CIS121 Laboratory One

Objectives of this lab:

In this laboratory session you will do the following things:

1. Install Microsoft Visual C++ Express Edition on your laptop.
2. Enter a C++ program, compile it and execute it.
3. Find and correct compiling errors.
4. Investigate output in C++.

Part 1: Building a Microsoft Visual C++ Project

After we write a C++ program, we need to be able to compile and run it. In order to do this, many things need to be set up. In Microsoft Visual C++ this is done using something called a "Project". The Project contains all of the C++ files needed to make the executable program (large programs are often made of many individual C++ files). The project also has all of the "options" which tell the compiler how to build the program. The Project must also have room to store temporary files.

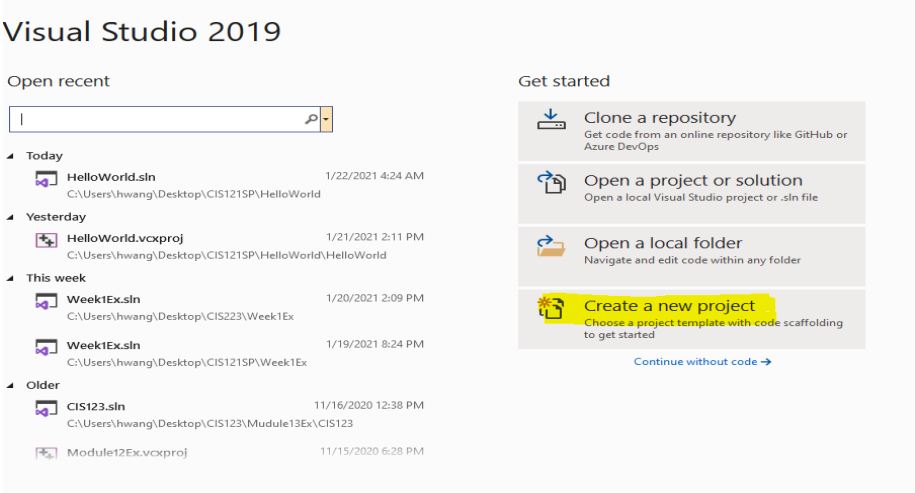
Use the following steps to create a project:

1) Create a folder that will contain the project.

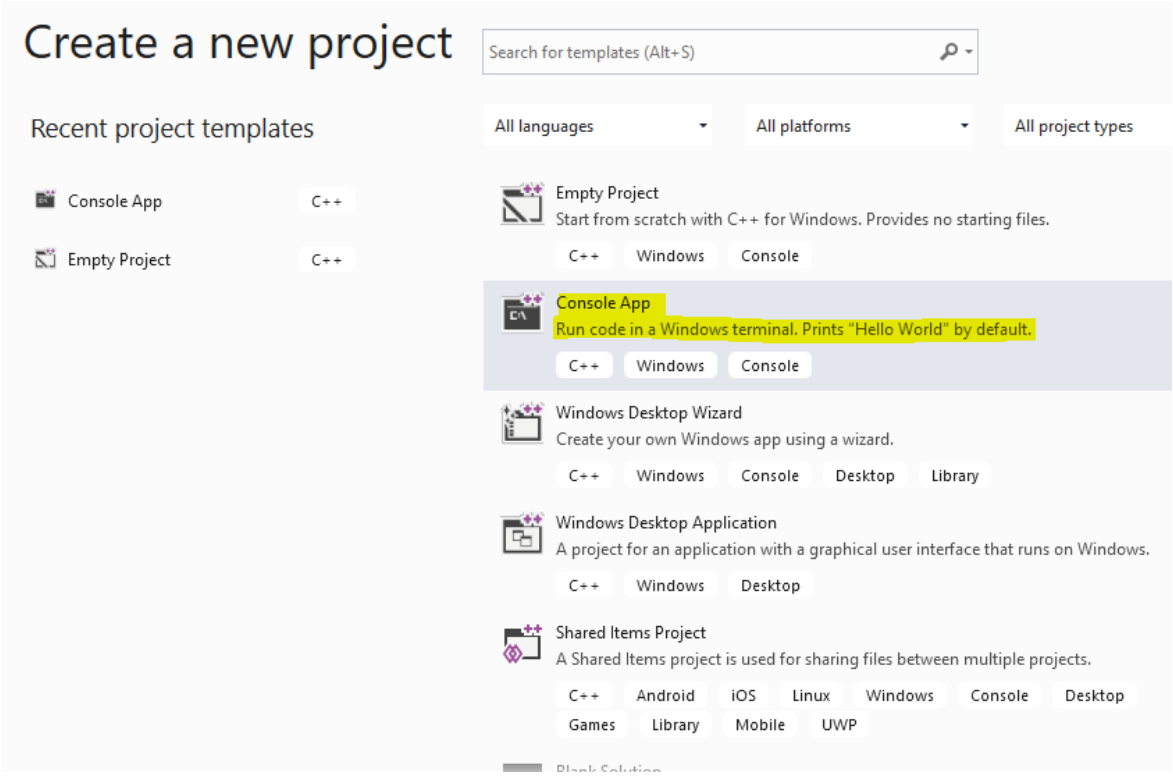
To create a folder is the C:\ drive.
Open C:\, right click and select New->Folder. A new folder will appear.
Rename the folder (you can use CIS121)

2) Create the project

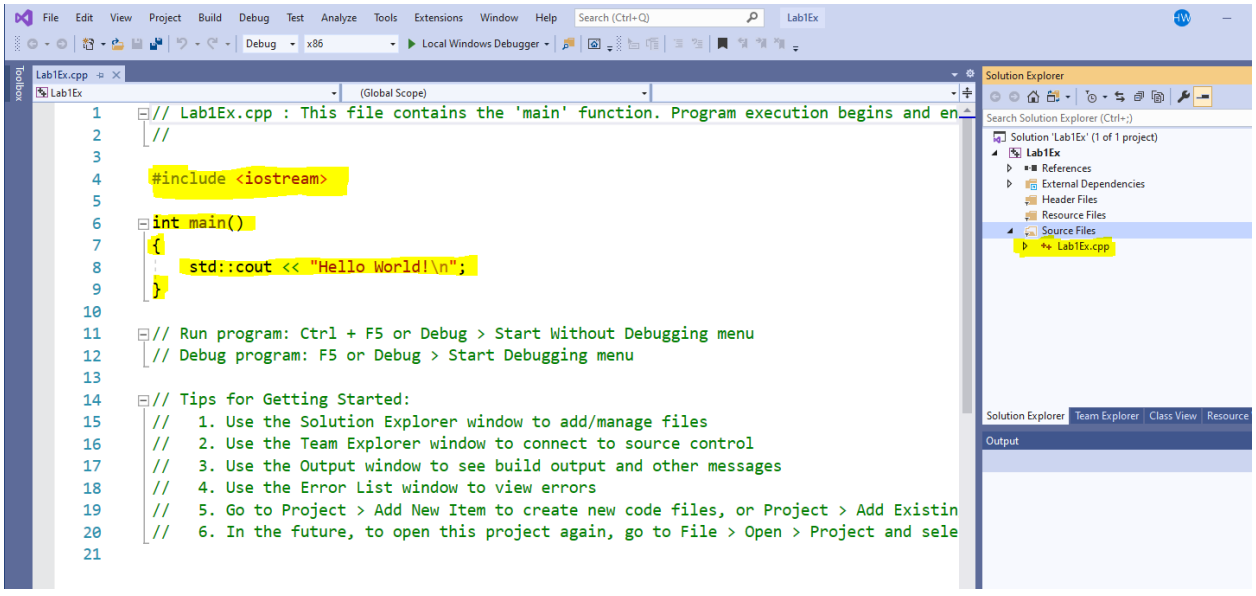
a. Select **Create a new project** ->Console App->Project name



b. Choose **'Console App'**



c. Name your project



You now have a project that you can use to compile your programs.

3) To create a new C++ file in the project

- Select Project ->Add New Item…
- Click on Code
- Click on C++ File (.cpp)
- Specify the name of the file
- Specify the location of the file (it will default to the project folder, this works well)
- Click on Add
- You can now type in C++ code and run it by selecting Debug Start Without Debugging.

4) To open an existing C++ file into the project:

- Select Project->Add Existing Item…
- Navigate to your .cpp file.

You can now run your file by selecting Debug->Start Without Debugging.

5) To open an existing project:

- Select File->Open ->Project/Solution…

Navigate to your workspace file (it will be a .sln file) and open it.

Experiment 1.1 Type the following program in your computer, compile and run it.

```
#include <iostream>
using namespace std;
int main()
{
        cout << "Hello World.\n";
        cout << " Everyone!\n";
        return 0;
}
```

Part 2: A Simple C++ Program

Let us examine the program listed below

```
// This is our first C++ program.
#include <iostream>
using namespace std;
int main()
{
  cout << "What is your favorite\n";
  cout << "flavor of ice cream?\n";
  return 0;
}
```

In addition to statements that will ultimately be translated into machine language, a C++ program can contain explanatory remarks for the aid of human readers (like your professor). These remarks, known as comments, are placed after the characters //. In turn, the C++ compiler ignores everything appearing on the same line after these special characters. In our example program, the first line

// This is our first C++ program.

is a comment. Comments have a variety of uses. They can be inserted to clarify a section of a program that might otherwise be hard to understand. They can also be used to give information about the creation of a program such as the date created and by whom.

The statement

#include <iostream>

in our example program is an example of a preprocessing directive. These directives cause the source program to be modified before the compiling process begins and are signified by beginning with a # symbol. The directive in our example causes a copy of the standard header file named iostream to be inserted at the beginning of the source code when compilation occurs. This file contains information that the compiler will need to perform its task. In particular, our example refers to the special object cout (which we will explain shortly) through which information can be sent to the monitor screen. Our program does not contain the details of cout but merely communicates with the object by means of the operator <<. The file iostream contains the information needed by the C++ compiler to create the required link between our program and the system library.

C++ uses *namespaces* to organize the names of program entities (variable, functions, and objects). The statement

using namespace std;

declares that the program will be accessing entities whose names are part of the namespace std. The program needs to access the std namespace because every name created by the iostream file is part of that namespace. Therefore, in order for a program to use the entities in iostream, it must have access to the std namespace.

A C++ program consists of units called functions. (We will learn about functions shortly.) Every program contains at least one function called main. Execution of a C++ program always begins in the function main. That is, the function main represents the beginning of the program even though the function may appear much later in the written program. The line

int main()

in our example indicates the beginning of the function main. This opening line of a function is known as a function header. We'll learn more about function headers in later laboratory sessions. For now we note that such a header consists of three parts: a return type (in our example int), the name of the function (in our example main), and a parameter list (in our example an empty parenthetical expression). The fact that the function header in our example has a return type of int indicates that the operating system should expect to receive a numeric (integer) value when our program terminates. This value is used to report whether our program executed successfully. We'll return to this point shortly.

The actual "meat" of a function is placed between braces. Immediately following the opening brace is the declarative part of the function. It is here the terminology relating to that particular function is introduced. Our example program is so simple that it does not contain a declarative part. Instead, the function **main** in our example consists of only a procedural part—the part containing the instructions to be followed when the function is executed. The procedural part of a function always follows the declarative part.

Statements in a C++ program are terminated by **semicolons**. Although not mandatory, it is customary to place each statement on a separate line and to use indentation to help the reader identify related portions of the program.

Each statement in the procedural part of our example program uses the predefined object cout and the operator <<. The object cout is just one of many standard units that are so commonly used in programs that C++ provides them for your use in standard C++ libraries. You will learn more about cout in later laboratory sessions. For now we need merely note that the statement

```
cout << "What is your favorite\n";
```

causes the characters that are enclosed in quotation marks to be printed on the monitor screen. Thus, when executed, our example program will cause the two lines

```
What is your favorite
flavor of ice cream?
```

to appear on the screen.

The last line in the function main

```
return 0;
```

indicates that our program has finished its task and that control should be returned to the operating system. As indicated in the function's header, the operating system will be expecting to receive a numeric value indicating whether our program executed successfully. This is the purpose of the value 0 in the return statement. In general, returning the value 0 means that the program executed successfully; other values are used to indicate various errors.

Experiment 1.2

Step 1. Omit the following line

```
#include <iostream>
```

from the program in Experiment 1.1 and try to compile the modified version. Record what happens.

Step 2. Remove both cout statements from the modified program and try to compile the program. Explain the results.

Experiment 1.3

Step1. Insert the following lines into the procedural part of the function main in Experiment 1.2. Explain how the output produced by each of these statements differs from the others.

```
cout << "Chocolate, butterscotch, strawberry, vanilla?\n";
cout << "Chocolate, butterscotch,\nstrawberry, vanilla?";
cout << "Chocolate, butterscotch\n\nstrawberry, vanilla?";
```

Step 2. What does the \n character combination mean?

Experiment 1.4

Insert the following lines into the procedural part of the function main in Experiment 1.2. What rule can you derive about the placement of comments?

```
cout << "Send money quick!\n";  // To Mom!
cout << "Send money quick!\n"  // To Mom! );
cout << "Send // To Mom! money quick!\n");
```

Experiment 1.5

Step 1. Insert the following lines into the procedural part of the function main in Experiment 1.2. What rule can you derive about printing sentences containing quotation marks?

```
cout << "Oh, I love to program in C++...\n";
cout << "Oh, I love to "program" in C++...\n";
cout << "Oh, I love to '"program"' in C++...\n";
cout << "Oh, I love to \"program\" in C++...\n";
```

Step 2. What is the meaning of the backslash mark?

Part 3: Programming Exercises

- **Only One question is required for lab 1.**
- **Send the source code and output to the Dropbox on D2L.**

1. Write a program that will display your name on the first line, your street address on the second line, your city, state, and ZIP code on the third line, and your telephone number on the fourth line. Place a comment at the top of the program… Then compile and run the program. The output should look something like this:

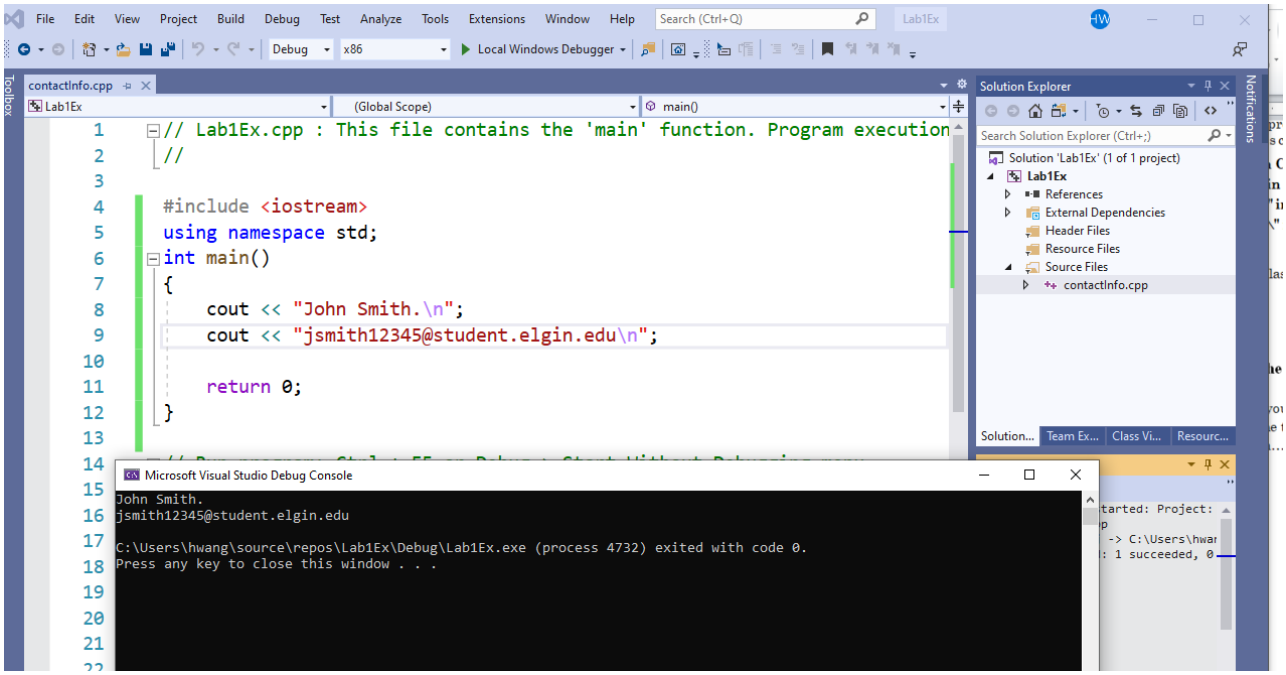
```
Jane Doe

847-214-5555
Jane.Doe@students.elgin.edu

*
***
*****
*********************************
Computer Science is "Cool" staff !
*********************************
*****
***
*

Good-by!

This is the end of the program.
```

2. [Extra Exercise] Write a program that allows the user to enter a time in seconds and then outputs how far an object would drop if it is in freefall for that length of time. Assume that the object

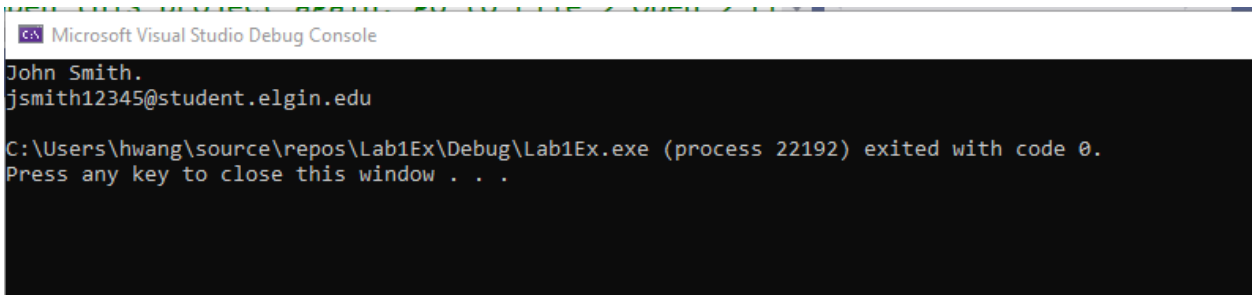
starts at rest, there is no friction or resistance from air, and then there is a constant acceleration of 32 feet per second due to gravity.

Submit:

- Select "***Assignments***" from the choices in the menu on the top of the screen.
- Click on the specific lab/assignment to be submitted.
- Under ***"Submit Assignment"*** there is an '***Add a File***' option. Select the "***My Compute"*** button and locate your source code file **(.cpp) and the output screenshot (.png)**
- After the file has been attached, click the "***Submit***" button.



To take screen shot: use **Snipping Tool and save as .png file**



Part 4: Things to note

1) If you double click on your C++ file it will open the file in a text editor, HOWEVER, no Project will be open and therefore you will not be able to compile or run your program.

2) Always access the My Documents folder through the U:\ drive. The Microsoft Visual Studio software does not like the logical link that points to My Documents.

3) You can put an empty Project on a flash drive and make a copy whenever you need it.

4) You can reuse a Project. Just remove the C++ file you were working on from the Project and add or create a new file.

5) If you create the wrong kind of Project (for example a Win32 Application instead of a Win32 Console Application), everything will work fine until you go to build, then you will get an error like "error LNK2019: unresolved external symbol WinMain@16")

6) The Solution Explorer window that is on the right side of the screen shows the files that are part of the project. When you build the project, those are the files that get built. IT DOES MATTER WHAT FILES YOU MAY BE LOOKIN AT IN THE EDITOR WINDOW!

7) If you do a "Save As…" to create a new file, the new file WILL NOT be added to the project. When you build, you will build the old file, not the one that you are looking at in the editor.

8) If you are running a program and you try to build and run it a second time without stopping the first run you will get the following error "LINK : fatal error LNK1168: cannot open Debug/ConsoleApp.exe for writing": If you somehow get two programs in one project you will get the following error when you try to build the project:

    Hello2.obj : error LNK2005: _main already defined in Hello.obj
    Debug/ConsoleApp.exe : fatal error LNK1169: one or more multiply defined symbols found

10) Do not use a "." in any file names. Visual C++ puts the proper suffix on all of the files (for example all of your C++ files will have .cpp at the end) HOWEVER, if you put a period in a file name it will think that you want to over-ride the default suffix. This will prevent things from working properly.

11) To print the output in the console application window (the black window) go the upper left corner of the window to the C:\ icon and click on it. A menu will come down, select Edit -> SelectAll. Go there again and select Edit ->Copy. Now open the Notepad editor and paste the text into the editor window and print the text from there. If you use some editor other than Notepad, make sure you use a fixed width font like Courier.

12) If the menu items that you want do not appear in the menu, you may have the wrong window selected. For example; if you want to add an item to the project, you must have the Solution Explorer (Project) window selected.

13) If you load the Express version of Visual Studio you will not have the “Start Without Debugging” menu option. To add to you need to do the following: Under the commands Tab, select Toolbar, then select the debug toolbar. On the right side, you should be able to select click Add Command, the Add command dialog should appear. Select Debug on the left side, and start without debugging in the right side.