Programming Fundamentals I
COSC 1336

Lab 1. Introduction to Visual Studio IDE

Objectives:

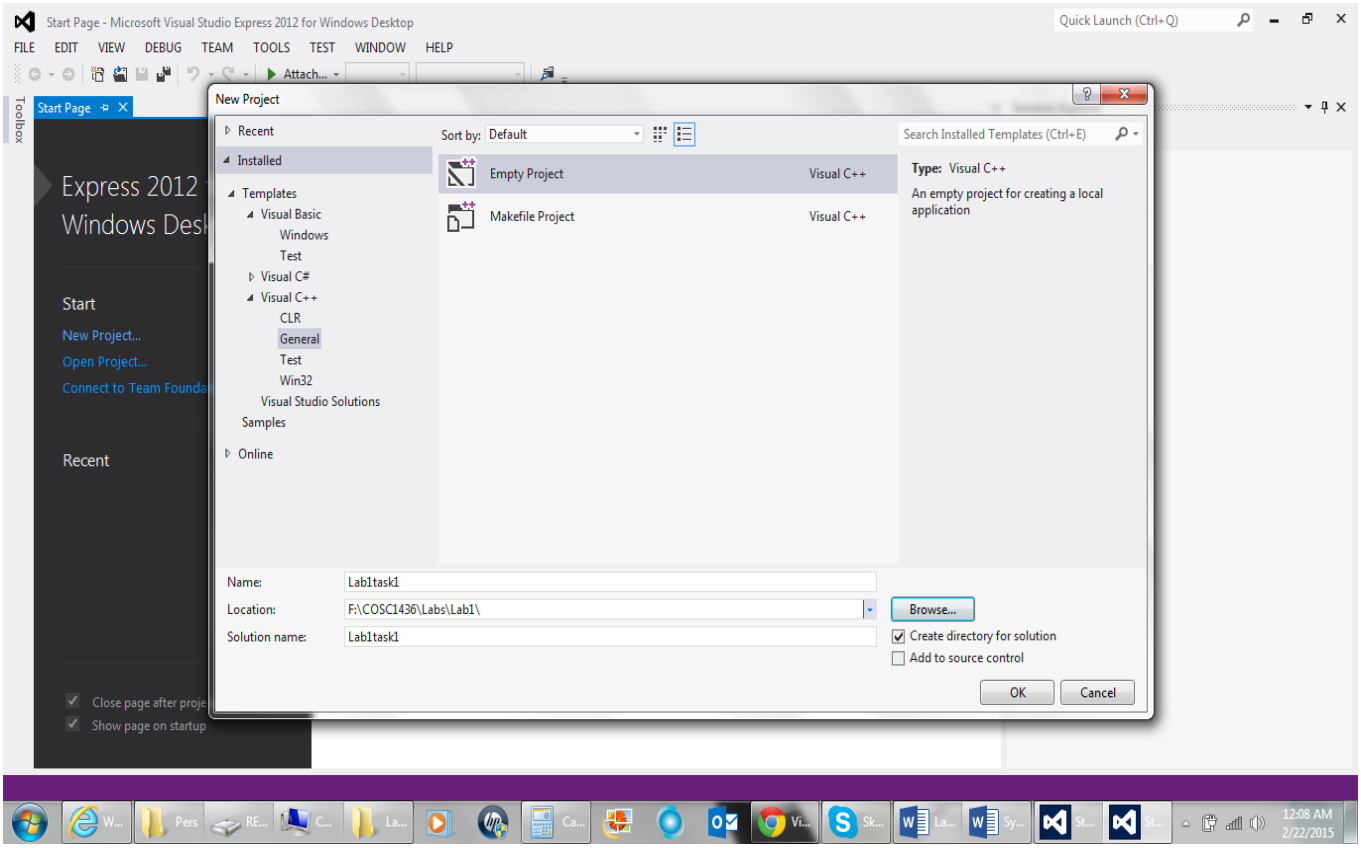
1. To become familiar with Visual Studio IDE.
2. To be able to create a project, add a source C++ file, build the project and run it in the IDE.
3. Be able to run your program from outside the IDE.
4. To be able to recognize and correct the three type of programming errors: syntax errors, run-time errors, and logic errors.

Make sure to demo your work to your instructor for each task to get credit.

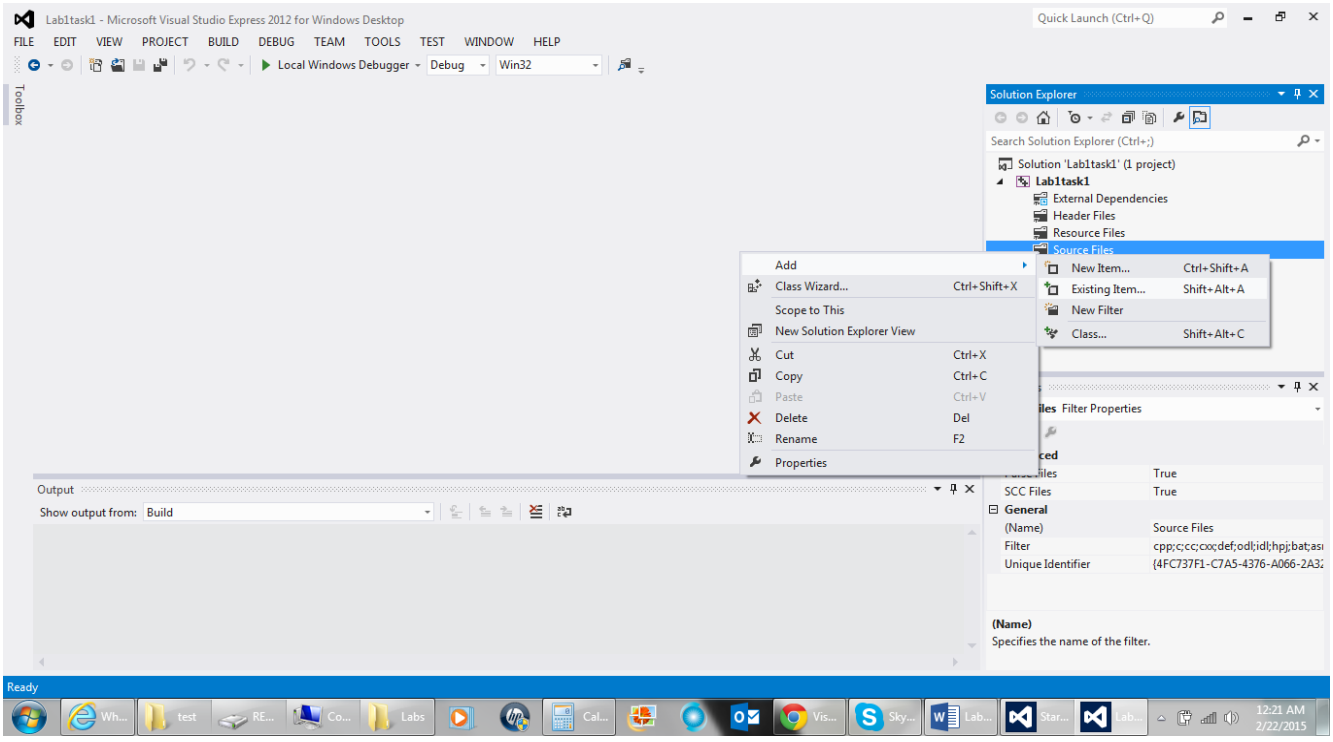
Task 1: Introduction to Visual Studio

1. Create a folder named Lab1 in your flash drive. Copy file *lab1task1.txt* from *D2L—Unit/Chapter 1 Lab* into the new folder.
2. Open Visual studio and create a new C++ project named Lab1Task1 in the new folder. See sample screen shot below for selected project options.

Note the options selected: (1) language: Visual C++, (2) type of project (General, Empty Project), (3) name of project (Lab1task1), and (4) project location should be in your flash drive.



3. Add a new source file to your project:
    - Under the **Solution Explorer** (top right of your screen. See screen shot below) right-click on **Source Files→ Add→New Item**.
    - From the next screen, select **C++ file (.cpp)** and enter a name for the file (e.g. Lab1task1). The new file will be listed under the **Source Files** folder under the **Solution Explorer** with a **.cpp** extension.



4. Copy the code in file *lab1task1.txt* into the new source file you just added to your project. You can do this by copy and paste.

5. Build the project

6. Run it. What's the output?

    _____

    _____

    _____

    _____

7. Save your project.

8. **Demo your output before you quit.**

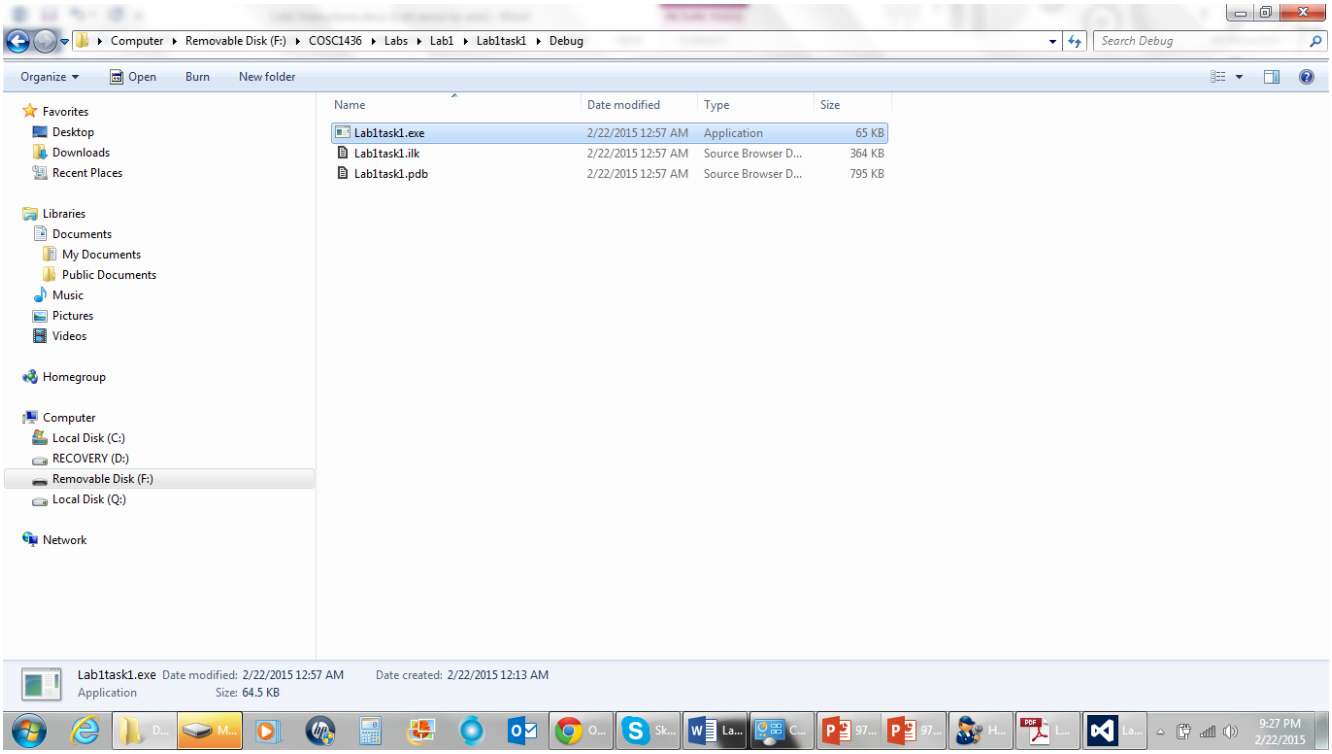
Task 2: Running a program outside the IDE

When you build a project, Visual studio compiles the source files into object files and then uses the object files with other Visual Studio library files to create an executable file. The executable file is saved in your project file by default in the Debug subfolder. By default, the executable file has the same name as the project with extension **.exe**. So the executable generated in the previous task is **Lab1task1.exe**. You don't need Visual Studio to execute an executable file. In fact you can copy the executable file out of the project folder and execute by simply double clicking on it. The objective of this task is to show you how to do that:

1. Using Windows Explorer, navigate to the subfolder where your created and saved your project.
2. Find subfolder **Debug.**
3. One of the files in subfolder **Debug** has extension **.exe**.

   Note: by default Windows Explorer hides extensions for known file types. You can turn-off this option using the **Control Panel→ Appearance and personalization→Folder Options→Show hidden files and folders**, and unchecking the "Hide extensions for known file types" from the drop down menu. To verify the file type, your can also right-click on the filename and select Properties from the dropdown menu.

4. To run the executable file from outside Visual Studio, simply double click on the filename.



Task 3: Syntax Errors

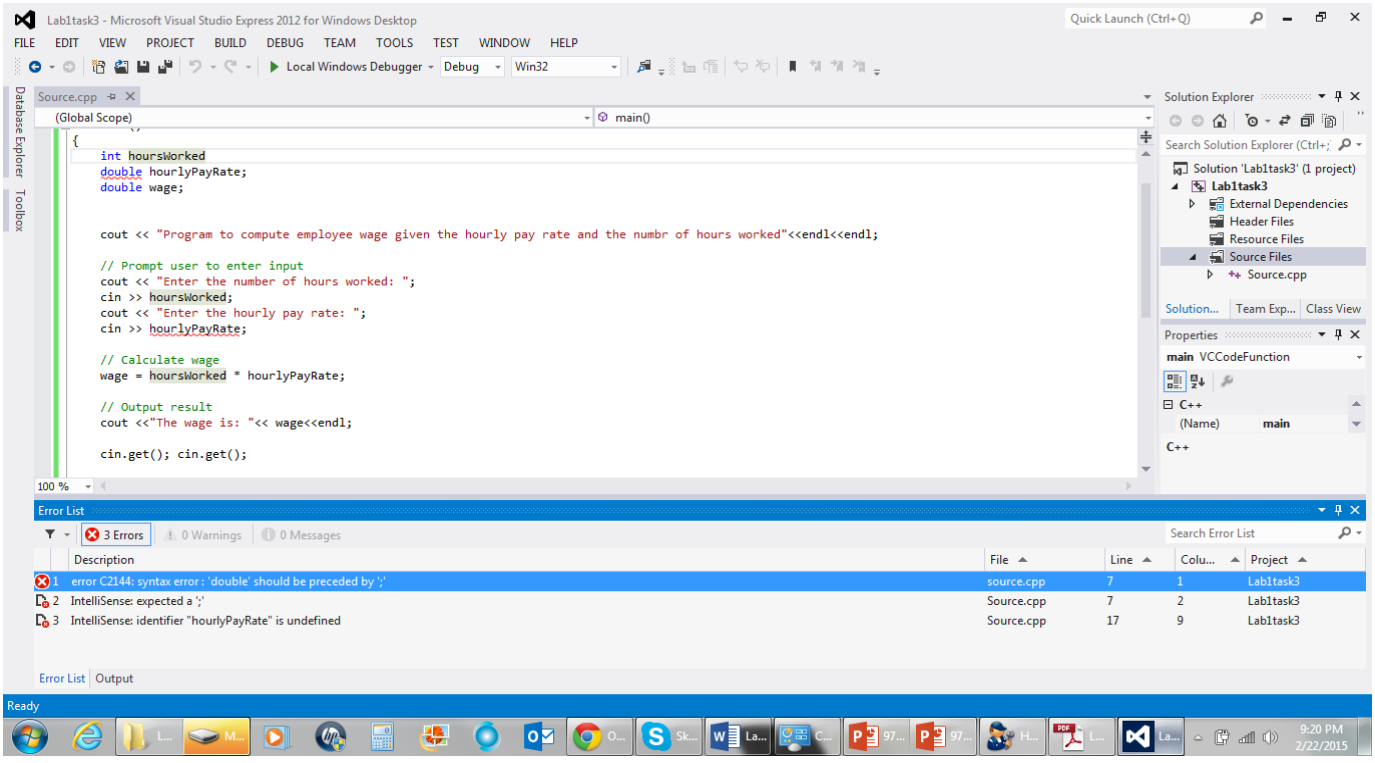
1. Create a new project, named Lab1Task3 in your Lab1 folder.
2. Download lab1task3 from D2L lab1 folder and add to the Source Files of your project and a C++ file.
3. Build your project.
4. What happened? We have our first example of several types of syntax errors.
5. Use the first line in the **Error List** pane at the bottom of the screen under Description to fix the error (see screen shot below).

   Note that

   (1) Even though the compiler reports 3 errors, there is only one syntax error. The others are caused by the original error because even when the compiler encounters a syntax error, it continues compiling the rest of the program. Consequently, you should always fix syntax errors in the order they are listed, and after fixing each error rebuild you program. Because fixing one error usually eliminated several others, you save yourself time using this strategy

   (2) If you double click on the description of the error, the cursor will jump to the line that caused the error.

   (3) Note the red wiggly line underneath the word ***double***. A red wiggly line indicates a syntax error is caused by the underlined word or something related to it.
6. After fixing the syntax error rebuild your program.
7. Run program and report output (**demo**).



Task 4: Run-Time Errors

1. Create a new project Lab1task4 and add file lab1Task4 from D2L lab1 folder.
2. Build your project. You shouldn't get any syntax errors.
3. Run the project.
4. Described what happened.

ANSWER IS ALWAYS "1.#INF"

5. Note that the error occurred during program execution, so it's called **run-time error**. Run-time errors occur when a statement in the program has an error (e.g. division by 0) that prevents the computer from proceeding with program execution (program is also said to have crashed).
6. Fix the error, rebuild and run program again.

Task 5: Logic Errors

1. Create a new project Lab1task5 and add file lab1Task5 from D2L Lab1 folder.
2. Build your project. You shouldn't get any syntax errors.
3. Run the project. Any run-time errors?
4. When prompted enter a sample test data e.g. 70, 80, and 90.

   What is the expected output? 80

   What is the output displayed by your program? 80
5. This is an example of a logic error in your program. Logic errors are caused by errors in your solution to the problem e.g. your algorithm (i.e. the sequence of steps you devised to solved the problem) is wrong or it can be caused by using a wrong formula or making a mistake in an expression. Logic errors are obviously the most difficult to debug. Can you figure out what caused the error in project Lab1task5? average_=_score1_+_score2_+_score3/numberOfScores; SHOULD BE ENCLOSED BY "(" ")" BEFORE "score1" AND AFTER "score3". ALSO "a" SHOULD BE DELETED FROM THE END OF LINE 21.
6. Fix the logic error, rebuild and run program again. Don't forget to demo your work for credit.

Task 6: Flow Charts

1. Watch Video Tutorial 1.1 under Unit1 (D2L).
2. Create a flow chart for the program from Task 3. You can use a word processor to create your flow chat.
3. Upload your flow chart to the dropbox under Lab1 on D2L.