**Programming Assignment 1  
Getting Started with C++**

**Time due: 9:00 PM Tuesday, October 6**

The purpose of this assignment is to have you start learning how to use the Visual C++ and either Xcode or g++ environments and understand a variety of programming errors.

Here's what you are to do:

1. (optional) Obtain a copy of [Visual C++](http://web.cs.ucla.edu/classes/fall15/cs31/visualcpp.html) and install it. You don't need to do this if you prefer to do your Visual C++ work on the SEASnet computers (in the lab or remotely). If you will be using another development environment (such as [Xcode](http://web.cs.ucla.edu/classes/fall15/cs31/xcode.html) on a Mac), use this project to get comfortable using it as well as Visual C++.
2. Enter [this C++ program](http://web.cs.ucla.edu/classes/fall15/cs31/Projects/1/spec.html#program) into your development environment. Do not change the program.
3. Build the executable from the program. (Fix any typos you may have made when entering the program.)
4. Execute the program with a variety of reasonable input integers to see if it runs as one would expect from reading the source code. (If the Visual C++ console window disappears when your program finishes executing, before you have a chance to see the output, you probably forgot to do step 4 from the [Visual C++ 2015](http://web.cs.ucla.edu/classes/fall15/cs31/visualcpp.html#step4) writeup, or you started execution by selecting the Start Debugging item from the Debug menu or by double-clicking on the .exe file. What you want to do is select the Start Without Debugging item from the Debug menu; if there is no such menu item, [fix it](http://web.cs.ucla.edu/classes/fall15/cs31/visualcpp.html#step7) as directed in step 7.)
5. Using the program as given, run it with input integers that cause it to produce incorrect, unusual, or nonsensical output. (Notice we're saying to try input *integers*, not input like 124765.23 or fraud.)
6. Starting from the program as given, introduce into the source code at least one error that someone might make that, while not preventing a successful build, causes the program when it runs to produce incorrect results from reasonable input.
7. Again starting from the program as given, introduce at least two distinct types of mistakes that someone might make, each of which would cause the program to fail to compile correctly.

You should create a separate project for each of steps 2, 6, and 7, since you're not allowed to have multiple files in the same project if more than one has a main routine.

In addition to running the programs under Visual C++, run them using clang++ or g++ (perhaps using [Xcode on a Mac](http://web.cs.ucla.edu/classes/fall15/cs31/xcode.html) or [g++ with Linux](http://web.cs.ucla.edu/classes/fall15/cs31/linux.html)).

What you will turn in for this assignment is a compressed file in zip format containing exactly four files:

1. A file named **original.cpp** that contains the program as given.
2. A file named **logic\_error.cpp** with the program you produced in step 6.
3. A file named **compile\_error.cpp** with the program you produced in step 7.
4. A file named **report.doc** or **report.docx** (in Microsoft Word format) or **report.txt** (an ordinary text file) that describes the input you provided in step 5 and each of the errors you introduced into the logic\_error.cpp and compile\_error.cpp programs. Briefly discuss any error messages the compiler reported, and incorrect, unusual, or nonsensical results. This report may well end up being much less than a page long.

The zip file itself may be named whatever you like.

Do **not** include anything else in the zip file. (Some Windows users seem not to be aware of [Windows filename extensions](http://web.cs.ucla.edu/classes/fall15/cs31/filenameext.html), so end up putting the wrong files in their zip file.) To create a zip file on a SEASnet machine, you can select the four files you want to turn in, right click, and select "Send To / Compressed (zipped) Folder". Under Mac OS X, copy the files into a new folder, select the folder in Finder, and select File / Compress "*folderName*"; make sure you *copied* the files into the folder instead of creating aliases to the files.

We will be using software tools to help us grade your projects, so there are certain requirements you must meet for the tools to work: **The zip file you turn in for this project must have *exactly* four files in it, with *exactly* the names indicated. If you do not follow these requirements, your score on this project will be *zero*.** "Do you mean that if I do everything right except misspell a file name or include an extra file, I'll get no points whatsoever?" Yes. That seems harsh, but attention to detail is an important skill in this field. A draconian grading policy certainly encourages you to develop this skill.

The only exception to the requirement that the zip file contain exactly four files of the indicated names is that if you create the zip file under Mac OS X, it is acceptable if it contains the additional files that the Mac OS X zip utility sometimes introduces: \_\_MACOSX, .DS\_Store, and names starting with .\_ that contain your file names.

By October 5, there will be links on the class webpage that will enable you to turn in your zip file electronically. Turn in the file by the due time above. Remember that most computing tasks take longer than expected; this applies especially to steps 1, 2, and 3 above. Start this assignment now!

Use this project as an opportunity to learn what happens when you make mistakes. After you've turned in what's required for this project, play around. Introduce a mistake into the program and see what happens. Fix it. Introduce a different mistake and see what happens then. Fix it. Keep doing this so you can see the kinds of problems that might arise when you develop your own programs and what the compilers say for each particular problem (if they even detect them at all). See what happens if you make more than one mistake in the program. Will they all be detected? Will an earlier mistake interfere with the reporting of a later one.

Here is the C++ program:

// Code for Project 1

// Report survey results

#include <iostream>

using namespace std; // pp. 38-39 in Savitch 6/e explains this line

int main()

{

int numberSurveyed;

int numKeep;

int numRefund;

cout << "How many diesel Volkswagen owners were surveyed? ";

cin >> numberSurveyed;

cout << "How many of them say they will take a payment and keep their car? ";

cin >> numKeep;

cout << "How many of them say they will return their car for a refund? ";

cin >> numRefund;

double pctKeep = 100.0 \* numKeep / numberSurveyed;

double pctRefund = 100.0 \* numRefund / numberSurveyed;

cout.setf(ios::fixed); // see pp. 32-33 in Savitch 6/e

cout.precision(1);

cout << endl;

cout << pctKeep << "% say they will keep their car." << endl;

cout << pctRefund << "% say they will demand a refund." << endl;

if (numKeep > numRefund)

cout << "More people will keep their car than demand a refund." << endl;

else

cout << "More people will demand a refund than keep their car." << endl;

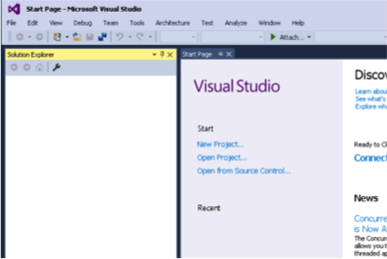
}

### **Creating a Visual C++ Project**

Visual C++ is designed to support developers of large systems, so to use it as we will initially to write simple one-file programs may seem like overkill. However, over time you will come to appreciate some features (the debugger, especially).

Here is a way to set up a Visual C++ project to compile and run a simple one-file program. A number of these steps may have alternate ways to achieve the same end.

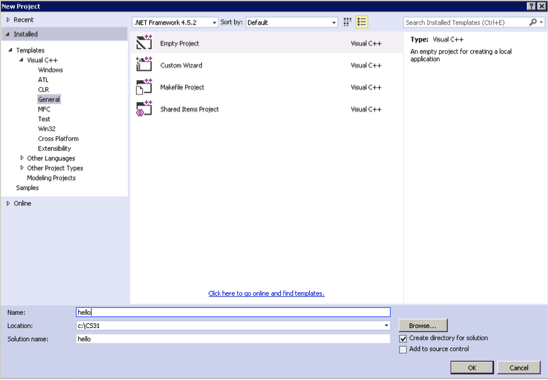
1. Start up the Visual C++ IDE (Integrated Development Environment). If you don't have a shortcut on the Desktop, then from the taskbar you'll probably select something like Start / All Programs / Microsoft Visual Studio 2015. The first time you fire it up after installation, it may take a couple of minutes to get you to the start page. If it says "Start with a familiar environment", for the Development Settings, select Visual C++ from the drop-down list. (If you're not offered Visual C++ as a choice but see othe languages, you forgot to specify a custom installation when you installed Visual Studio Community.)
2. In the left part of the Start Page tab, follow the New Project... link. Alternatively, from the menu, select File / New / Project...



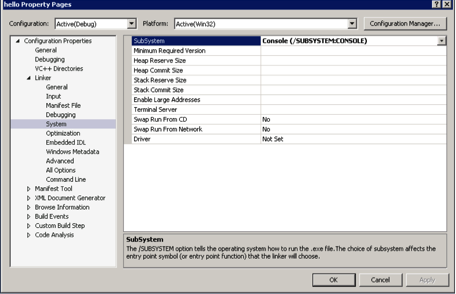
On the SEASnet Windows Server, a dialog box may appear saying that the 'VSPackage' package did not load correctly. It appears that you can just dismiss this dialog box without harm.

1. In the New Project dialog, select General in the left panel and then Empty Project in the middle panel. In the text boxes below, enter for Name a project name, such as "hello". So that you can more easily find your projects later, you might want to change the Location to something like "C:\CS31" on your own machine, or "Z:\CS31" on a SEASnet machine. On a SEASnet machine, even if you don't use a simple location like "Z:\CS31", at least change the initial part of the location from "\\labsamba2.seas.ucla.edu\yourusername\" to "Z:\". In general, to make reading error messages easier in step 8, a short location like "C:\CS31" or "Z:\CS31" is better than a long one like "\\labsamba2.seas.ucla.edu\yourusername\my documents\visual studio 2015\projects", and a short project name like "hello" or "P1" is better than a long one like "example CS31 project".

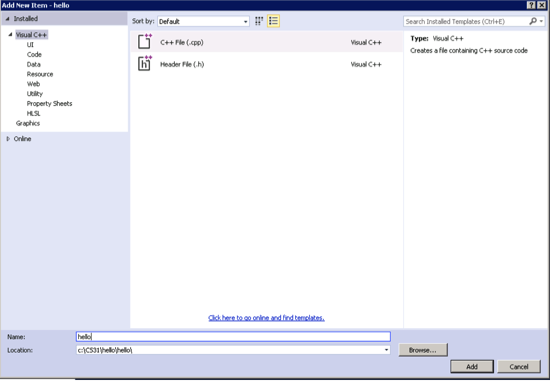
Click OK. (If you're on a SEASnet machine you may then see a dialog saying "The Project location is not fully trusted by the CLR runtime." It's harmless to click OK to dismiss it.)



1. Select Project / Properties. In the Property Pages dialog, in the left panel, select Configuration Properties / Linker / System. In the right panel, select SubSystem, and in the drop down list to its right, Console (/SUBSYSTEM:CONSOLE). Click OK.



1. Select Project / Add New Item. In the Add New Item dialog, select C++ File (.cpp) in the middle panel. Enter a source file name, such as "hello", in the Name text box below. Click Add.



1. Edit the hello.cpp file in the window that appears. Type in a program like

#include <iostream>

using namespace std;

int main()

{

cout << "Hey, this really works!" << endl;

}

1. From the Debug menu, select Start Without Debugging. This will save your source file, compile it, and run the resulting executable if there were no build errors. (If you select Start Debugging, or if you forgot to do step 4, or if you launch your program by finding and double-clicking on the .exe file, your console window screen will disappear as soon as your program finishes executing, which you don't want.)

If you see a dialog box that says "This project is out of date: ... Would you like to build it?", check the "Do not show this dialog again" checkbox before clicking Yes.

1. If your program has any compile-time or link errors, a dialog box will appear telling you there were build errors. Check the "Do not show this dialog again" checkbox before clicking No, you don't want to continue. The Output window below your source file will list the errors. Double-click the first error message to see what it's referring to in your code. Fix the error. You can click on and fix other errors as well, but be aware that sometimes one mistake produces multiple error messages, so your first fix may resolve those problems as well. Go back to step 7.
2. If your program runs, its output will appear in a new console window that pops up. When your program finishes, that console window will display the text
3. Press any key to continue . . .

Press any key to dismiss the console window.