Programmazione

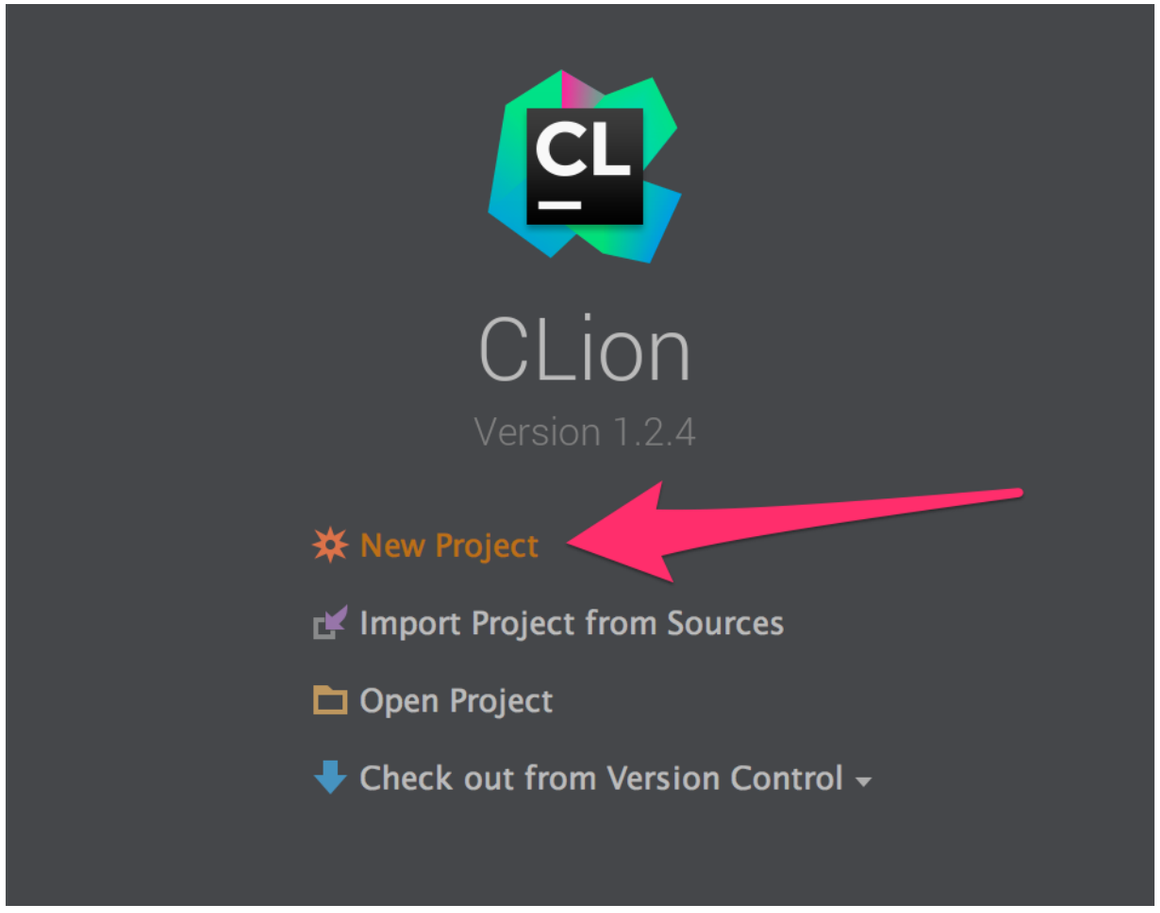
Prof. Marco Bertini
marco.bertini@unifi.it
http://www.micc.unifi.it/bertini/

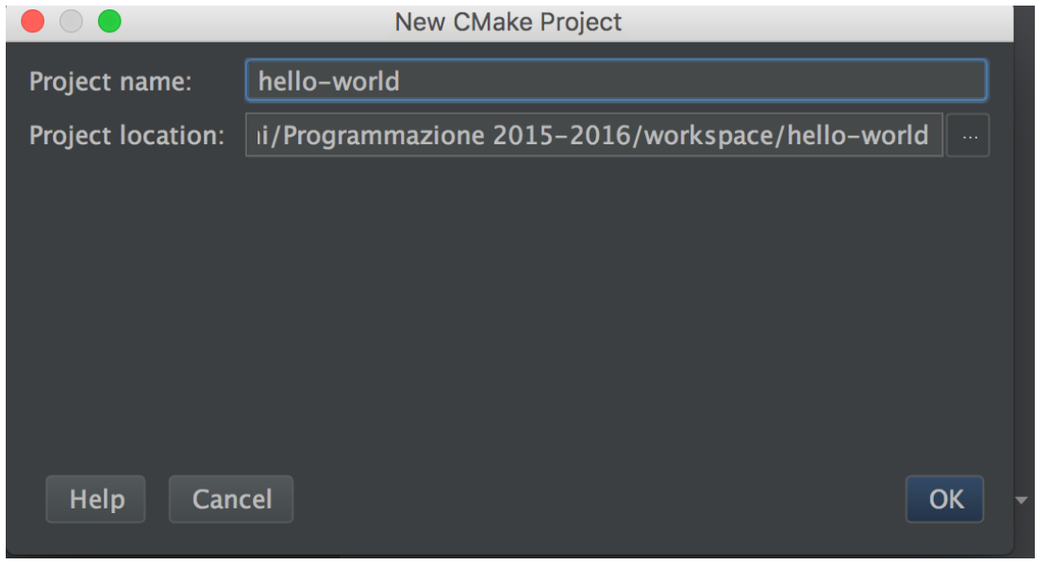
Building a "Hello world" with CLion

"When debugging, novices insert corrective code; experts remove defective code."
- Richard Pattis

Use the start dialog

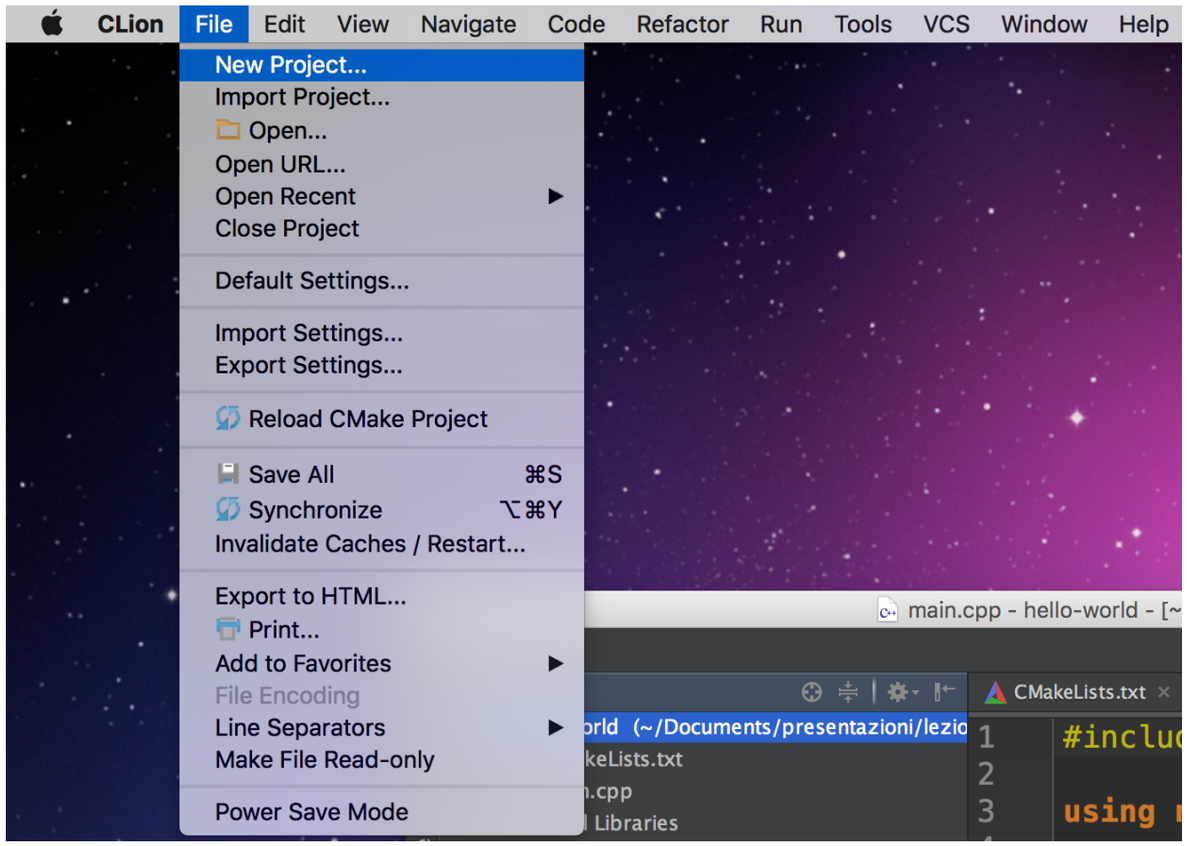
- Create a new project using the start dialog of CLion

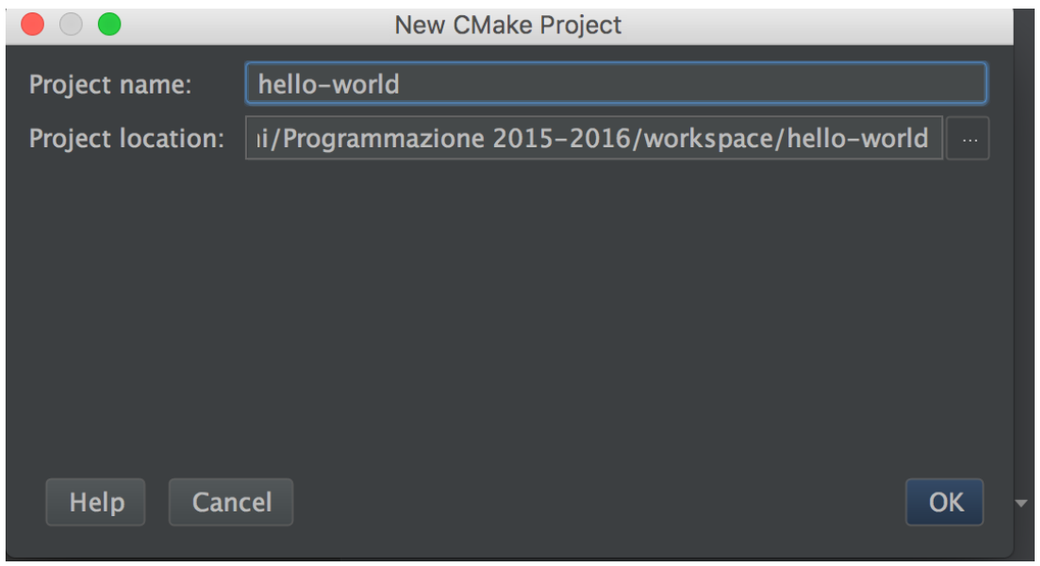




Use the project wizard

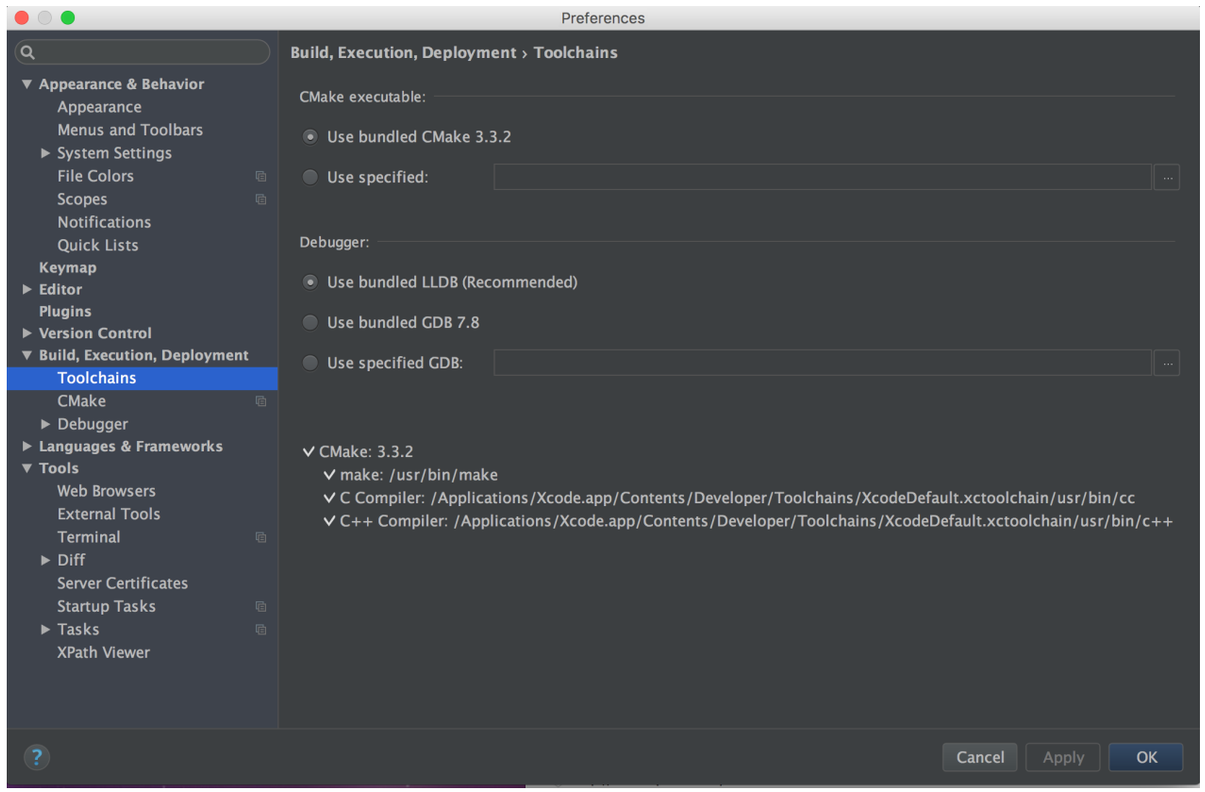
- File > New Project
- You can decide to use the old window or create a new wind, to work on two projects at the same time





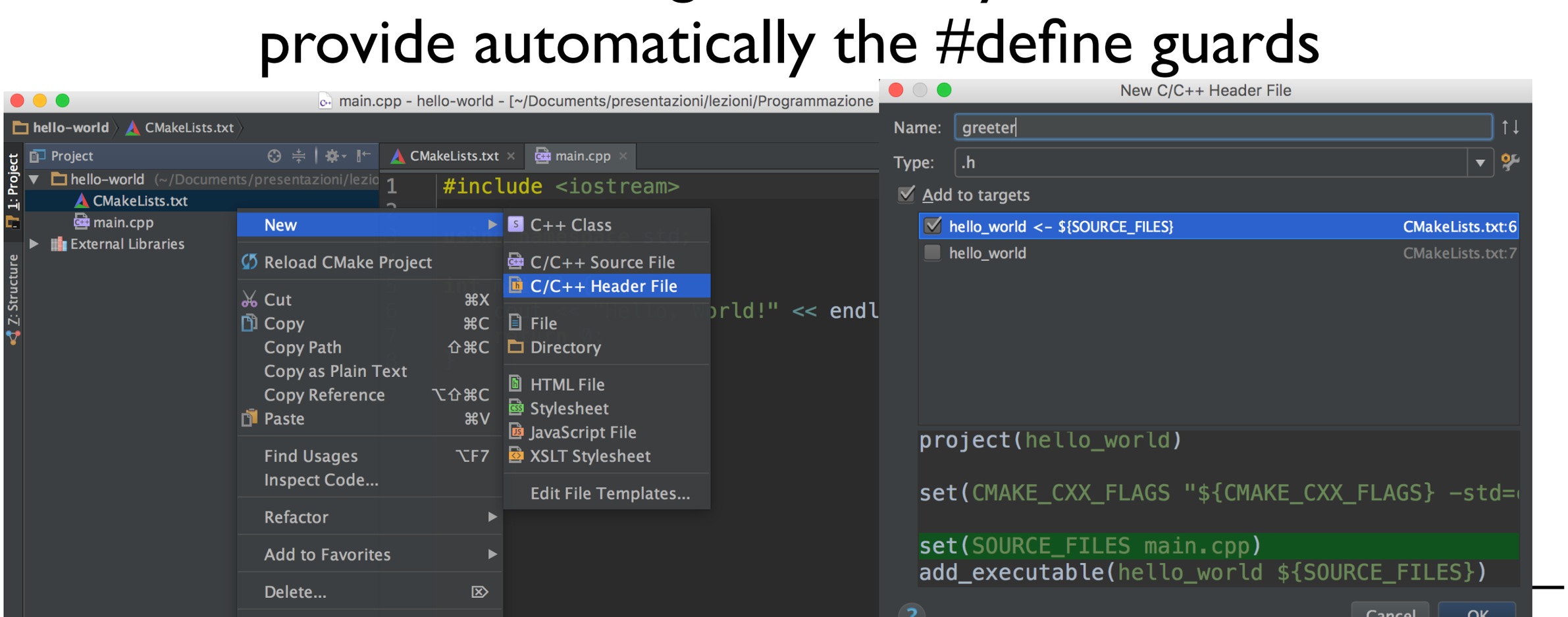
Select the tools

- When you install CLion it will discover if there is the required compiler:
  - OS X: LLVM with Clang (most recent)
  - Windows: MINGW GCC
  - Linux: GCC
- CLion has its own debugger. You can change the selected tools.



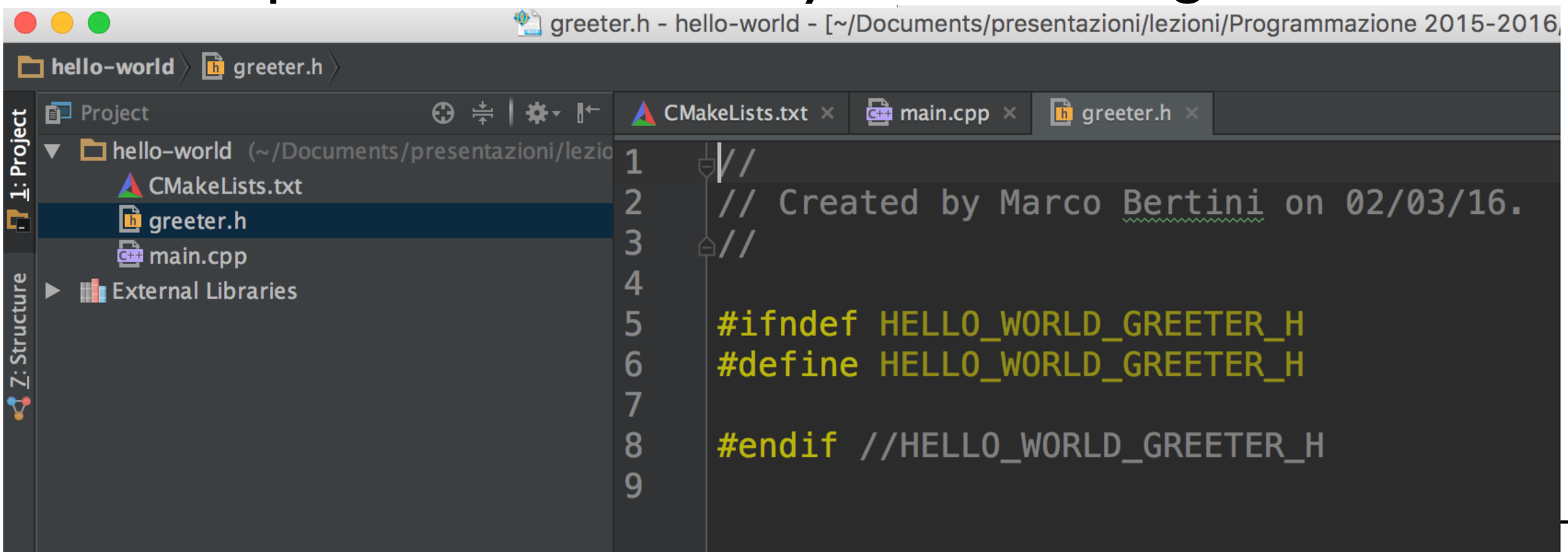
Add a .cpp and .h files

- Add, for example a .h file that contains a function to greet a user, given his name, and add the prototype in the include
  - if the include is generated by CLion, it will provide automatically the #define guards



Add a .cpp and .h files

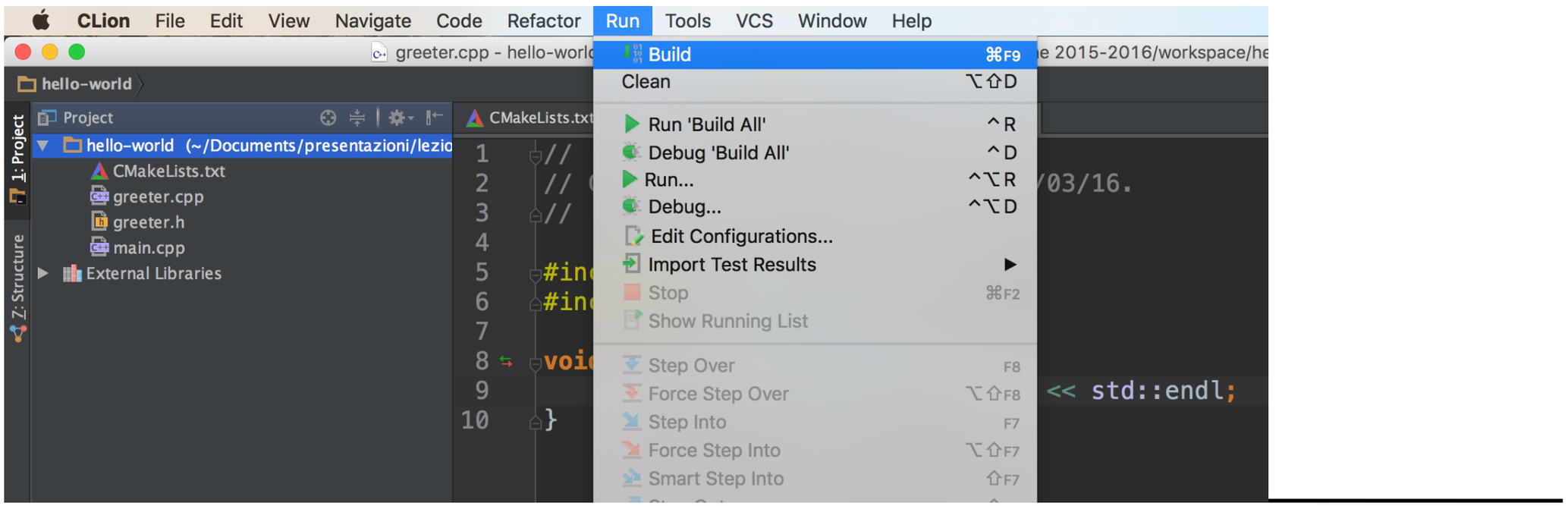
- Add, for example a .h file that contains a function to greet a user, given his name, and add the prototype in the include
  - if the include is generated by CLion, it will provide automatically the #define guards



Compile

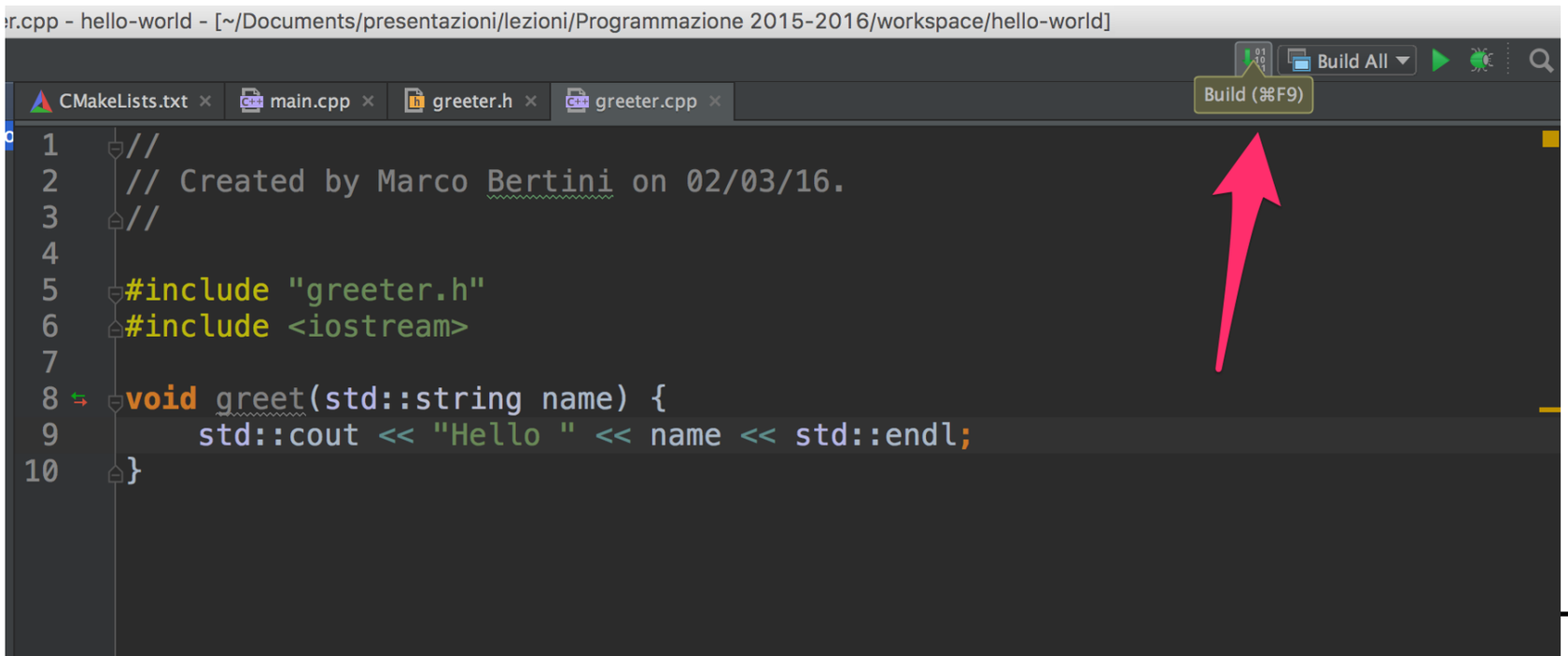
- Let's say the code has been written in the .cpp (including all the includes required, e.g. `iostream` and the `greeter.h`): compile using Run > Build or using the toolbar icon.

- Check the compile errors (shown in the message panel)



Compile

- Let's say the code has been written in the .cpp (including all the includes required, e.g. `iostream` and the `greeter.h`): compile using Run > Build or using the toolbar icon.
- Check the compile errors (shown in the message panel)



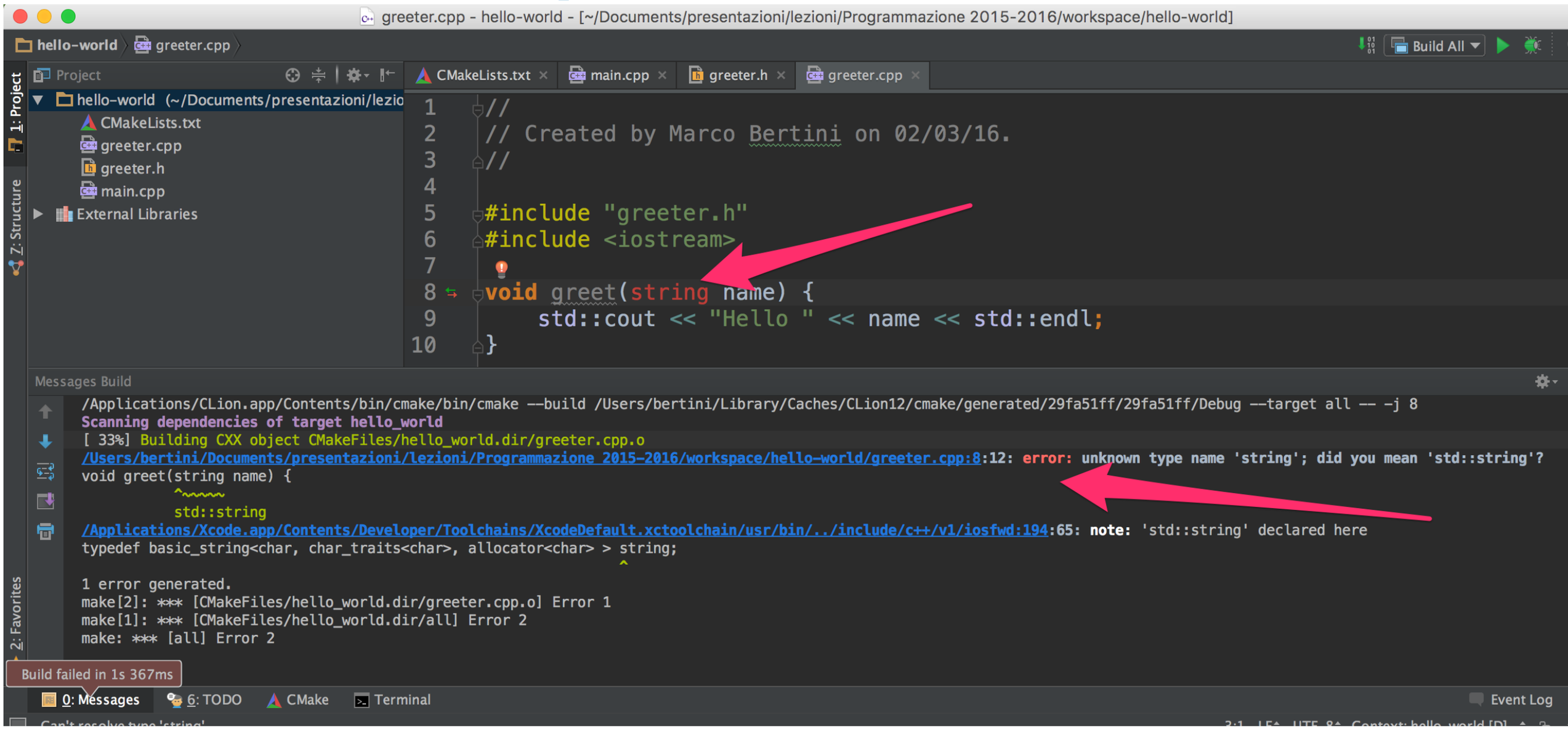
Compile

- Let's say the code has been written in the .cpp (including all the includes required, e.g. `iostream` and the `greeter.h`): compile using Run > Build or using the toolbar icon.
- Check the compile errors (shown in the message panel)

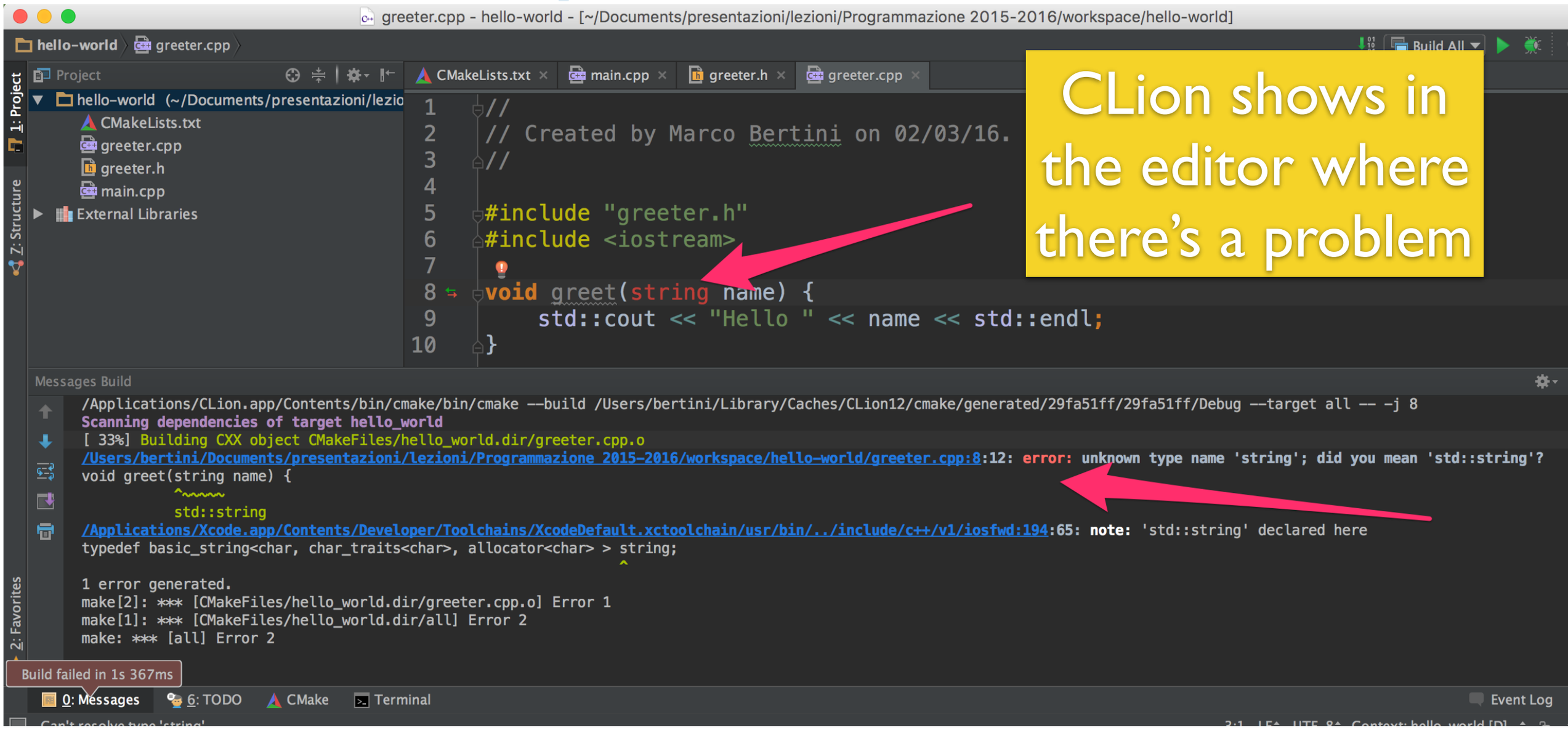
Compile errors

- Don’t panic
- Start reading (carefully) the messages from the first to the last. Solve the first errors, perhaps they have an influence on the others.
  - In the example the first error is in the `.cpp`

Compile errors - cont.



Compile errors - cont.



Compile errors - cont.

CLion shows in the editor where there's a problem

Read the message: the type string is unknown. The compiler even suggests the correct type.

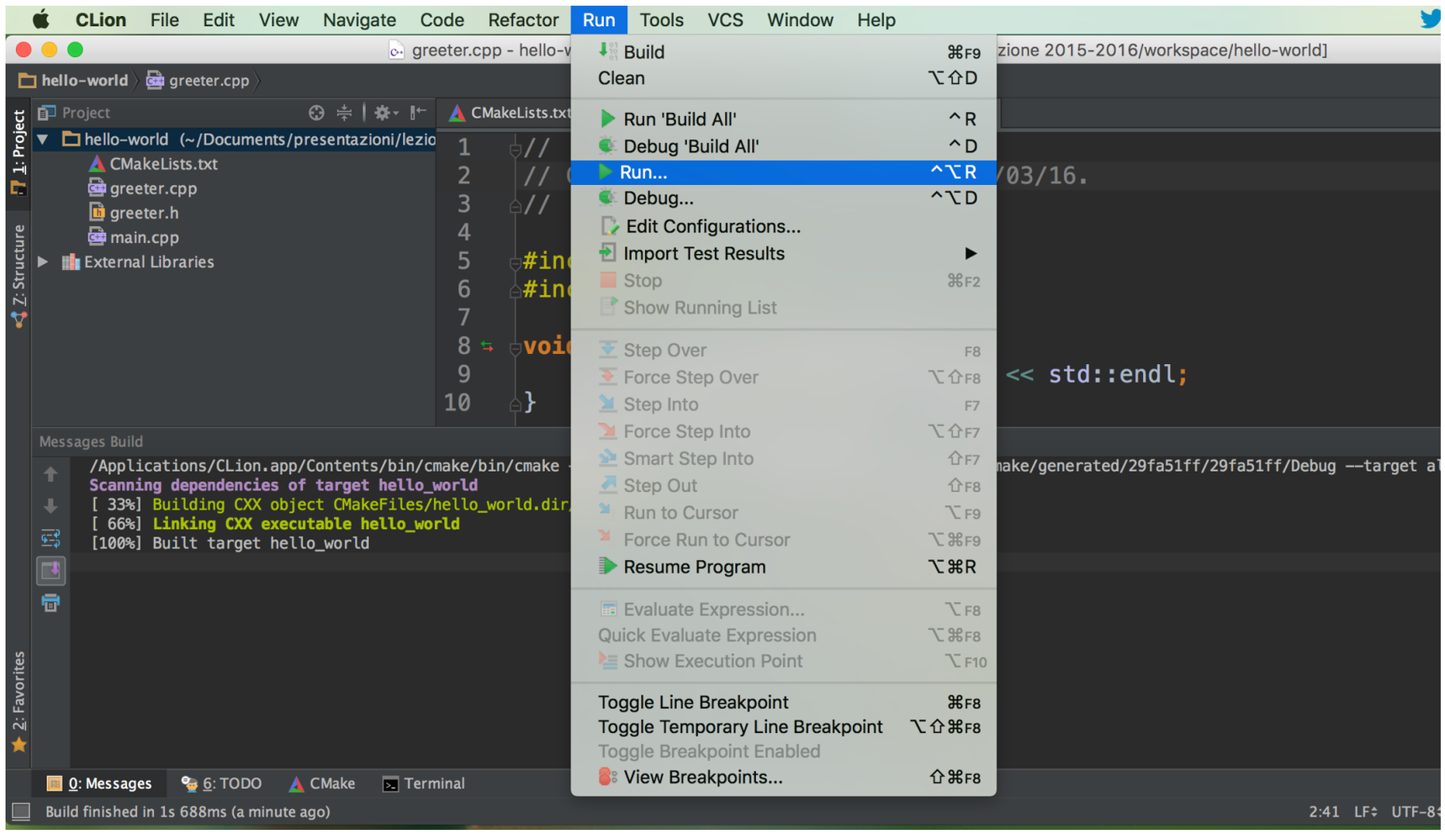
Error messages will vary depending on the compiler though !

Compile errors - cont.

- Correct the error: in this case it was necessary to add `std::` to string (we are not using "`using namespace std;`" in this file !
- Build again to check the correction

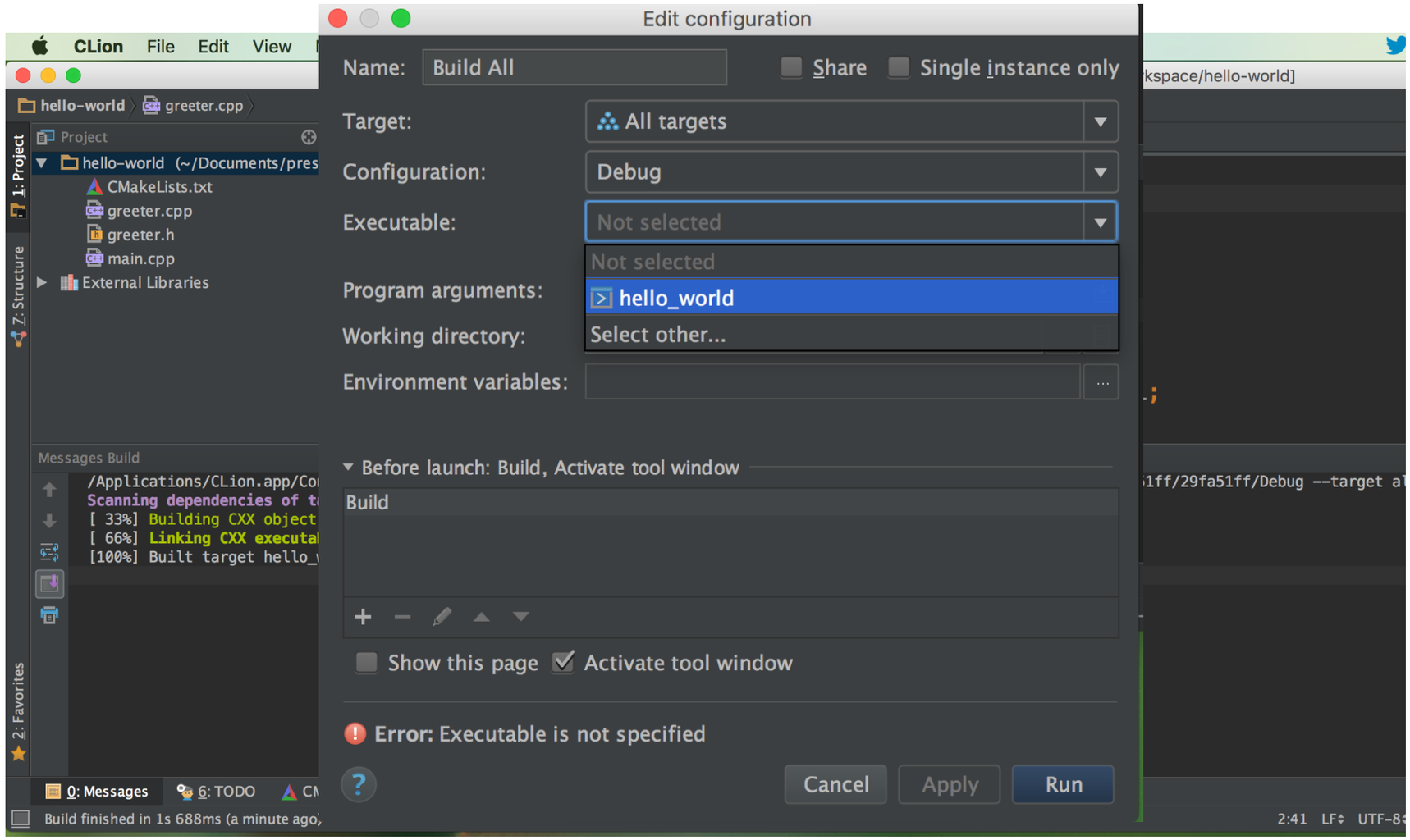
Run the program

- Use the menu Run > Run… > Select the executable to run (a project may have more than one).
  Later on the program will appear in the Run History



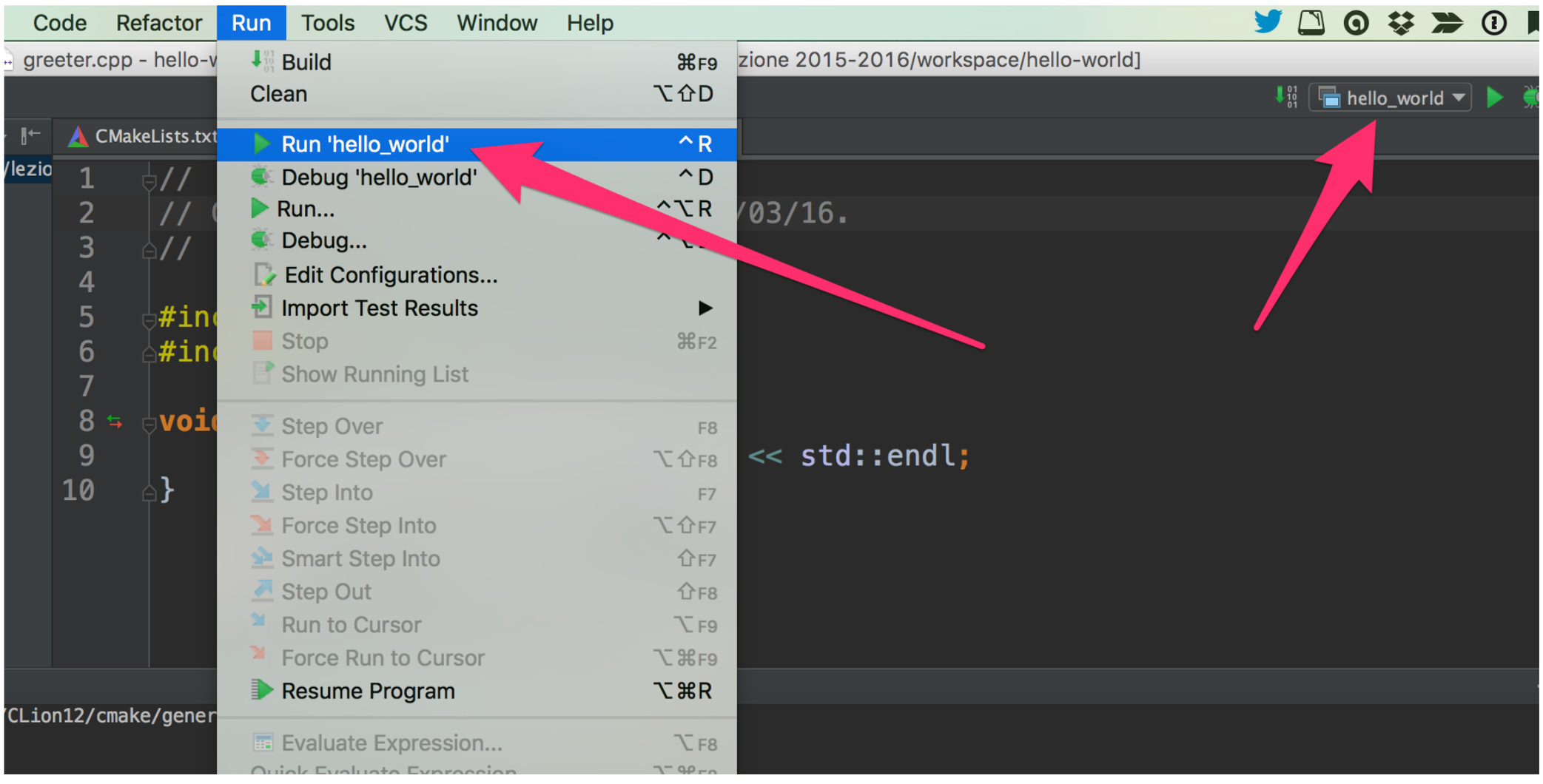
Run the program

- Use the menu Run > Run… > Select the executable to run (a project may have more than one).
  Later on the program will appear in the Run History



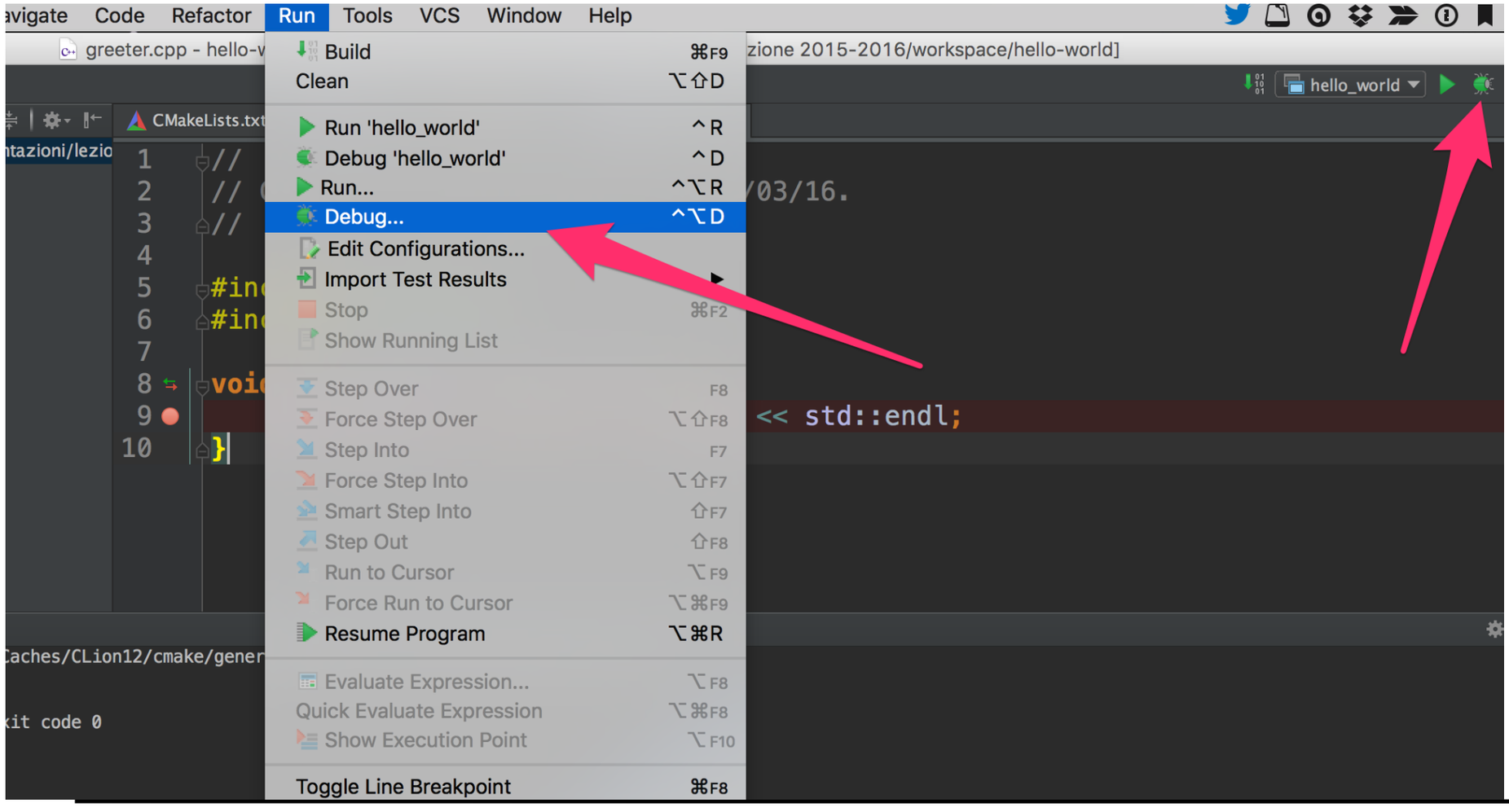
Run the program

- Use the menu Run > Run… > Select the executable to run (a project may have more than one).
  Later on the program will appear in the Run History



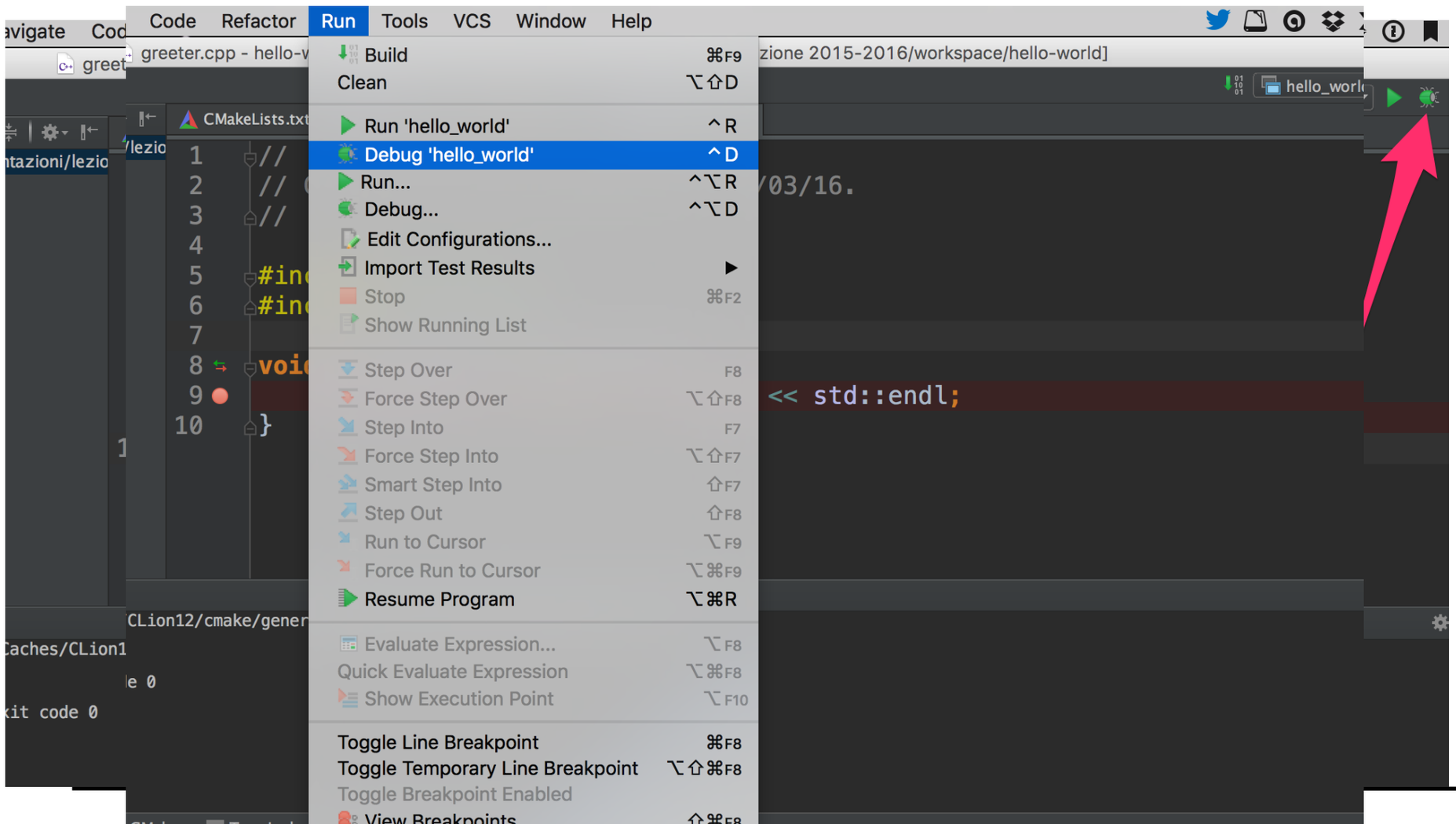
Debug the program

- In order to debug the program must be compiled so that additional information, useful for the debugger, is added to the files
- Add a breakpoint in CLion (click the left side of the line), then execute the program in the debugger (Run > Debug application)



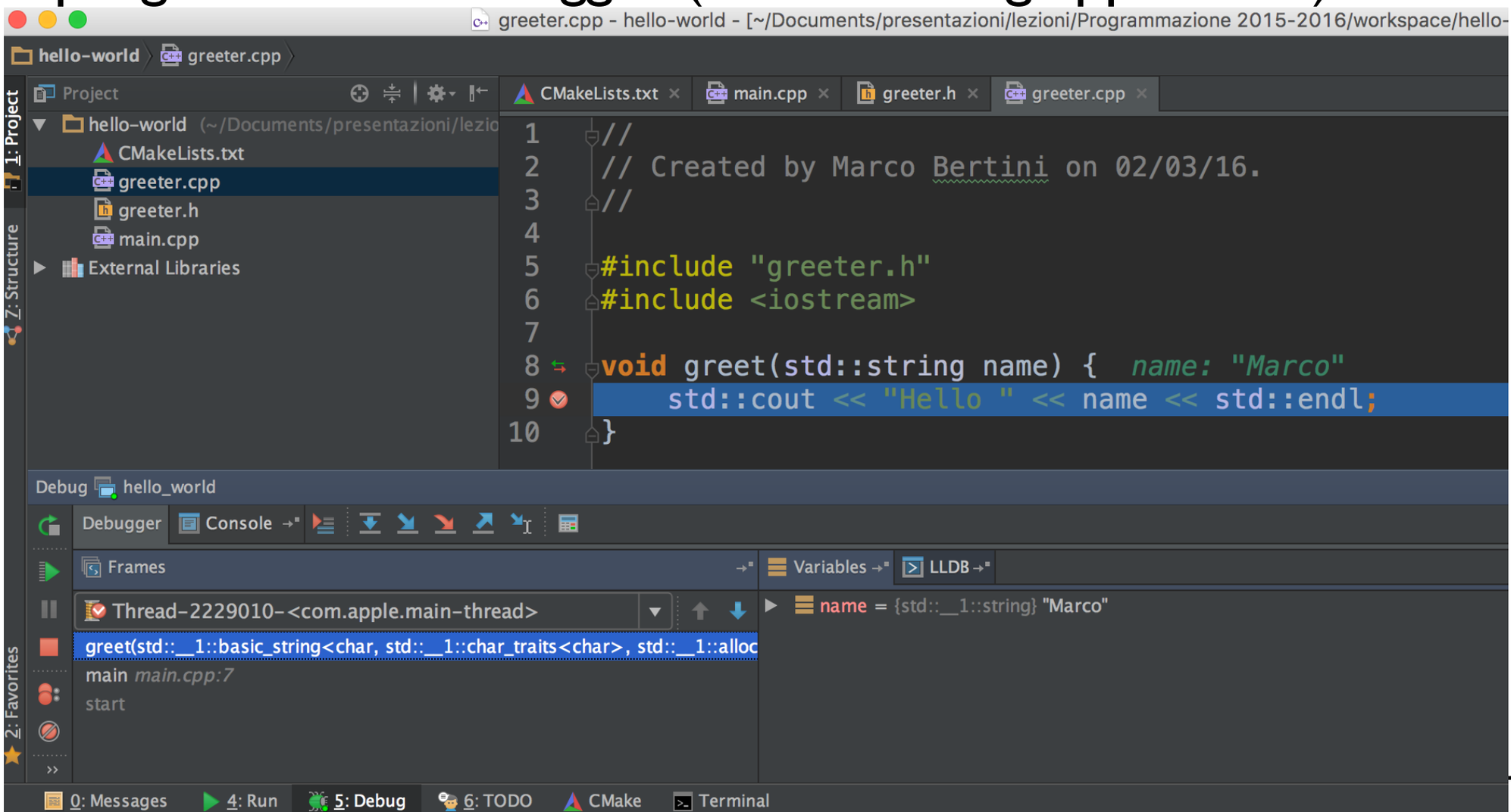
Debug the program

- In order to debug the program must be compiled so that additional information, useful for the debugger, is added to the files
- Add a breakpoint in CLion (click the left side of the line), then execute the program in the debugger (Run > Debug application)



Debug the program

- In order to debug the program must be compiled so that additional information, useful for the debugger, is added to the files
- Add a breakpoint in CLion (click the left side of the line), then execute the program in the debugger (Run > Debug application)



Some style guidelines

- There are a plethora of C++ coding style recommendations, sometimes even contradictory.
- Two very good recommendations:
  1. Any violation to the guidelines is allowed if it enhances readability.
  2. The rules can be violated if there are strong personal objections against them.

Naming conventions

- Names representing types must be in mixed case starting with upper case: follow this rule when writing classes.
- Variable names must be in mixed case starting with lower case (like Java).
- Names representing methods or functions must be verbs and written in mixed case starting with lower case (like Java).

Naming conventions - cont.

- Names representing namespaces should be all lowercase.
- All names should be written in English.

Files

- C++ header files should have the extension .h (preferred) or .hpp. Source files can have the extension .c++, .C, .cc or .cpp.
- A class should be declared in a header file and defined in a source file where the name of the files match the name of the class.
- Header files must contain an include guard.
- Include statements must be located at the top of a file only.