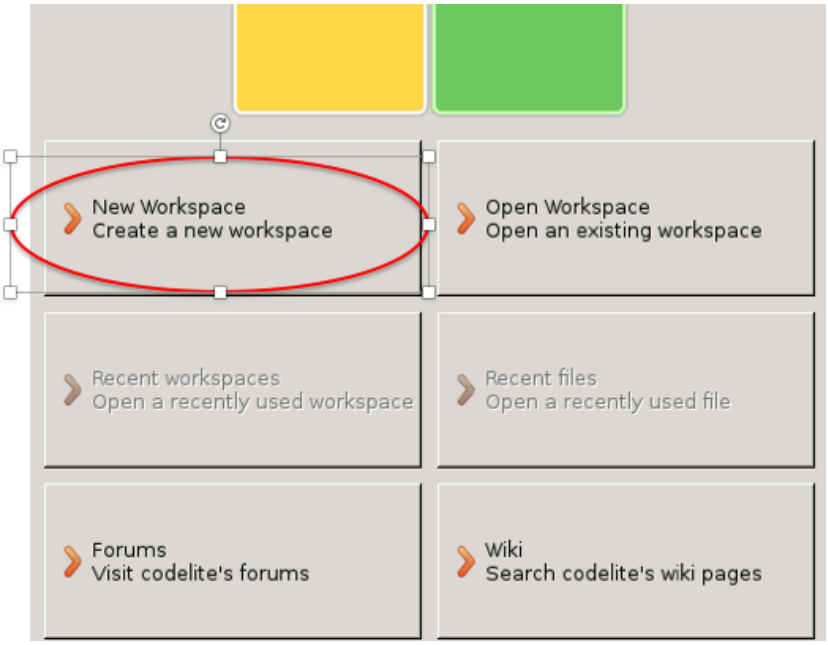
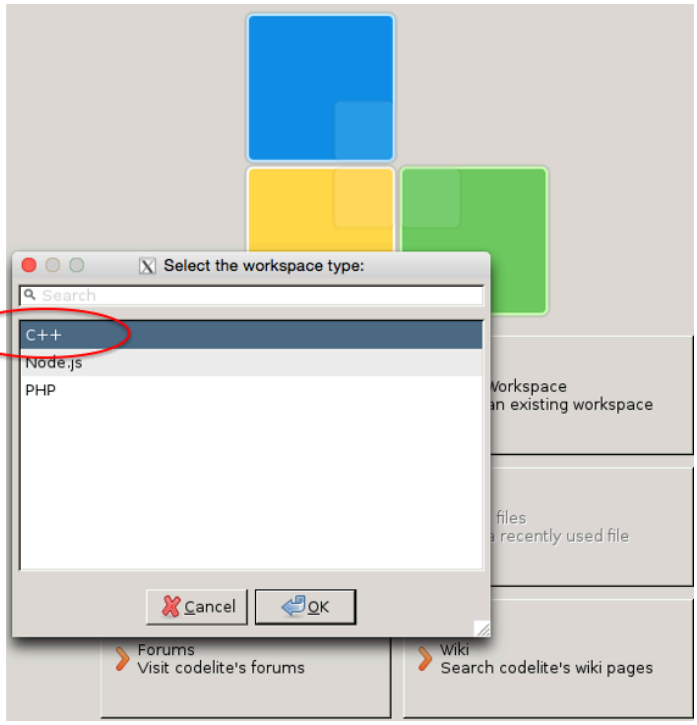
Lab 2A: Programming and Debugging with CodeLite

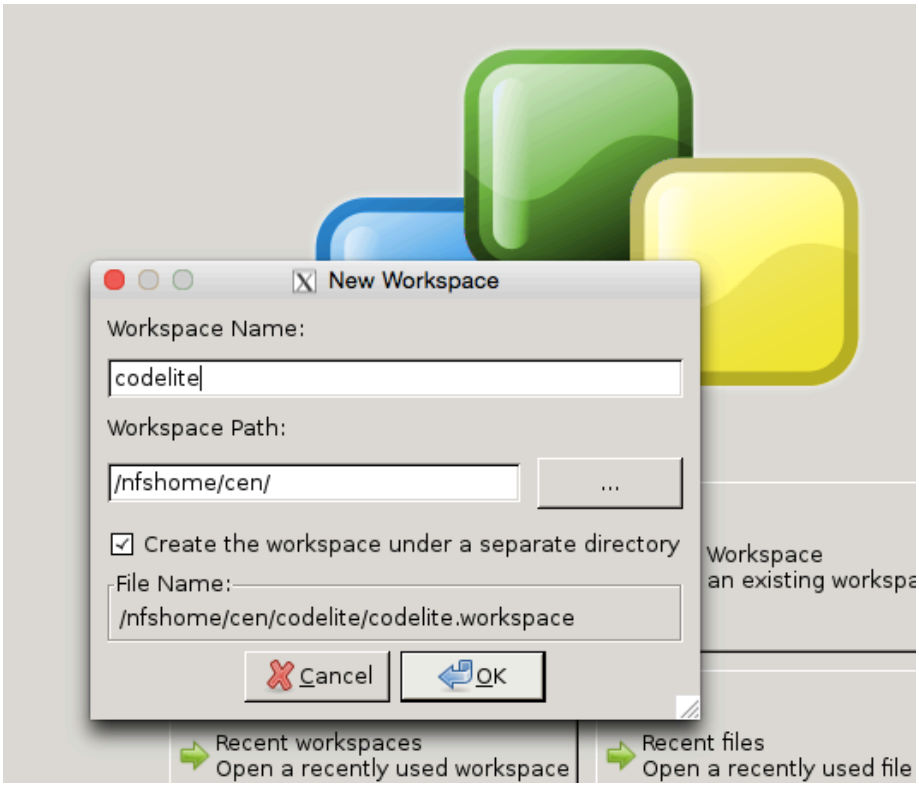
In this lab, you will learn to use CodeLite Integrated Development Environment(IDE) to write, compile, and and run a C++ program. Follow the steps shown below. In case of problem, ask the graduate teaching assistant for help.

First type ***codelite &*** in a terminal to start the program. Or, select the program from the application list, and double click to start the program.The first time you use CodeLite, you will be asked to create a workspace. Click on "**New Workspace**", and then select "**C++**" as the workspace type. Then, name your workspace, for example "**codelite**".



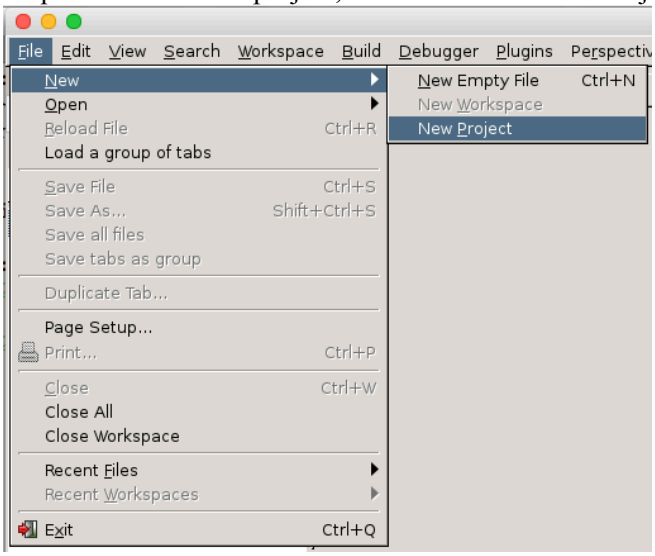


Remember the path to your project. For example in this case, my workspace is in the directory named "codelite" in my home directory: **~cen/codelite**. All the projects I create in this workspace will be stored under this directory. For this class, create all the projects in this workspace, i.e., there is no need to create additional workspace after this first one is created.

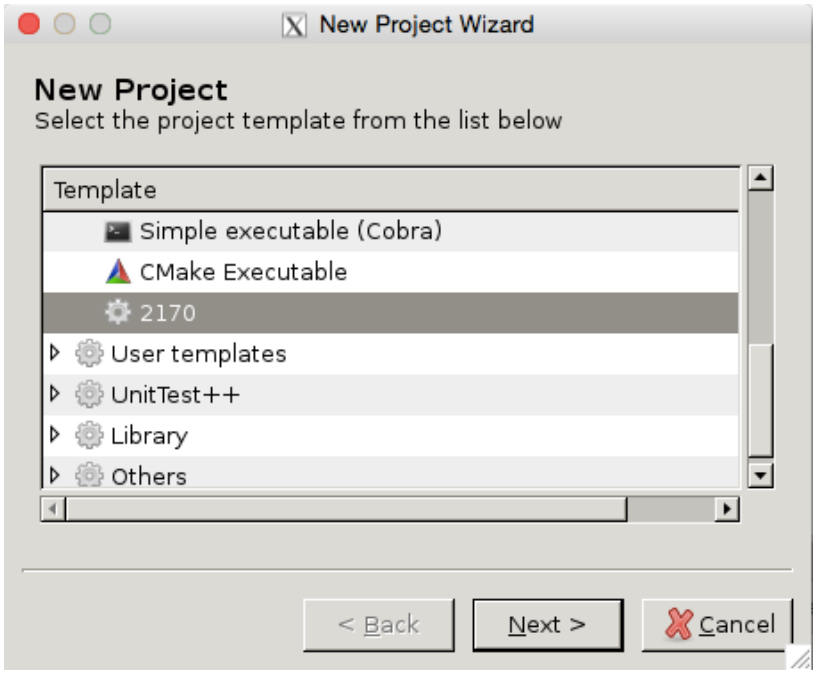


1. **After the workspace is created, you can start creating a new C++ project:**

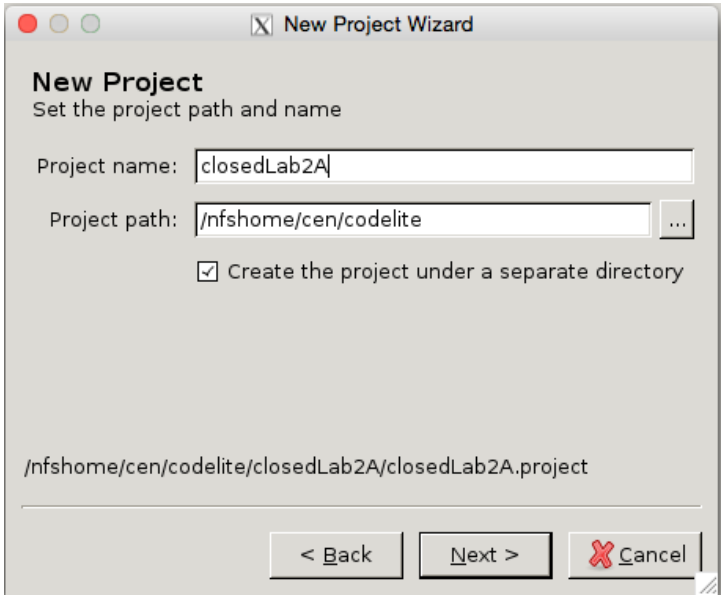
- Step 1: Create a new project, File → New → New Project



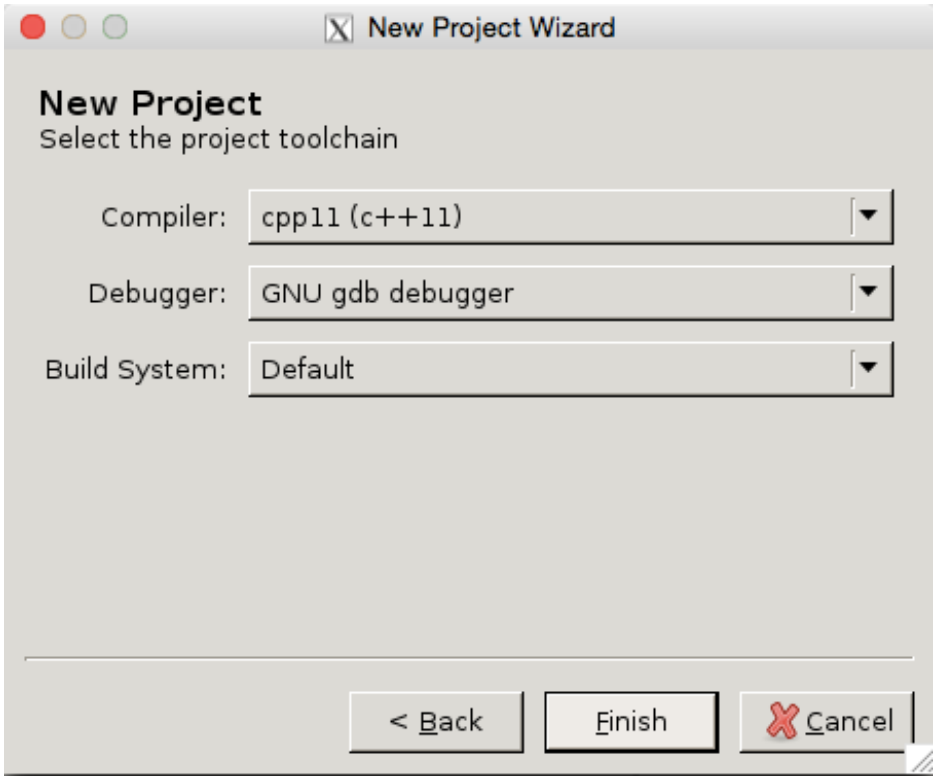
- Step 2: Select Console project → select “2170” Template



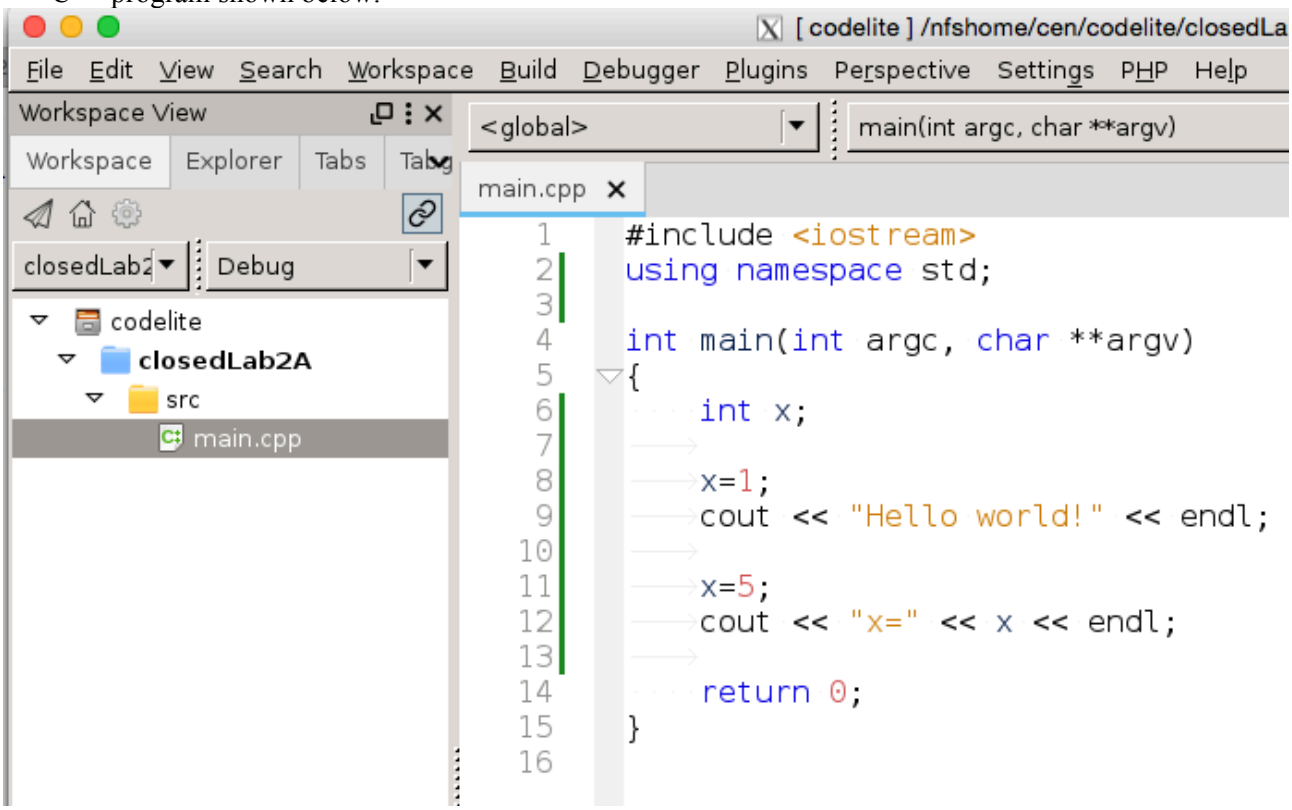
- Step 3: Specify project name and path for the project, i.e., where you want to store the project files.



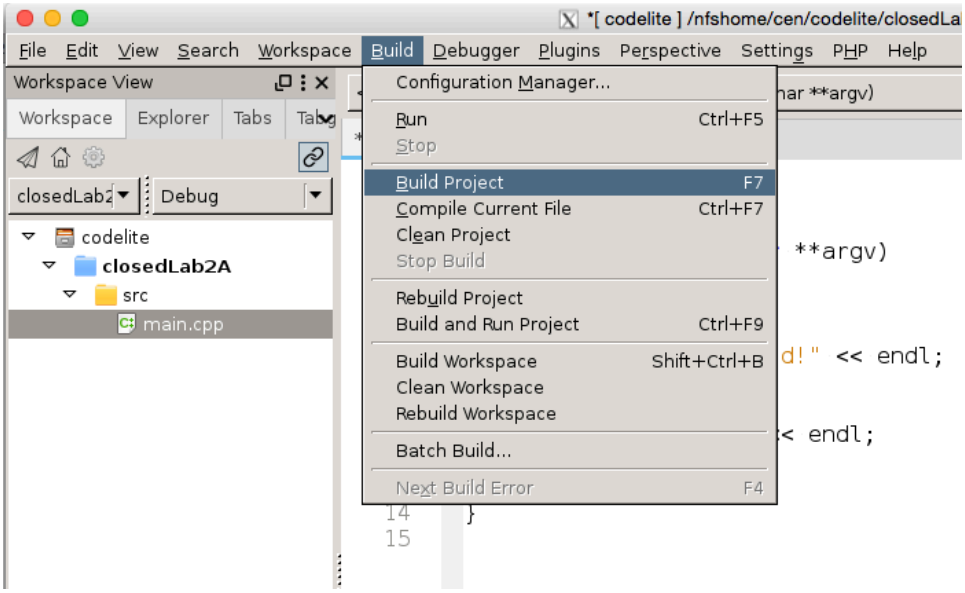
- Step 4: Specify compiler and debugger. Use “cpp11 (c++11) compiler” and “GNU gdb debugger”.



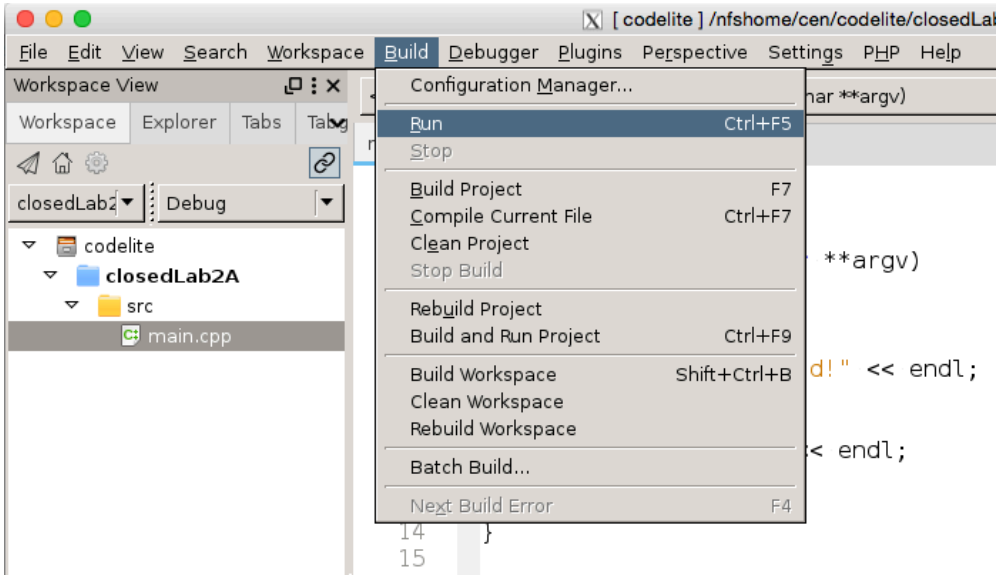
Step 5: Click to expand closedLab2A → src → main.cpp, overwrite the program "main.cpp" with the C++ program shown below.

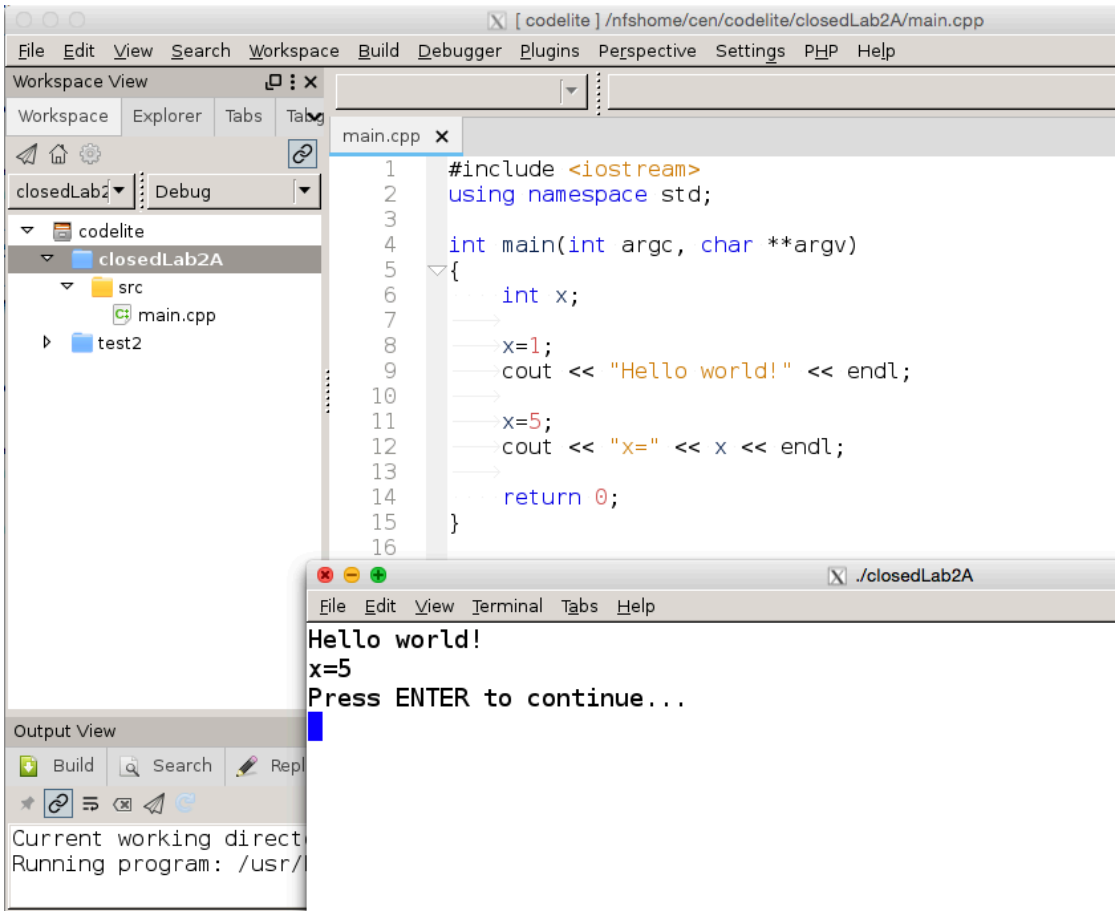


- Step 6: Save the file: File → Save File
- Step 7: Compile the program with: Build → Build Project



- Step 8: Run the program: **Build → Run.** The program output can be seen in a pop-up new window.



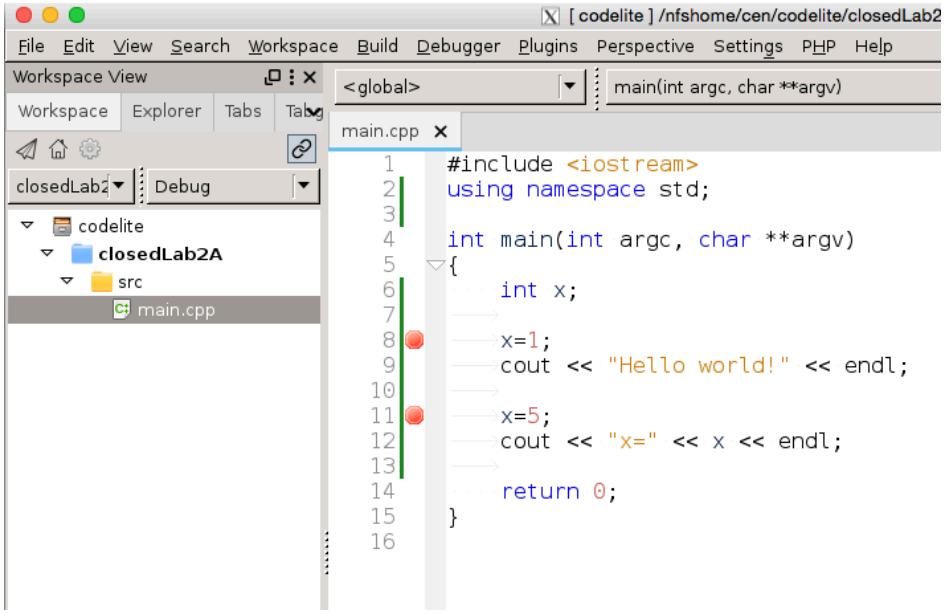


Debugging with CodeLite

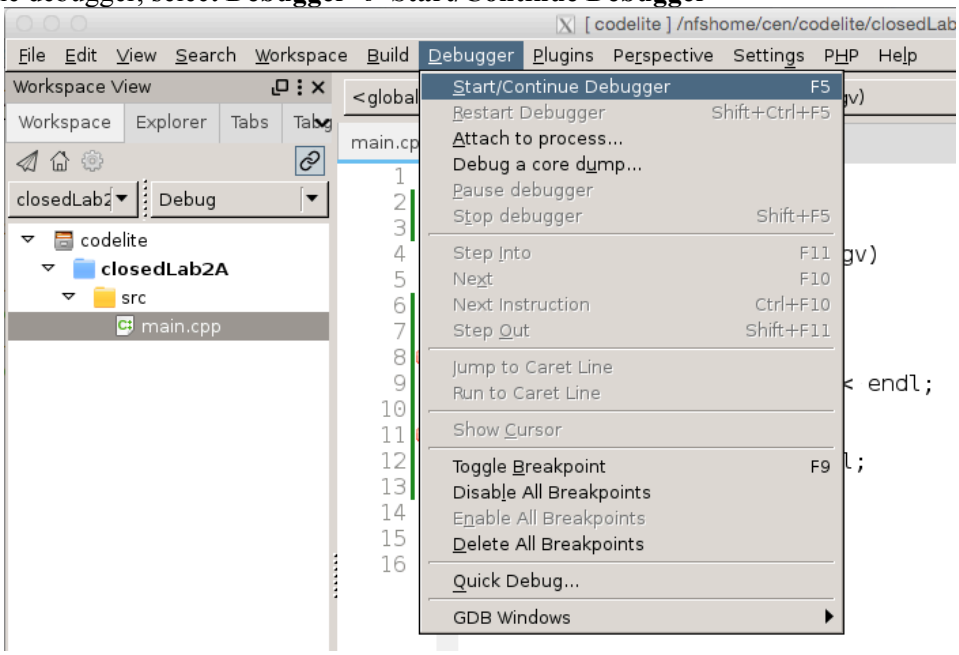
Debugging in an IDE allows one to step through the execution of the program, i.e., be able to monitor how the program is executed step by step, and to observe the value of the variables at pre-specified statements during program execution. This can be of great help in debugging a program.

First, decide where you want the program execution to pause during program execution. These pause points are called, break points. Once the break points are set, you can run the debugger, and step through the program with commands such as “next”, “step in”, “step over”, and “continue”.

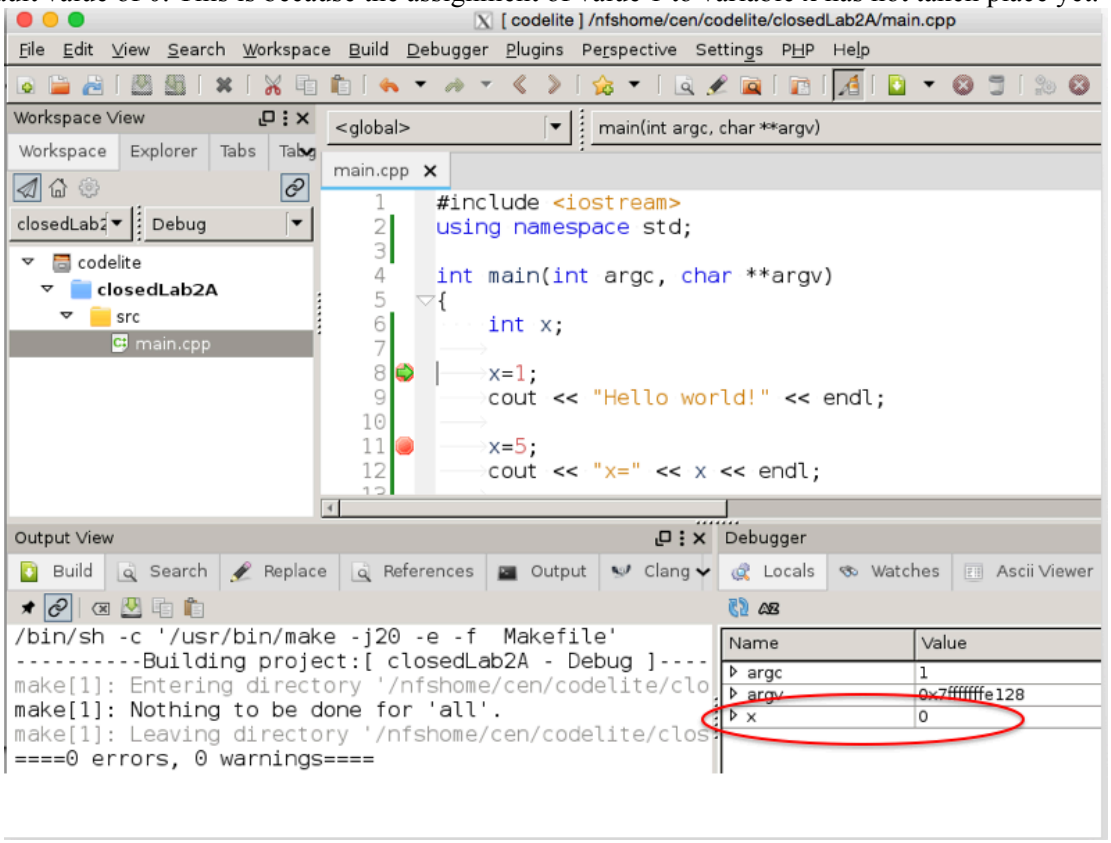
First, we take a look at how to set break points in a program. Point mouse in the gray area to the left of the line where we want to have a break point, and click. A small red circle appearing at the place of click indicating a break point is set for the line. In the example program below, break points are set for lines 8 and 11.



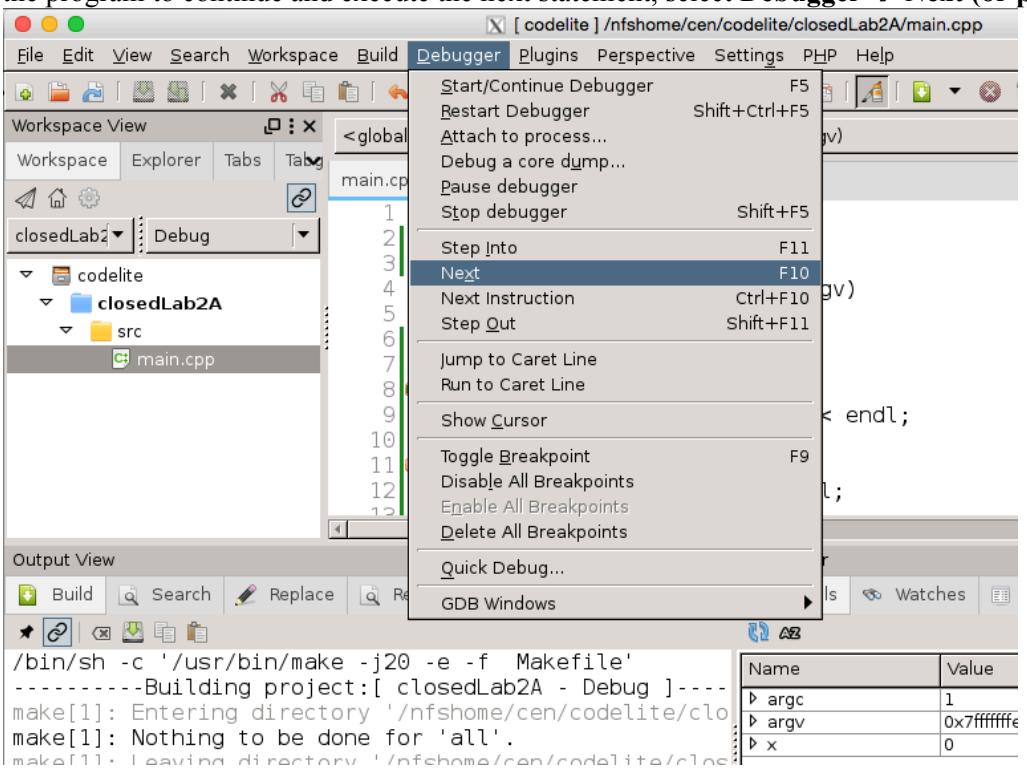
Now, to run the debugger, select **Debugger → Start/Continue Debugger**



The program execution will stop at the first statement of the program at line 8. Notice the small green arrow pointing next to line 8. Also, notice in the lower right window, the value of variable x is current the default value of 0. This is because the assignment of value 1 to variable x has not taken place yet.



To instruct the program to continue and execute the next statement, select **Debugger → Next (or press F10)**



The green arrow advances to line 8. The value of variable x is changed to 1 (in the lower right window).

Continue program execution by click on **Next** twice. Program executes the next two statements, and the value of variable x is now 5.