```
/Users/liliana1/CLionProjects/MyFirstProject/cmake-build-debug/MyFirstProject
The factorial of 4 is 6

Process finished with exit code 0
```

This output is highlighted with a red box.
- Event Log:** The bottom-right event log shows build logs:

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
- Bottom Navigation:** The footer includes tabs for Terminal, CMake, Messages, Run, Debug (which is selected), TODO, and Event Log (with 1 entry).

Correct Program

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

```
#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);
}
```