Financial Computing with C++ 1, Michaelmas Term - Getting started

The course involves reading, writing, compiling, debugging and running C++ code. For these purposes, you will need to download some files and install some tools. In particular, you will need to install

1) a (platform specific) C++ compiler and a (platform specific) C++ debugger,

2) an IDE (integrated development environment) for reading, writing code, running the compiler, running and/or debugging the compiled code

3) `CMake` - a tool that generates C++ projects specific to the installed compiler and the installed IDE (note: `CMake` is to be installed before installing the IDE).

The recommended and supported IDE is `CLion` that is available for Windows, Mac OS and Linux. This document contain some instructions on how to set up and use `CLion` .
On top of installing these tools, you need to download:

- the CppCourse.zip file that contains various sample projects and other libraries on which some of the practicals will be based,
- the `boost` header files.

All this may look a bit involved, but the setup has to be done only once.

1 Installing a C++ compiler and debugger

There are different combinations of compilers and debuggers for each of Windows, Mac OS and Linux. Here, one specific combination is covered for each of these operating systems.

1.1 C++ compiler and debugger for Windows

The install file for `mingw-w64` can be downloaded from here (follow the `sourceforge` link on the page):

`http://mingw-w64.org/doku.php/download/mingw-builds`

1.2 C++ compiler and debugger for Mac OS

`Xcode` is an IDE developed by Apple. It comes with the `clang` compiler and the `lldb` debugger. It can be installed from the App Store. Make sure that the clang version is 8.0.0 or later.
If you prefer to use the `gcc` compiler (version 5.5.0 or later), then you can install it using `MacPorts`. For installation instructions, follow this url:

`https://www.macports.org/`

1.3 C++ compiler and debugger for Linux

If not already installed, the `gcc` compiler (version 5.5.0 or later) and the `gdb` debugger can be installed using `apt-get`.

2 Installing `CMake`

`CMake` is a tool that automates the generation of IDE specific C++ projects from the source code and some configuration files. `CMake` is available for Windows, Mac OS, and Linux. You will need version 3.10 or later.
`CMake` can be installed from `CMake`'s website, or through command line. Follow this link to the website:

`https://cmake.org/download/`

When running the installer, it may ask if `CMake` is to be added to the `PATH`. Select yes.
Alternatively, you can install `CMake` through `MacPorts` on Mac and through `apt-get` on Linux.

3 Installing an IDE - `CLion`

`CLion` is a product of `JetBrains` and it requires licence. You can apply for a licence using your university e-mail address here:

`https://www.jetbrains.com/student/`

Once the `JetBrains` licence is sorted, you can install `CLion` from here.

`https://www.jetbrains.com/clion/`

More instructions on how to use `CLion` are included in section 5.

4 Downloading files

4.1 CppCourse.zip

The compressed file `CppCourse.zip` is available on the course website. Download and extract the file into a folder, that will be referred to as `[CppCourse]`. You are expected to see the following folder structure:

```
[CppCourse]
  -> [boostRoot]
  -> [Exam2019MT]
  -> [Lectures]
     -> [Lecture01]
     -> [Lecture010203]
     ...
  -> [Libraries]
     -> [DOPLib]
     -> [googletest]
     -> [MCLib]
     ...
  -> [Practicals]
  -> [SampleExam2015]
```

Some of these folders (for instance `[Exam2019MT]` and `[Practicals]`) are empty for now; files will be released in due course that you will add to these folders.

4.2 Boost header files

Some of the example and practical projects use features of the `boost` library. The main website of `boost` is

`http://www.boost.org/`

Click on the "download" link and look for the latest version of `boost` library (something like `boost_1_70_0.zip` or any other zipped version that you can unzip). What you need from the zip file is the `boost` folder; you do not need to extract the other folders.
Unzip the boost folder under `[CppCourse]\[boostRoot]`. You are expected to create this file structure:

```
[CppCourse]
  -> [boostRoot]
     -> [boost]
        -> [accumulators]
        -> [algorithm]
        ...
  ...
```

5 Using CLion

5.1 Initial setup

If you have installed the required components (the compiler, the debugger, and `CMake`), `CLion` should recognise them. You can check it in the `Preferences` (on a Mac) or in th `Settings` (on Windows) under `Toolchain`; see figure 1 for what toolchain you may get on a Mac, and see figure 2 for what toolchainyou may get on windows.

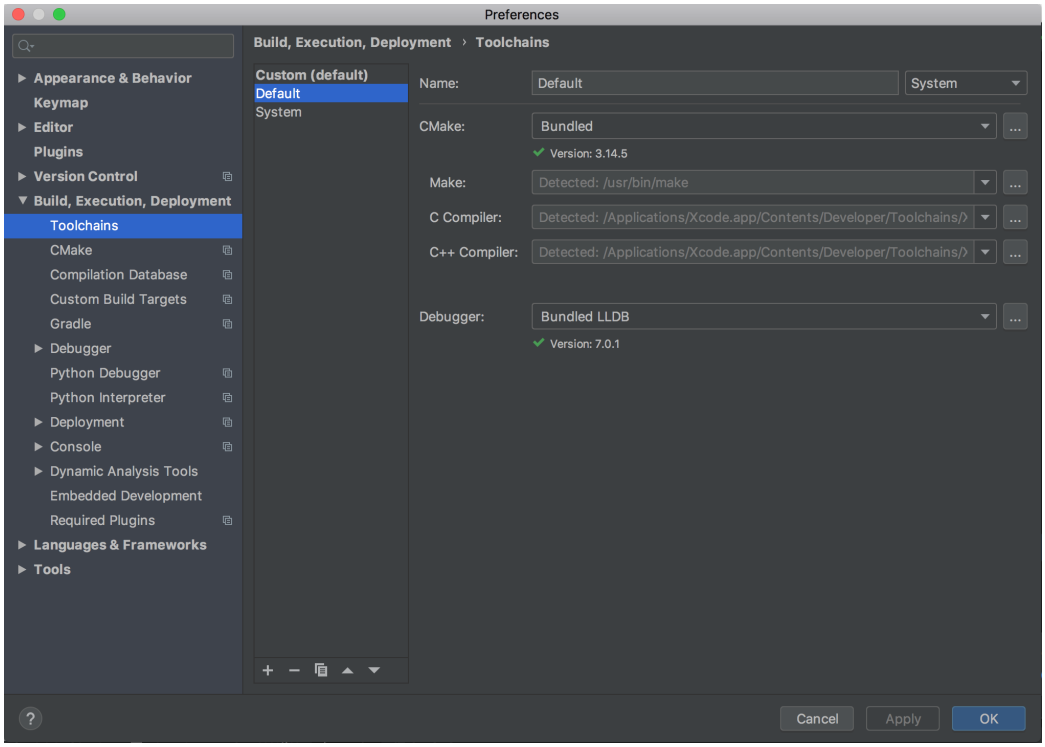

Figure 1: Toolchain on Mac

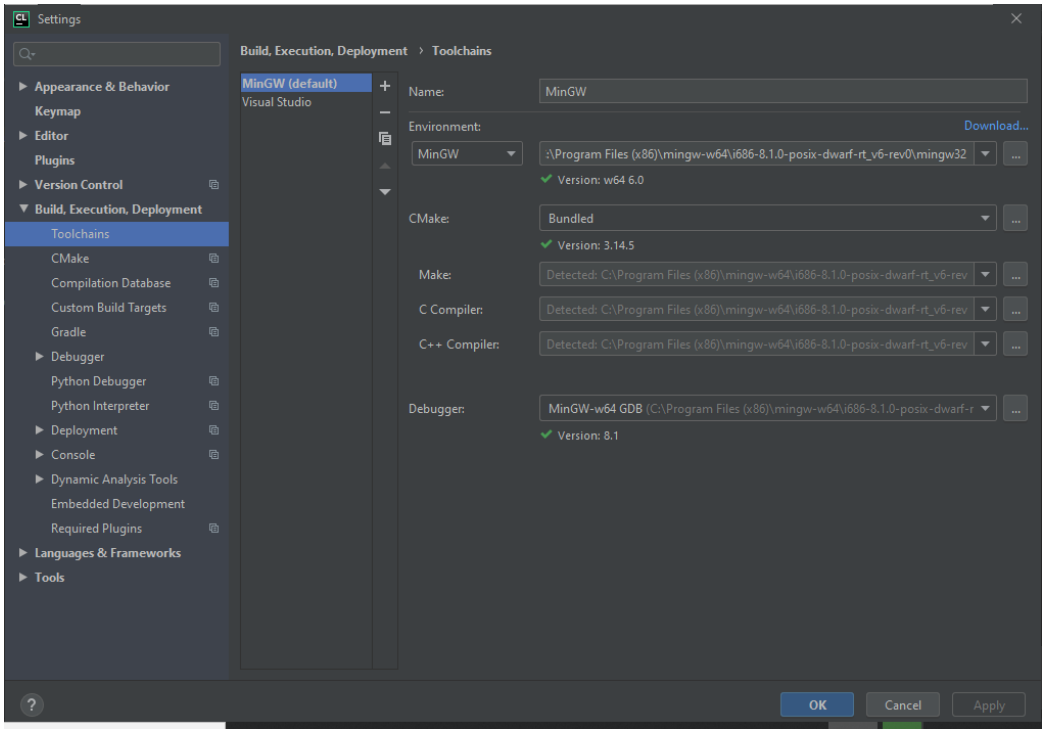

Figure 2: Toolchain on Windows

5.2 Running `CMake`

When first opening the course material (`File->Open`, navigate to and select the `CppCourse` folder), or when adding a new practical or sample exam or exam project, or when adding a new `.cpp` file to write your code into, you have to run `CMake` by right clicking on the `CppCourse` folder in the project view of `CLion` and select "Reload `CMake` Project" (see figure 3).
The output of the `CMake` run appears in the `CMake` -view at the bottom of the screen (see figure 4).

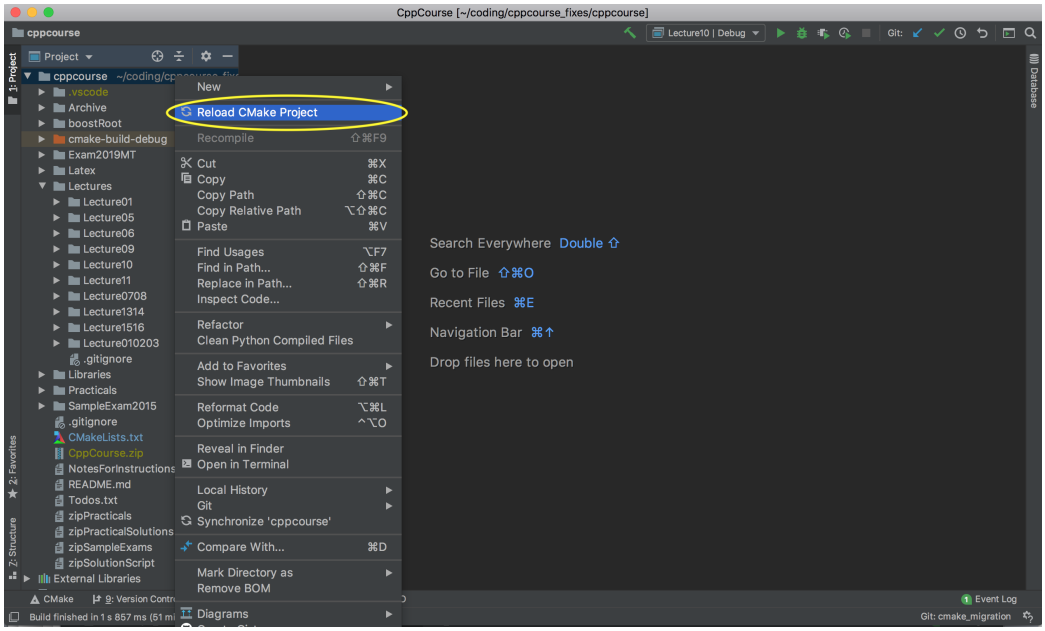

Figure 3: Running `CMake`

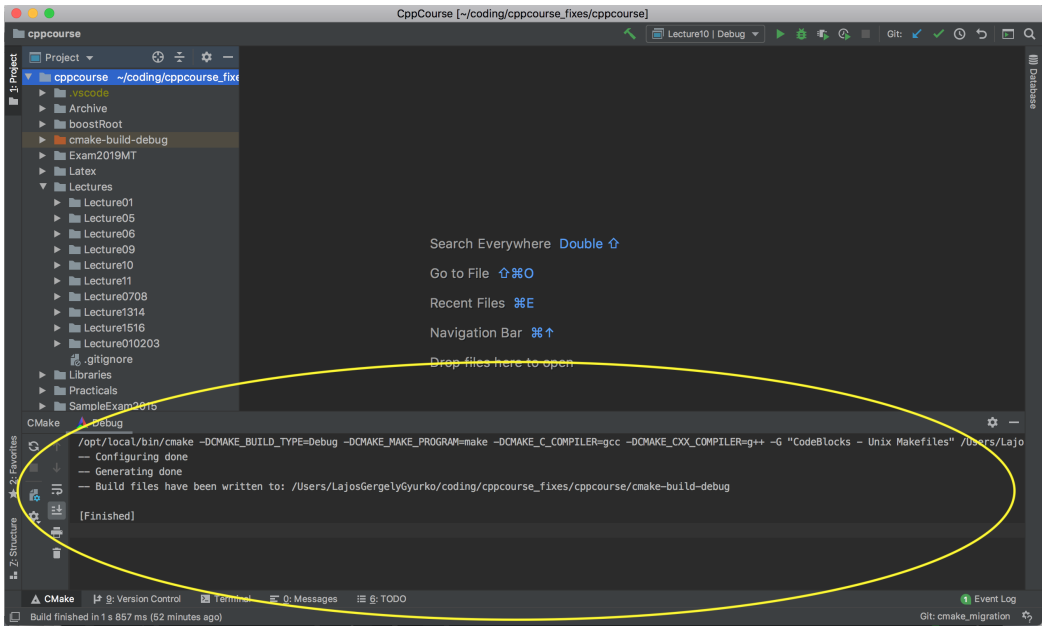

Figure 4: Output of running `CMake`

5.3 Building and running a project

Before building a project, you have to select a build target from the drop down box as indicated in figure 5. If the project you would like to select does not appear in the drop

down list, then executing one of the following two steps are likely to fix the issue:

- run `CMake` ,
- uncomment the appropriate `#add_subdirectory(...` line in the main `CMakeLists.txt` file.

Once the build target is set, you can run the build by clicking the icon marked in figure 6. The output of the project appears in a window at the bottom of the screen as shown in figure 7.
If the build was successful, you can launch the project by clicking on the icon indicated in figure 8. The output of the run appears in a window at the bottom of the screen as shown in figure 9.
Note that one of the lecture projects has a `main` function that takes argument. This can be done by evoking the "Edit configurations..." menu (see figure 10), and adding the arguments to the text box indicated in figure 11.

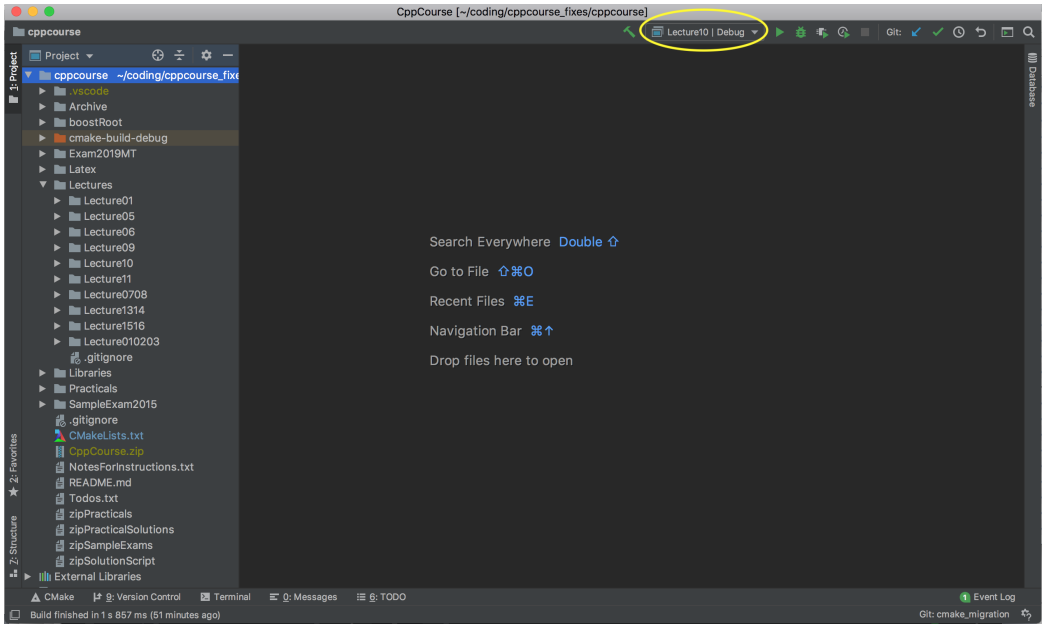

Figure 5: Selecting build target

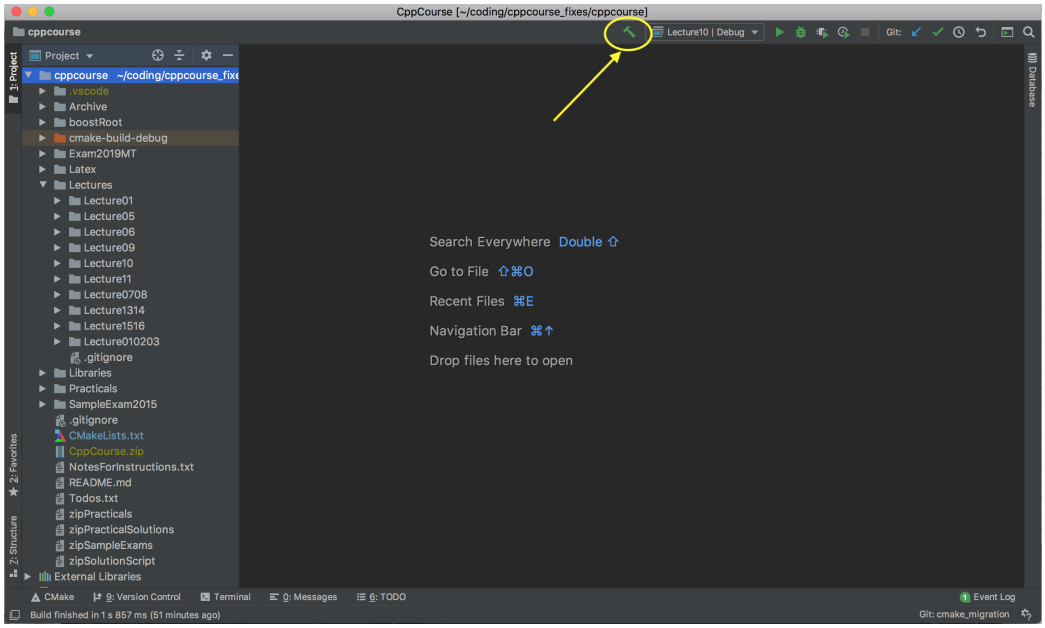

Figure 6: Build project

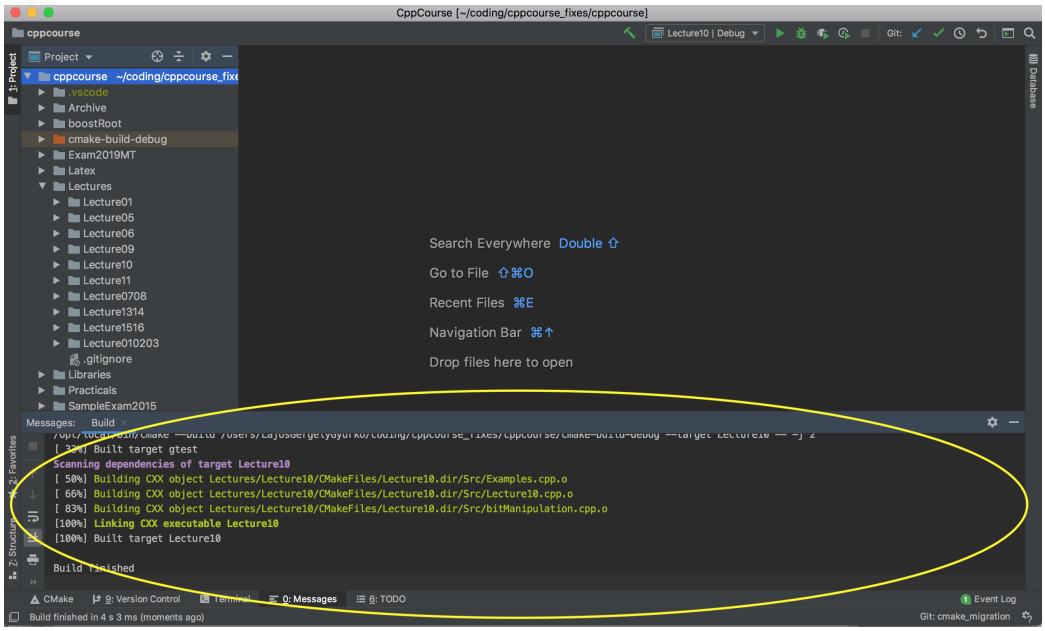

Figure 7: Output build project

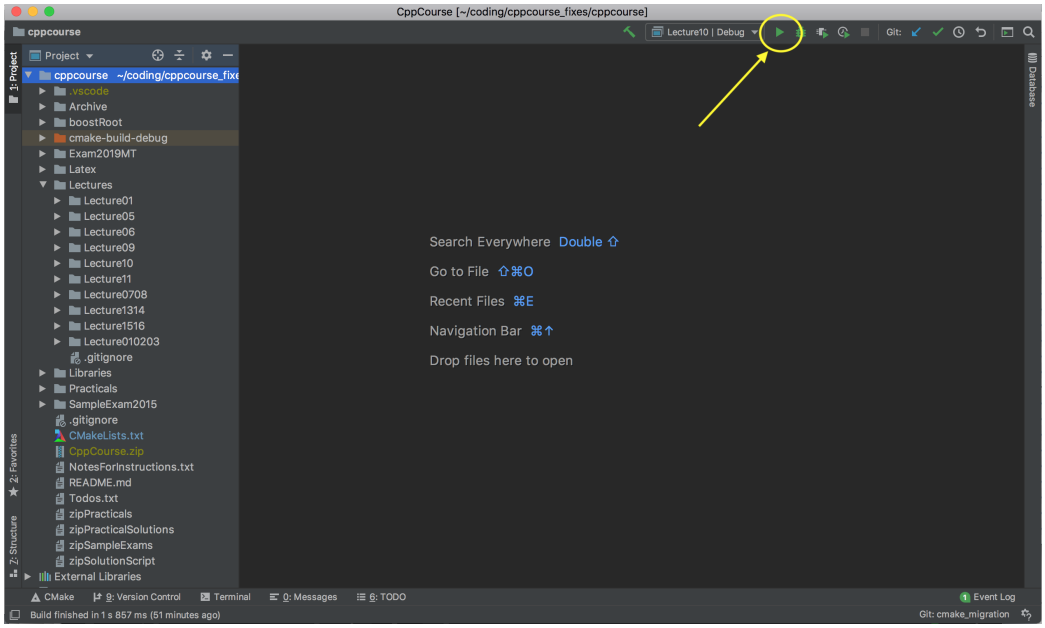

Figure 8: Run project

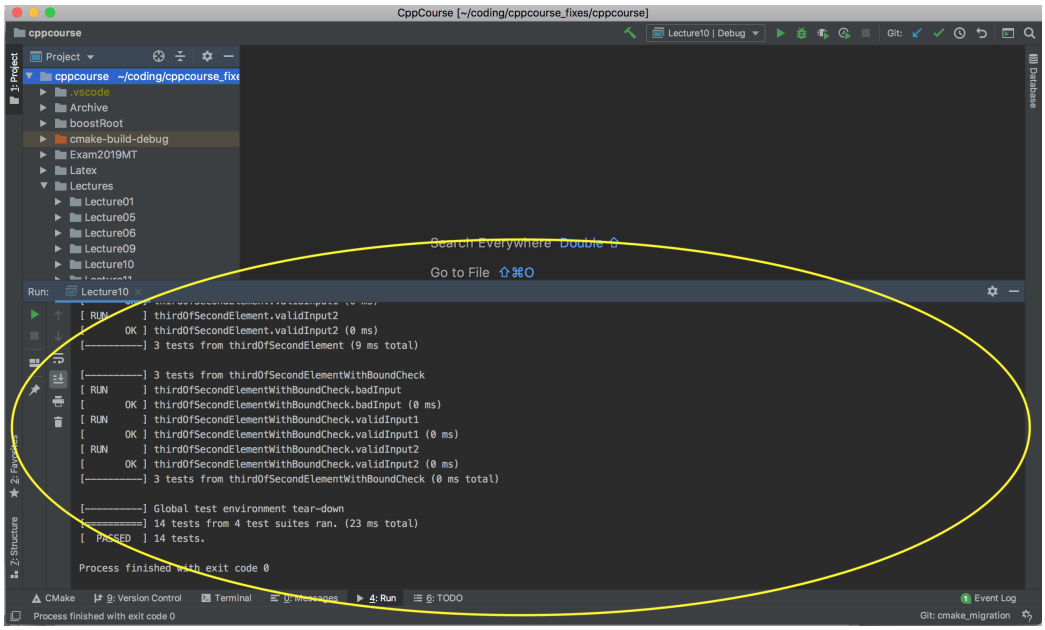

Figure 9: Output of project

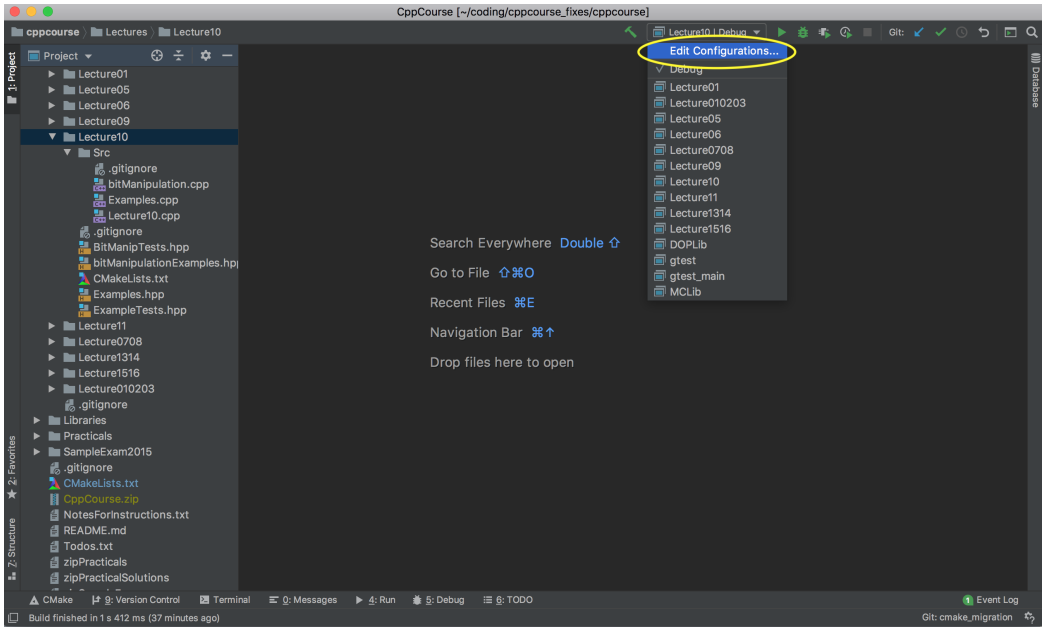

Figure 10: Set project arguments

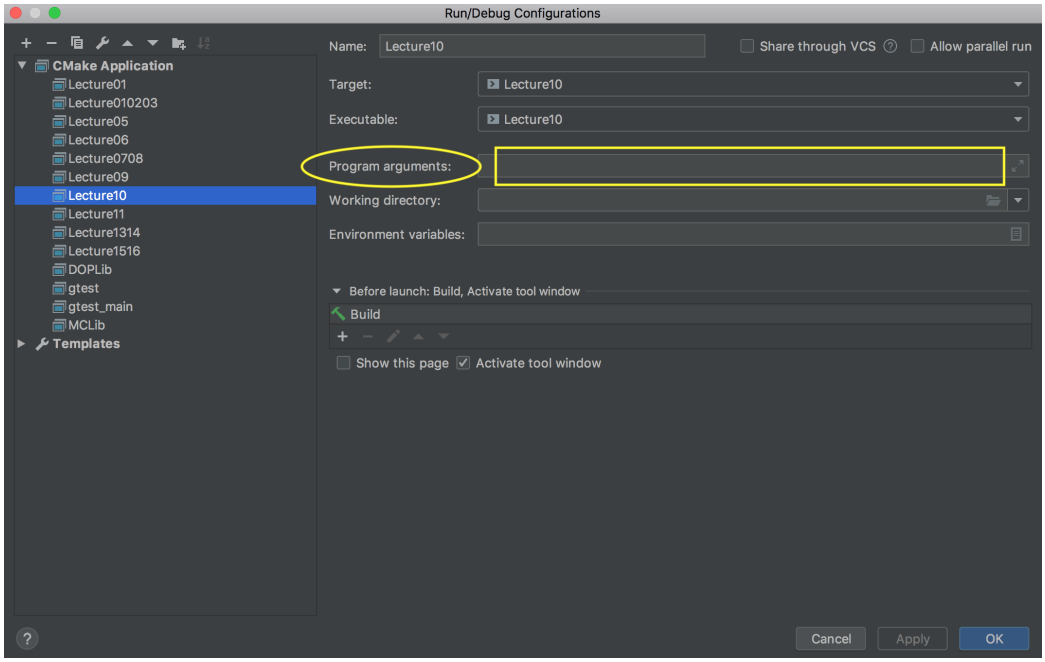

Figure 11: Set project arguments 2

5.4 Debugging a project

You can open source coude files from the project view, set breakpoints to lines where you'd like to stop the code execution. You can start debugging by clicking the icon indicated in figure 12.
When the execution stops at a breakpoint, you can track the call stack and the values of the local variables in the debug window at the bottom of the screen as shown in figure 13. The usual debug controls ("Rerun", "Resume", "Stop", Step over", "Step in", etc.) are available, see the icons indicated in figure 14.

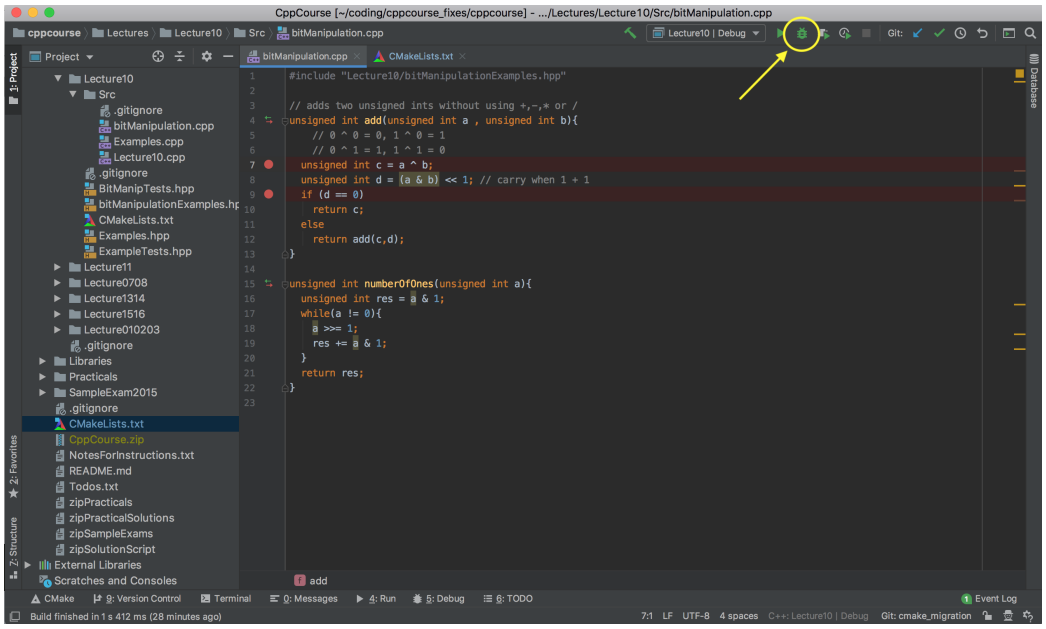

Figure 12: Start debugging

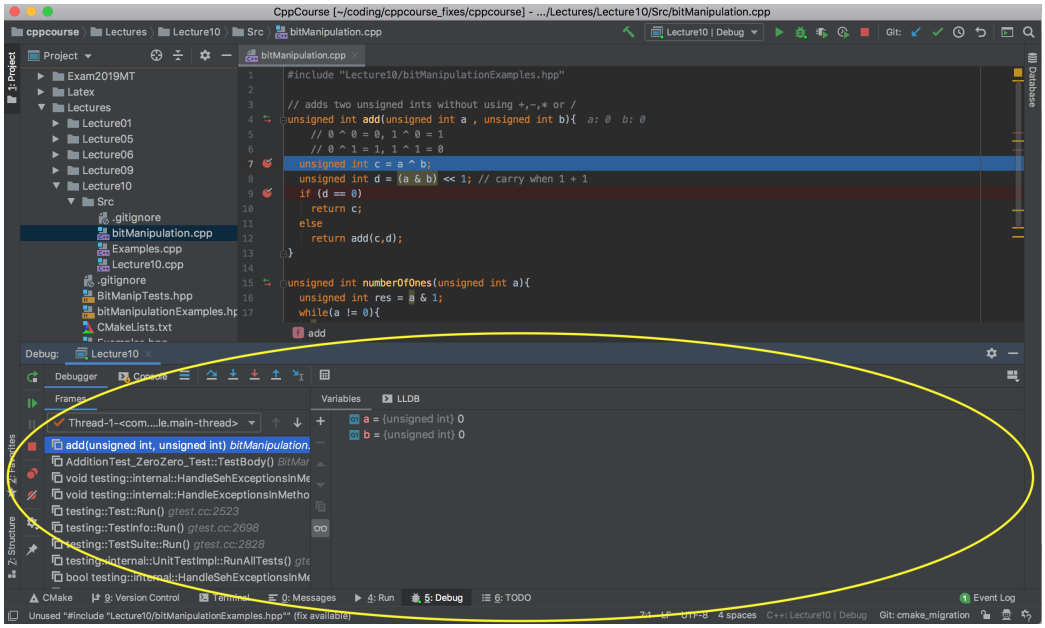

Figure 13: Debugging

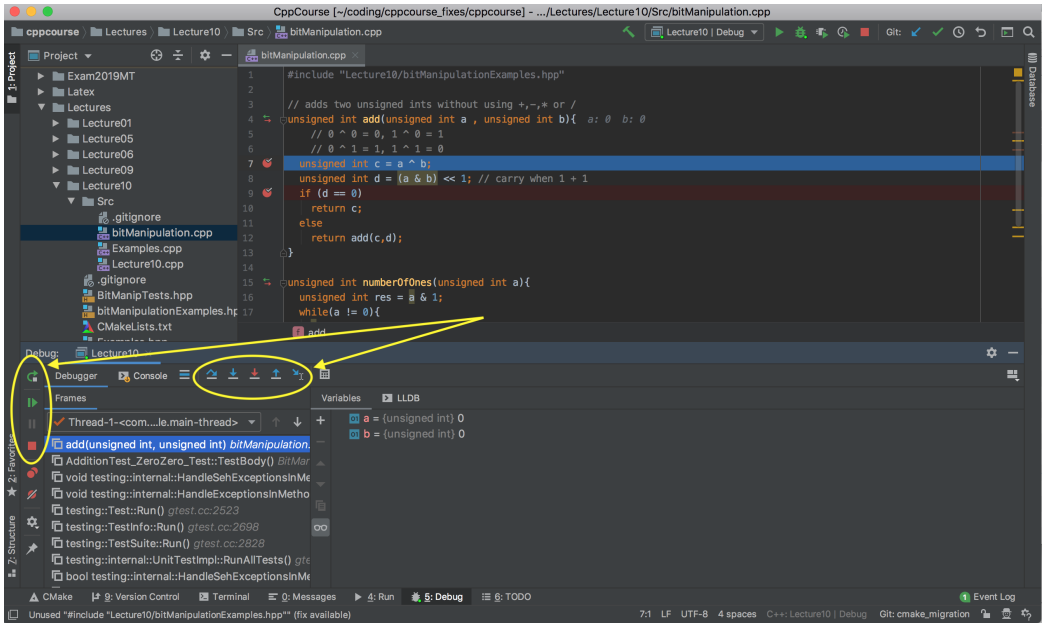

Figure 14: Debugging controls