

Learning to Use the Development Environment

Obviously you cannot program unless you understand how to create a file containing your C++ language source code, how to compile and link the source code to produce a program, how to execute (run) the program, test its behavior, fix errors, etc.

For the first programming project, you will be given C++ source code for a working program. Your task is to type in the given source code, including the comments, without making any unspecified modifications, and verify that it does indeed perform as specified. Along the way, you'll be exposed to quite a bit of C++ language that we haven't even begun to cover, so don't get paranoid. Try to understand the given source code (I've included lots of comments to help), but realize that this program uses elements of the C++ language from the first seven chapters of the class notes.

The Problem:

Speed Trap

For this first assignment, you will not be writing a program. Nevertheless, it is useful to provide a detailed specification for the problem that the program given below solves. Read this carefully. As we cover C++ basics, review this specification and the following program and be sure you understand how the program was designed.

We all know that the speed of an object can be calculated by using radar. But how? The general solution is somewhat complex, but there is a simple sub-problem that admits of a simple solution. Assume the object is traveling directly toward the radar unit. In that case, the speed of the object can be calculated from the change in frequency in the radar signal. A radar signal has a frequency, measured here in cycles per second or Hertz. When the radar signal is reflected by the oncoming object, its frequency is changed. If the radar unit can measure the frequency of the reflected signal, the speed of the oncoming object can be calculated using the formula:

$$\Delta f = \frac{v \times f}{334.8}$$

where f is the emitted frequency in Hz, Δf is the change in the frequency (in Hz), and v is the speed of the object in miles per hour.

Write a program to read an input file, described below, and use the formula given above to calculate the speed of the object. The program should then produce a report file, also described below.

Sample input and corresponding output:

Here is a sample input file for the program. The first line contains the name of the observer, the second contains the date of the observation. The third and fourth lines specify the emission frequency of the radar unit and the return frequency, respectively. Each of the data values is preceded by a text label and a single tab character.

| | |
|-----------|--------------|
| Observer: | Barney Fife |
| Date: | May 22, 2006 |
| Emit Hz: | 2450 |
| Ret Hz: | 2970 |

The specification of the input file format is relatively tight. This will be typical of input data for the projects in CS 1044. The program must be designed and implemented to read the data file correctly. A clear specification of the input format is necessary in order to make the program reasonably simple.

Here is a sample output file for the program. The output file reports the results computed by the program. The output format must also be specified rigidly because of the automated grading system (Curator) we will use in CS 1044. In this case, the output format would be specified as follows.


```
// Attach an input stream to the data file:
ifstream Data("Observation.txt");

Data.ignore(INT_MAX, '\\t'); // discard label and trailing tab
getline(Data, reportingOfficer); // read name of observer
Data.ignore(INT_MAX, '\\t'); // discard label and trailing tab
getline(Data, reportDate); // read date of observation
Data.ignore(INT_MAX, '\\t'); // discard label and trailing tab
Data >> emitFrequency; // read radar emission frequency
Data.ignore(INT_MAX, '\\t'); // discard label and trailing tab
Data >> returnFrequency; // read radar return frequency

// There's no more data to be read, so close input file:
Data.close();

// Calculate the change in the frequency:
freqDelta = returnFrequency - emitFrequency;

// Calculate estimated speed of radar target:
Speed = freqDelta * ConversionFactor / emitFrequency;

// Attach an output stream to the report file:
ofstream Log("Report.txt");

// Prepare the stream for formatted decimal output:
Log << fixed << showpoint;

// Write the report data:
Log << "Observer: " << setw(20) << reportingOfficer << endl;
Log << "Date: " << setw(20) << reportDate << endl;
Log << "Est. speed:" << setw(20) << setprecision(1) << Speed << endl;

// That's the end of the output, so close the output file:
Log.close();

// Return 0 for a successful execution:
return 0;
}
```

How to create and test the program:

First of all, you must either work in the University Labs in Torgersen, or have obtained Microsoft Visual C++ 2008 Express and installed it on your computer. Read through Visual C++ introduction on the course website for a short tutorial on using the Visual C++ development environment (IDE). The IDE is a standard Windows application and its interface is quite similar to other products you are probably already familiar with, such as Microsoft Word.

Once you're somewhat familiar with the IDE, you must type in the program listed in this assignment. Before doing that, you should decide where you are going to save your C++ language source and data files. It's best to adopt some sensible scheme for this; here's one suggestion. Using Windows Explorer, create a folder (somewhere, that's up to you) named CS1044Projects. Inside that, create another folder named HW1. Use this folder to save all the files you create for this project. When you advance to the later projects, just add a new folder for each one.

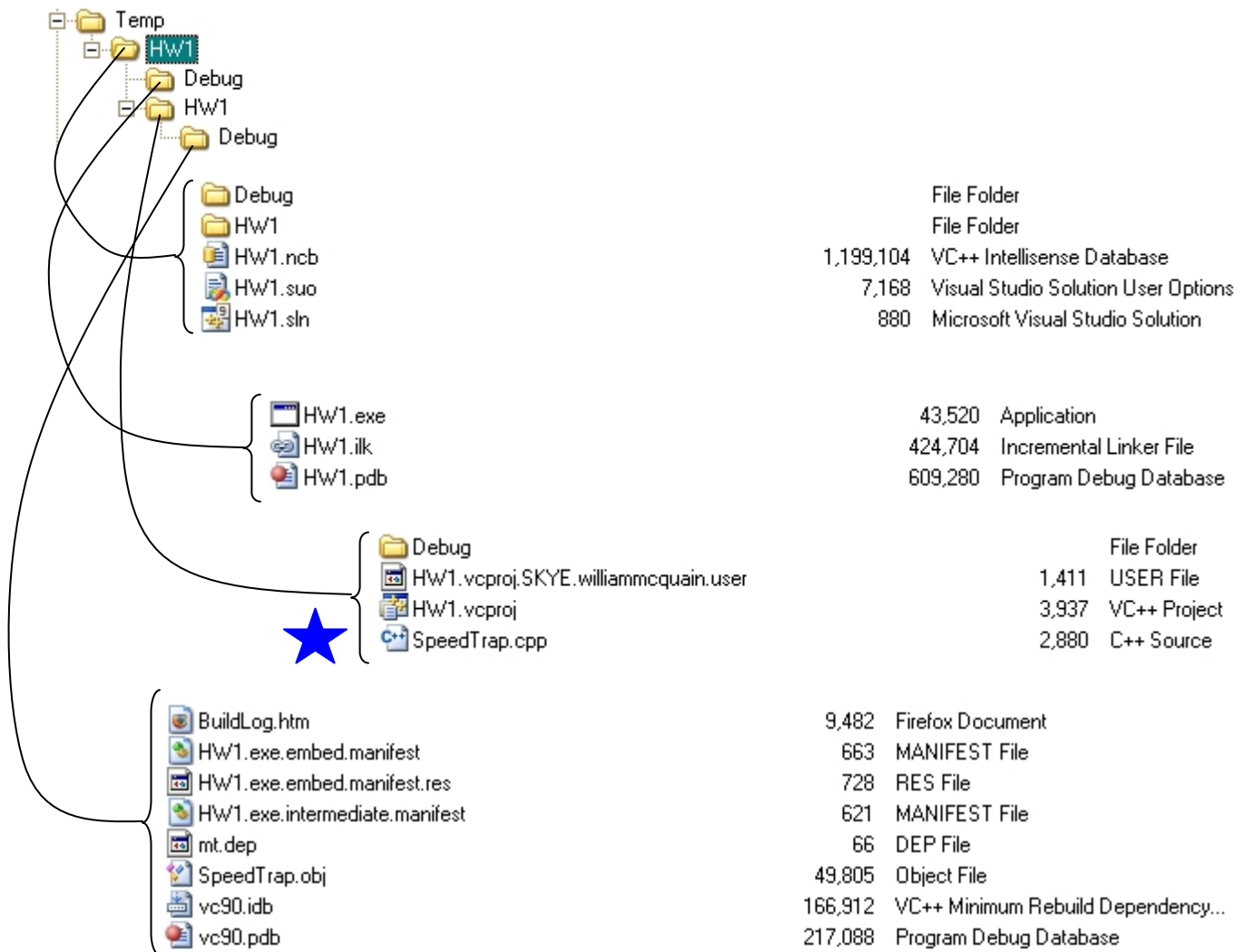
Start the Visual C++ IDE and create a new project named HW1 as described in the Visual C++ introduction. In the Solution Explorer pane, right-click on the Source folder and select "Add/New item...". In the following dialog box, select "Code" and "C++ file". Give the file a descriptive name, say SpeedTrap.cpp.

Now, use the Visual C++ IDE and type in the given program. (Again, the Visual C++ introduction shows how to do this.) For the most part, you must type in the program exactly as given. If you make changes you'll probably break the program so it doesn't compile, or produces incorrect results. Save your C++ language source file often as you type it in.

Once you've typed in the program, you should follow the example in the Visual C++ introduction and try compiling it. The source code given in this specification is correct, so if you typed it in exactly then it will compile without any errors. If you do get error messages, follow the example in the Visual C++ introduction to determine which lines the compiler is complaining about. Compare those lines (and maybe the preceding lines) to the given source code. Correct the differences.

Once your program compiles without errors, you must test it to be sure it produces the correct results. Note: the fact that a program compiles doesn't mean it's a logically correct program. Of course, the given source code is also logically correct, so if you typed it in exactly then it will produce correct results. But you can't know that unless you test it. Actually, testing it won't prove that it's really correct either...

At this point, you'll have a directory structure something like this:



Fortunately, there's no reason you need to worry about what most of those files are for. However, you should make sure that you notice which directory contains the source (cpp) file that you created.

This program, like all of the projects in CS 1044, requires an input file from which it will read data to be processed. You must create that input data file, and give it the right name, and put it in the right place. The mandatory name of the input file is given in the program source code. You must save the input file in the same folder as your source file.

How do you create the input file? There are a couple of possibilities. You could type in the sample input data given above and then save it in the right folder with the correct name. If you do this, do not use a word processor like Microsoft Word to type in the file! Use a text editor. The simplest choice is to just use the Visual C++ IDE. If you use Notepad, be careful about the file name when you save it because Notepad has a nasty habit of adding ".txt" to the name you select so you might wind up with a name like "Observation.txt.txt" and the program won't recognize that.

You can also download sample input data files from the course website; just right-click on the link for the file you want and select "Save target" or "Save link". It's not a good idea to copy the text from your web browser (Internet Explorer, Netscape, etc.) and paste it into an editor to save it because that will sometimes add invisible garbage characters to the file.

Be sure to pick the right folder and type in the right name for the file when you save it. The *right folder* is the folder that contains the source file you just created. Note the STAR in the directory illustration above.

Once you've got an input data file you can test your program. Just run the program (see the Visual C++ introduction). A console window will pop up, usually with the message "Press any key to continue". Press (almost) any key to get rid of the console window. But where are your results? The program will have created an output file, in the same folder as your source file, with the name specified (in the given source code) for the output file. To view your results you must open the output file. Just go to File..Open in the Visual C++ IDE; the dialog box will probably only show the files Visual C++ cares about, but you can go to the "Files of type" list and select "All files" at the bottom. Now the output file your program created should appear in the file list. Select it and check its contents against the posted correct output. If there's a difference, you didn't type in the given source code correctly. Find the differences and correct them, then run your program again and check the results again.

Submitting your program:

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* and other pertinent information, such as the link to the proper submit page, can be found at:

<http://www.cs.vt.edu/curator/>

Pledge:

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
// On my honor:  
//  
// - I have not discussed the C++ language code in my program with  
// anyone other than my instructor or the teaching assistants  
// assigned to this course.  
//  
// - I have not used C++ language code obtained from another student,  
// or any other unauthorized source, either modified or unmodified.  
//  
// - If any C++ language code or documentation used in my program  
// was obtained from another source, such as a text book or course  
// notes, that has been clearly noted with a proper citation in  
// the comments of my program.  
//  
// - I have not designed this program in such a way as to defeat or  
// interfere with the normal operation of the Curator System.  
//  
// <Student Name>
```

Failure to include this pledge in a submission is a violation of the Honor Code.