Western Engineering

ES1036: Programming Fundamentals for Engineers – Winter 2018
Lab 00: C++ with Xcode (For Mac Users)
Faculty of Engineering, Western University Canada

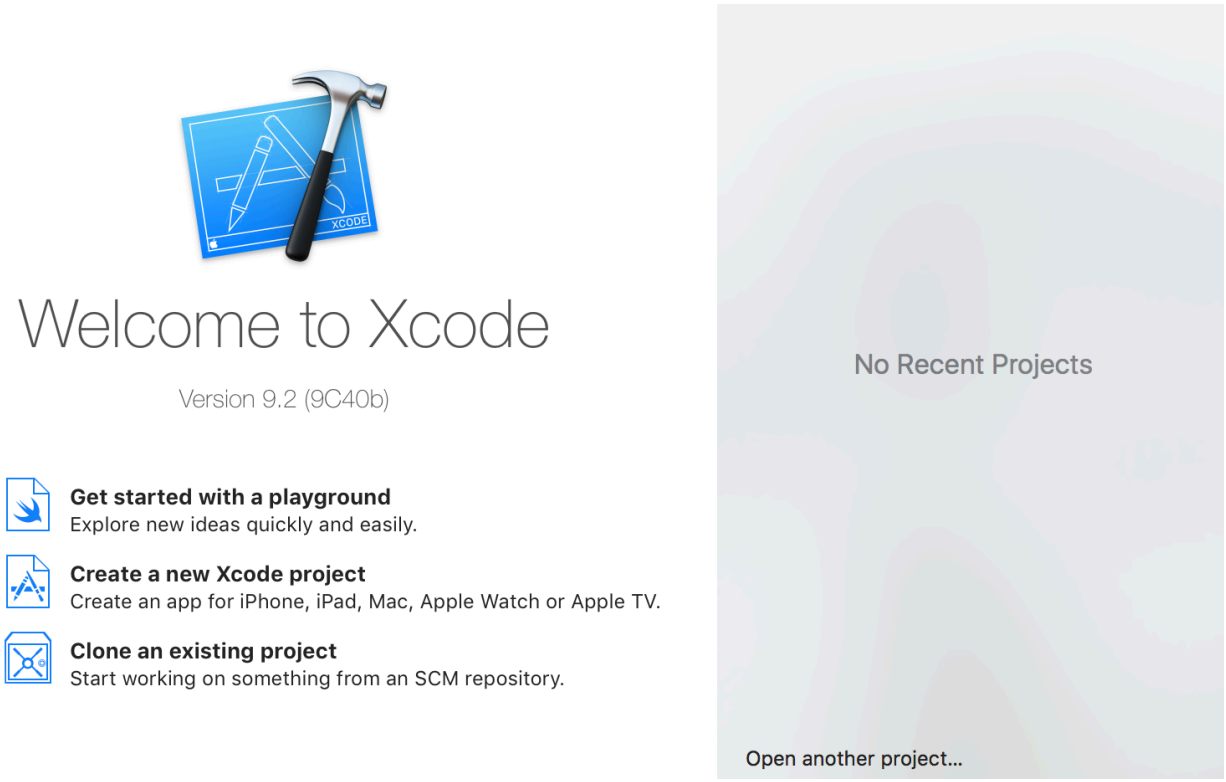
This guide will help you to get started with Xcode integrated development environment (IDE) for C++ programming. If you have any difficulties with this lab, do not hesitate to email your TA. Please note that Xcode is not available on the lab workstations but you may use it on your personal computers to complete and demonstrate your assignments.

A. Introduction to Xcode

These steps do not change from lab to lab and should be learned at your earliest convenience.

A.1 Starting Xcode

1. Download Xcode from the Mac App store.
2. Open Xcode. It will likely be found in your applications folder. If not, you can search spotlight.
3. You will now get a "**Welcome to Xcode**" window. If you launch Xcode and this window does not appear, you can open it be selecting Window > Welcome to Xcode from the menu bar.



A.2 Creating a Project in Xcode (to compile and run programs)

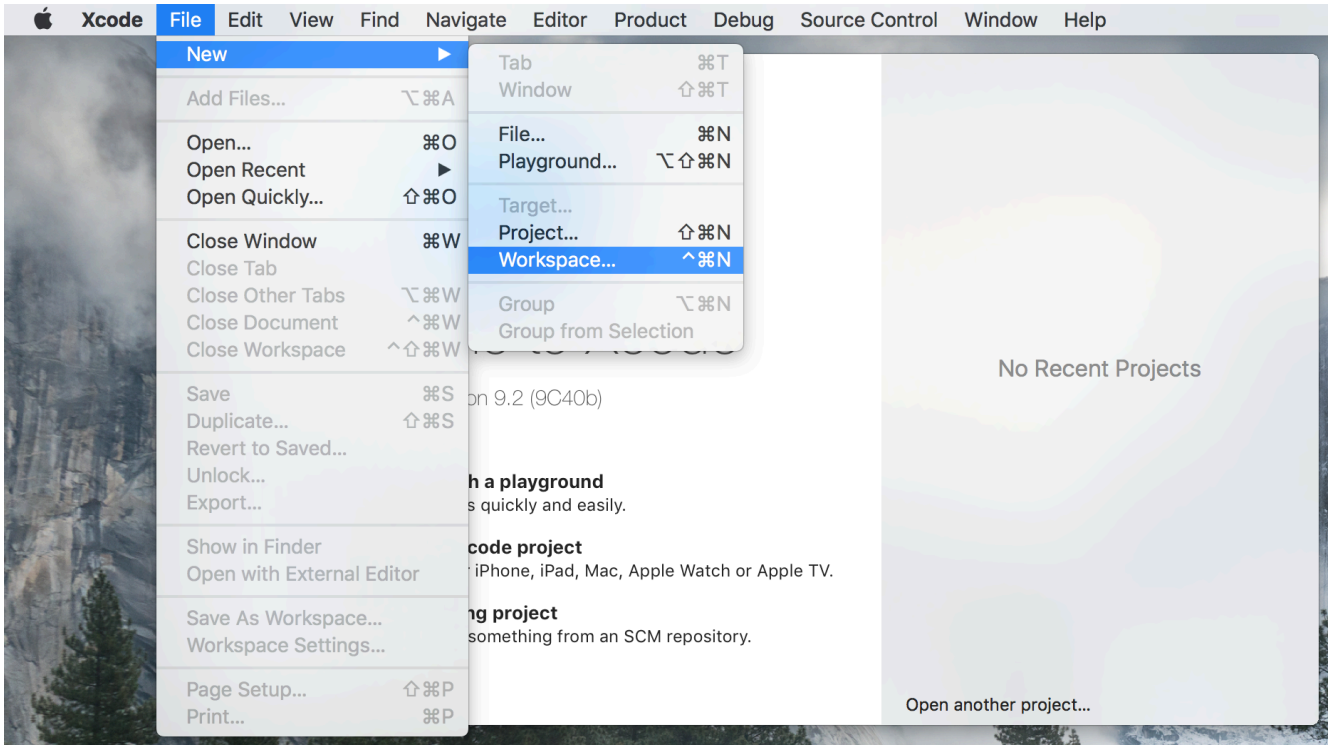
To begin writing a program, we must first create a Project. A project allows us to create and run files inside of Xcode. Different programs (e.g. different labs) can be contained in different projects.

In Xcode you can also make a workspace, *ES1036Labs*, which will contain all labs for this course. (Note: the *Workspace* in Xcode is the same as the *Solution* in Visual Studio).

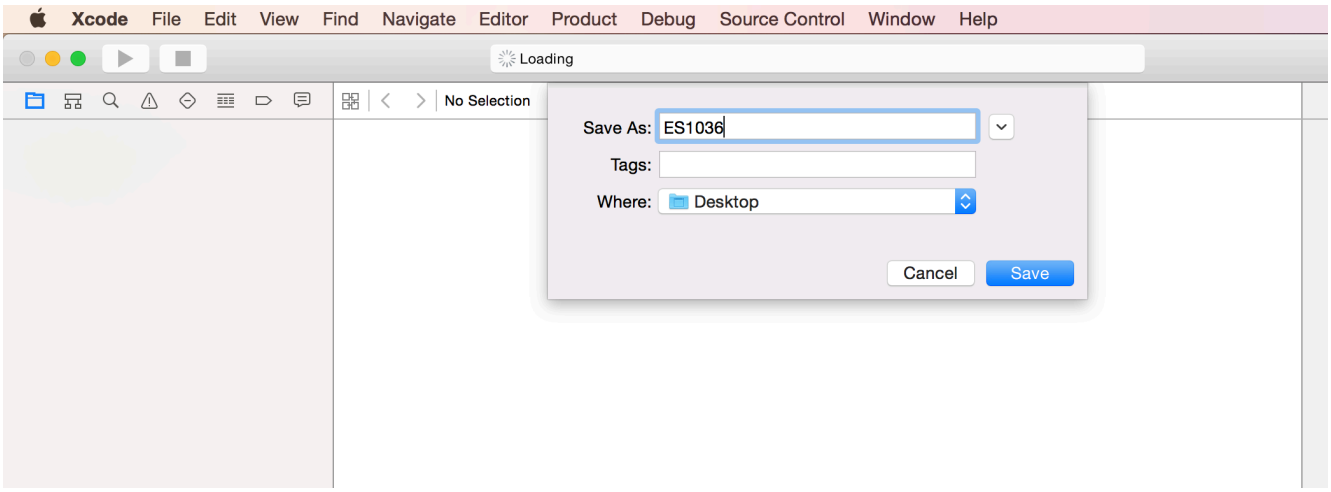
First we will create the workspace *ES1036Labs* and a project called *Lab00*. When you want to work on your Lab 01 next time, you can add another project called *Lab01*.

To create a workspace:

1. Click on the menu bar and follow File > New > Workspace

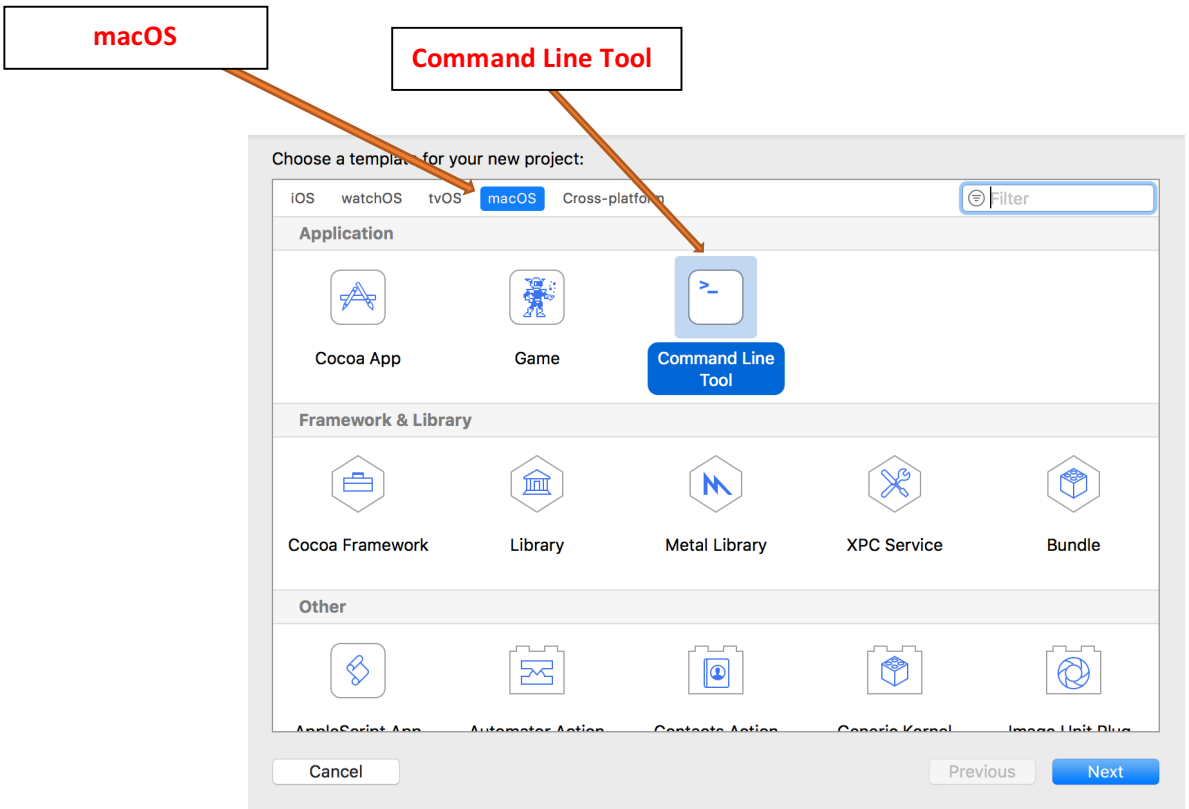


2. Enter the workspace name `*ES1036Labs*' and specify its location on your file system. Then click save.

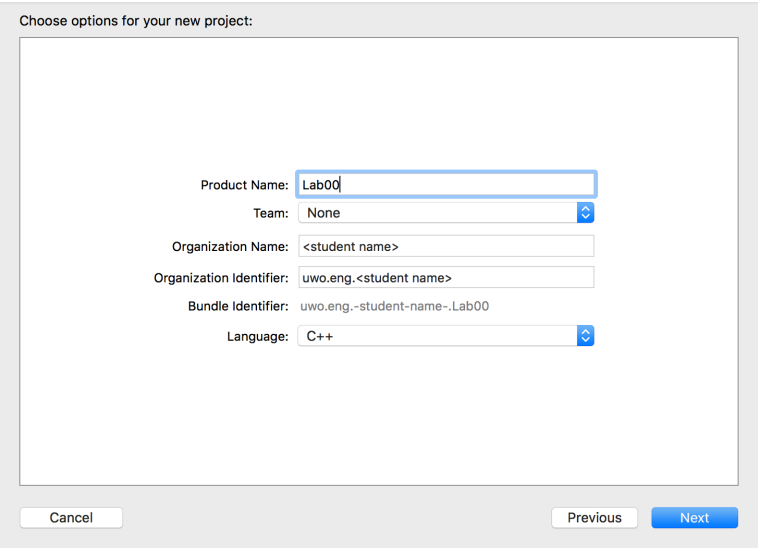


To create a project:

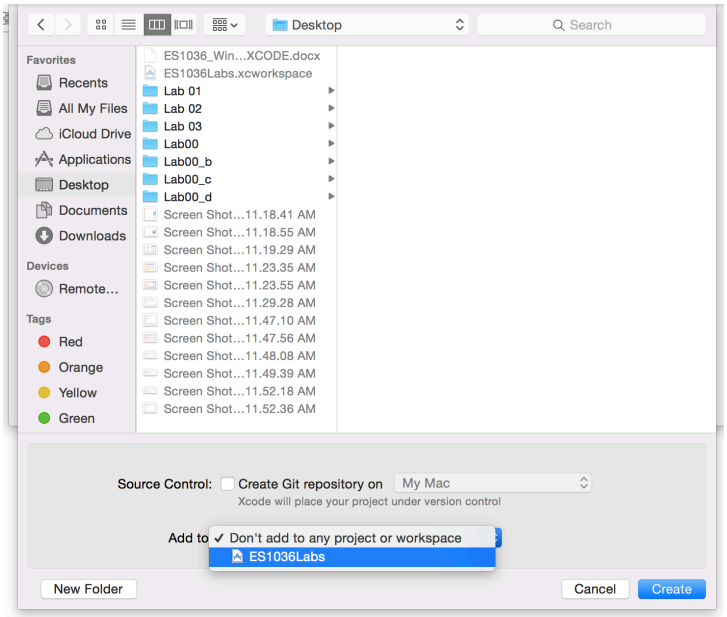
1. Choose File > New > New Project. Select the project template macOS, and then select the Command Line Tool.



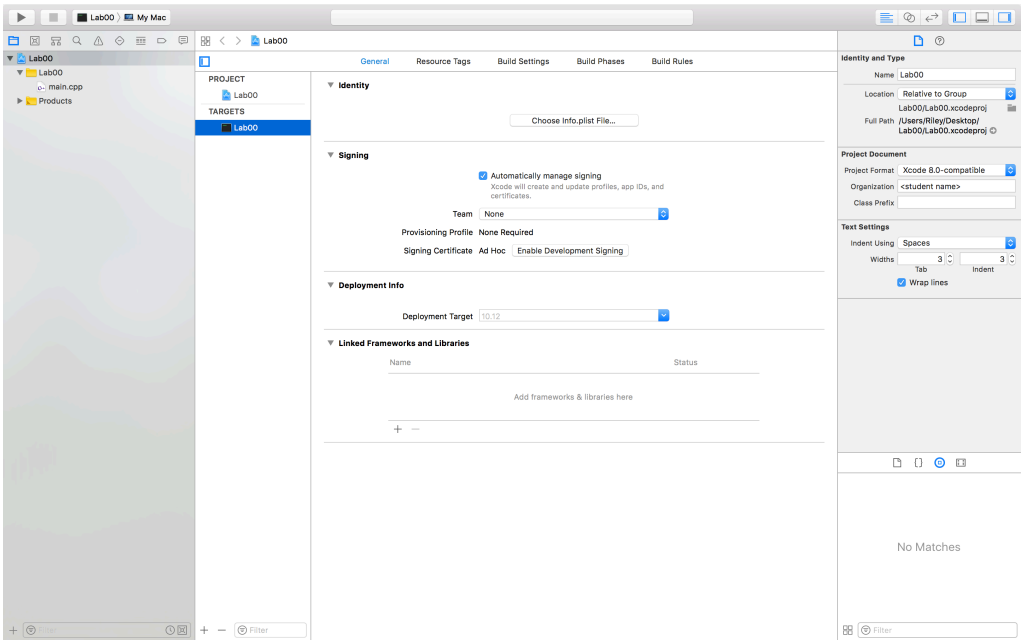
2. Enter your lab name in the Product Name (Lab00) and in the popup Language menu choose C++. For Organization Name write your own name. Company Identifier is what a Mac uses to tell apart various programs. Usually those who have a web server will use its domain name in reverse. But, you can just use uwo.eng for now or simply *ES1036*. Click Next.



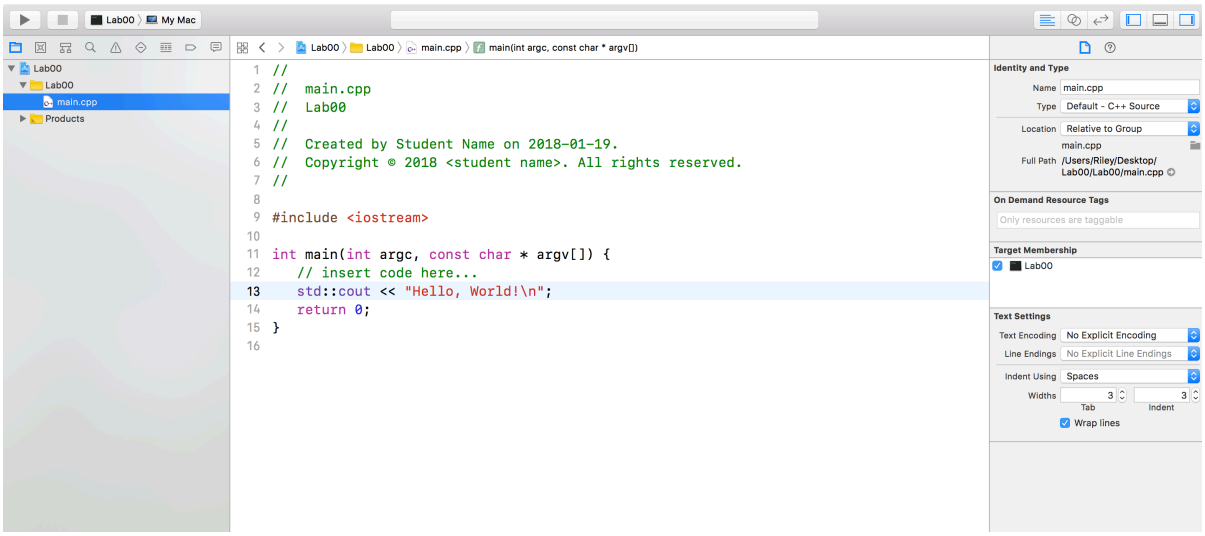
3. Specify the project's location in your file system (same place as you have saved your workspace) and from the popup Add to menu, choose the workspace you have created and click Save.



4. You will now get your project name in a list called Project Navigator on the left side, and to its right more detailed information about whatever is selected on the left.



5. If you now click on the entry “main.cpp” in the list shown in the left panel, you will see a text editor. This is where you will edit text using Xcode as shown. If you click on the main.cpp by default you will see a section of code.



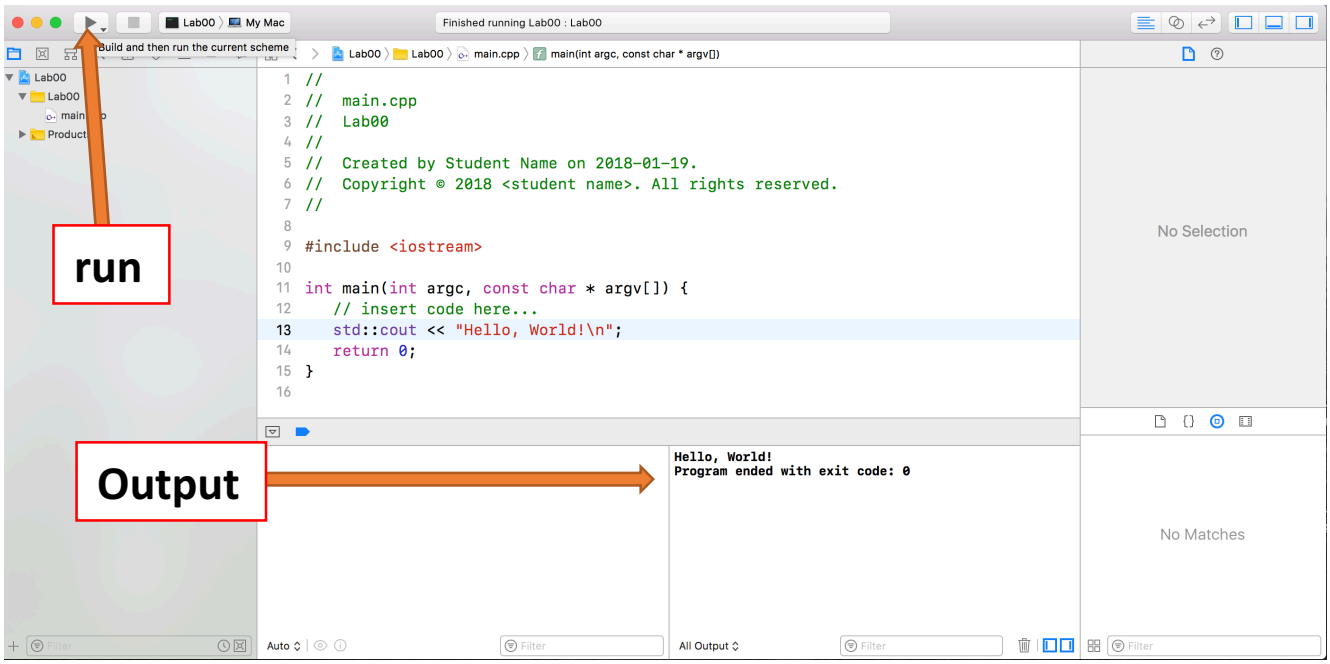
By following the steps above, you will have created the project Lab00. However, this project is still empty as shown in the above window. We need to add what we call "source code" in order to make the project 'do something'. As future programmers you will eventually write source code from scratch, but for this lab (Lab 00) some codes have already been created for you, which are located on the OWL course website.

Adding existing source code to this project:

1. Log into Owl and go to the ENGSCI 1036 page. Click on the **Lab Assignments** link, and then click on the **Lab00** link, where you should see some files with .**cpp** extensions. Download these files and add them to your Xcode project. You can also copy and post the code from your browser to the Xcode source editor.
2. In the Xcode project manager, select the destination project or group for the item you want to add. Click **File** > Add Files to "Lab00" and choose the file to add.

Compile and run (MyFirstProgram_v1.cpp):

1. To compile and run the code, click on the **run** button at the top left of the editor window. If it asks you to save just click save all and it will compile.
2. The output of your program will be in the bottom right corner in the *All Output* window



Reviewing the Xcode Environment

Now let's try everything again from scratch!

1. Go to the ES1036 OWL page and download the ArithmeticOperations_lab00.cpp file to your computer
2. Add the file ArithmeticOperations_lab00.cpp as the source code.
3. Compile and run this code by inputting the integer values 4 and 2.

Modifying the File ArithmeticOperations_lab00.cpp

4. In the file there are four statements that calculate 'result'. Three of these statements have been commented out by using two forward slashes "//" at the beginning of each of these statements. Compile and run the code and observe the result using the integer values 4 and 2 respectively.

5. This time, try to use integer values 2 and 4 respectively and check the output for each case. Try to uncomment the second **cout** statement and comment out the first one and then run the program to check the output. In both cases, the output will be same. This will give you an idea of the **endl** function.
6. Repeat the procedure by un-commenting one 'result' statement at a time by removing the slashes "//"; make sure the other three result statements are commented out.

7. Modify the above program to include the following lines of code just before the line with 'return 0'

```
cout << " 1 / 5 = " << 1 / 5 << endl;
cout << " 1.0 / 5 = " << 1.0 / 5 << endl;
```

Build and run your program. **What is the resulting output of the above code? Why are the results different?**

Finding and correcting code errors based on the compiler messages:

Debugging the code can be both tedious and frustrating when learning a new language. Lots of errors are usually extremely hard to find especially when you don't know where to look for the most common ones. Compiler messages provide useful information on the source and nature of errors. The ability to identify most common error messages will save your time. Let's try including another code to the project.

1. Go to the ES1036 OWL website and download the file common_cpp_mistakes.cpp to your computer
2. Add the file `*common_cpp_mistakes.cpp'* as the source code (as before).
3. When trying to compile this code you will run into errors. If there is more than one error it's always a good idea to start from the topmost error. Correcting a couple of them often removes the total number of errors.

This file contains the same code as Arithmetic_Operations_lab00.cpp with the following errors deliberately made in it:

- missing or misspelling #include <iostream> or using namespace std;
- Forgetting to type the parentheses when calling a function that takes no parameters, like main().
- forgetting to declare the variable or misspelling its name;
- Using the uppercase for lowercase keywords[1] or names, and vice versa.
- Omitting a semicolon (;) or using a colon (:) instead.
- Missing an opening or closing curly bracket { or } . Good practices used to avoid that type of error include (1) putting a closing bracket right next to the newly opened bracket, then working inside, and (2) indenting corresponding portions of the code (it also makes your code look neat!)

Please try to fix these errors after reading the following notes.

Very Important Notes:

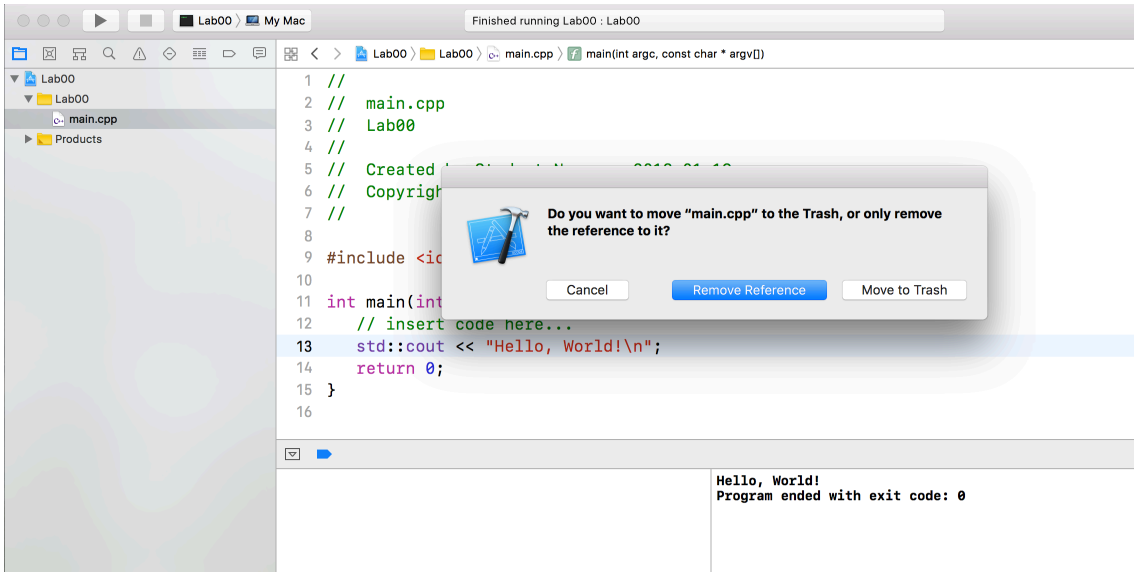
- Fix the topmost error shown and compile your code. After this compilation, if there is any leftover error, fix the topmost error again and re-compile it.
- Repeat the above procedure until all errors are fixed.
- ALWAYS fix the topmost error.
- NEVER try to fix more than one error at a time.
- Some error messages will not make much sense (and some never will), prompting a need to develop an intuition for their origins. Feel free to ask your TA for help if you're stuck.

B. Creating a New Project and a New Program from Scratch

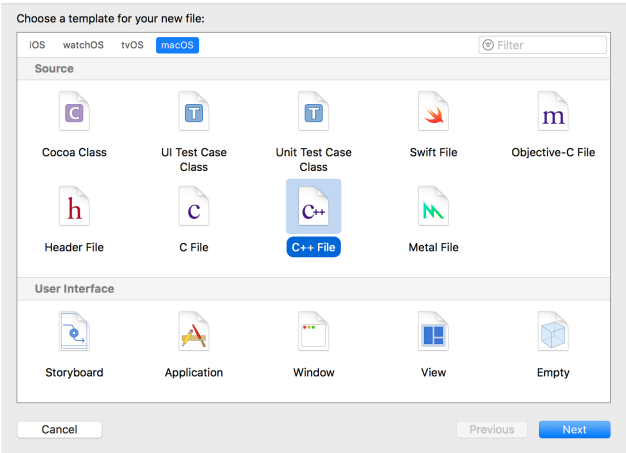
After completing this section you will be able to create a new program in C++ and add it to a project in your existing solution.

1. Under the File menu, click on **New -> New Project** as described previously.
2. Enter the name **Lab00_b** for the project. You should put the project in the workspace you have already created, *ES1036Labs*
3. Now you will see two projects in the Navigation area, to choose a project and run it, first you should specify that project. To do this, just in front of the Run button you will see a small arrow, click on it and chose the project (lab00 or lab00_b), then click Run button.
4. To close a project, just highlight on the project and from menu bar choose **File > Close**.
5. Now we want to create some source code from scratch. To do this, we first need to delete the file *main.cpp.* In the navigation area*, r*ight click on the file and choose delete. The click on *Remove Reference.* This option will ONLY remove the file from the project files list and keep it on the hard drive intact.

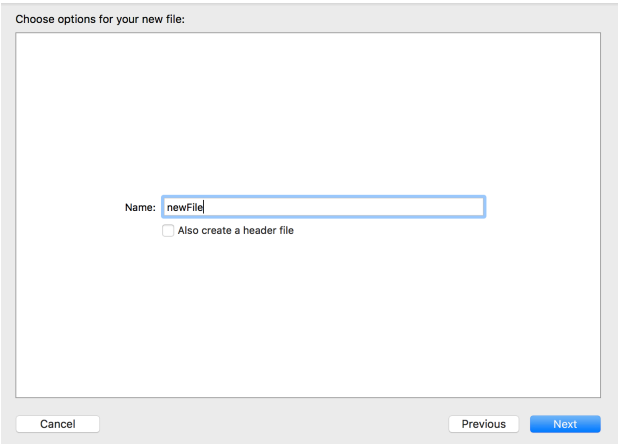
[1] The words which are reserved for a particular programming language (grammar) are known as reserved words; also known as keywords. Each of these words has a particular meaning to the programming language.



6. Click on the Lab00 project and right click on the lab_00 yellow folder. Select New File…
7. Select the C++ class file to add a .cpp file to your project.
8.



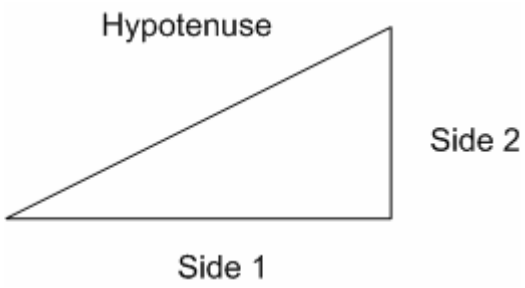
** Be sure to uncheck create header file



9. Type the name of you .cpp file (ex. *Test.cpp*). You are now ready to begin your program.
10. Type in the following code:

```
#include <iostream>
#include <cmath> //ask your TA why are we using it

using namespace std;

int main()
{
   // Declare and Initialize variables
   double x1 = 1, y1(5), x2(4), y2 = 7,
       side1 = 0, side2 = 0, distance = 0;

   // Compute the sides of the triangle
   side1 = x2 - x1;
   side2 = y2 - y1;

   // Calculate distance
   distance = sqrt(side1*side1 + pow(side2, 2.0));

   // Print the distance
   cout<<"The distance between points ("<<x1<<','<<y1
       <<") and ("<<x2<<','<<y2<<") is "<<distance<<endl;
   return 0;   // indicate that the program ended successfully

} // end function main
```

11. Replace the first set of comment lines with a set of comments that identifies you (name, student number, section, email address). Run the program
12. Modify the program (save it as a second file with a different name as a backup) so that the program also prints out the perimeter of the triangle inscribed by side 1, side 2, and the hypotenuse (distance). Note that your program should still output the value of the perimeter.



13. Compile the program and execute.

14. Call your TA to your computer so that she/he can verify your result.

Modifying your program to use cin

Note that in the above program, when the variables x1, x2, y1 and y2 were created, they were assigned values 1, 5, 4, 7 respectively. This is known as initializing the variables. Now you will modify your program by asking the user to enter in these values from the keyboard (standard input). The figure below gives the code that is required to ask the user to enter the value for the variable x1.

```
int main()
{
   // Declare and Initialize variables
   double x1 = 1, y1(5), x2(4), y2 = 7,
       side1 = 0, side2 = 0, distance = 0;

   // prompt the user to enter x1 value
   cout<<" Enter x1 value: ";
   cin>>x1;

   // Compute the sides of the triangle
   side1 = x2 - x1;
   side2 = y2 - y1;

   // Calculate distance
   distance = sqrt(side1*side1 + pow(side2, 2.0));
```

15. Modify your existing program to ask the user to input values for x1, x2, y1 and y2, and then calculate the perimeter and hypotenuse of the triangle.
16. * Submit **this final file (ONLY)** via Owl course website.
    WHEN SUBMITTING ANY LAB ACTIVITY YOU NEED TO SUBMIT THE .cpp FILES ONLY. DONOT SUBMIT ANY OTHER TYPES OF FILES, OTHERWISE YOUR WORK WILL NOT BE MARKED.
17. To do that you will need to identify the locations and names of the files to be submitted. If you're not comfortable using Owl ask your TA for the instructions.

C. General Lab Instructions to Help Labs Run Smoothly

- Attending all the labs is mandatory.
- Obtain the lab instructions from the ES1036 OWL web page.
- Read through the lab instructions and work on the assignment before coming to the lab.
- Write the quiz at the beginning of the lab and waiting for the quiz time to finish.
- Highly recommended: retain copies of the .cpp files that you create or submit via Owl

D. Good Programming Practice

- Include comments in your program.
- Choose meaningful and descriptive names for the variables. **See the attached document "Naming Conventions" in your *Assignment* folder**
- Initialize your variables appropriately.
- Indent your code. XCode is quite smart and automatically indents your code as you type. If you see that the indentation is modified as you code do the following simple steps:
- Highlight the portion of the code you want to re-indent
- Right-click on the selected text
- Select "Structure" from the drop-down menu
- Select "Re-indent section" from the sub-menu