

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 program source code:

```
// Project 1 for CS 1044 Summer I 2001
//
// Student:      <put your name here>
//
// Programmer:   William D McQuain
// OS:          Windows NT Workstation 4.0
// System:      Pentium II 400, 128 MB Memory
// Compiler:    Visual C++ 6.0, Service Pack 4
// Last modified: May 14, 2001
//
// Purpose
// This program reads a list of player names and scores from an
// input file, counts how many players are in the list, computes
// the sum and average of the scores, and determines the winning
// (lowest) score and the name of the matching player.
//
// The program then writes a summary of its findings to an output
// file.
//
#include <iostream> // for cout
#include <fstream>  // for file streams
#include <iomanip>  // for formatting manipulators
#include <string>   // for string variables
#include <climits> // for INT_MAX
using namespace std; // to put all of the above things in scope

int main() {

    const string dataFileName = "Data.txt"; // Name of the input file
    const string reportFileName = "Report.txt"; // Name of the output file

    ifstream In( dataFileName.c_str() ); // Attach an input stream to the
                                        // input file.

    // If the input file does not exist, this will detect that.
    // We handle that by printing an error message and stopping the program.
    if ( In.fail() ) {
        cout << "Data file not found: " // Write an error message...
              << dataFileName // including the file name.
              << endl // Bang "return"
              << "Exiting now..." << endl; // Finish the message.
        return 1; // Traditional to return 1 on failure.
    }
}
```

```
string NameOfLeader = ""; // Name of player with lowest score so far
int ScoreOfLeader = INT_MAX; // Lowest score seen so far

string nextName = ""; // Name just read from input file
int nextScore = 0; // Score just read from input file
int NumberOfPlayers = 0; // Counter for number of players
int SumOfScores = 0; // Running total of all scores

// Skip over the first line of the input file:
In.ignore(INT_MAX, '\n');
// Try to read a name and score from the input file:
In >> nextName >> nextScore;

while ( In ) { // Continue as long as the last read succeeded

    if ( nextScore < ScoreOfLeader ) { // See if this is a new low score.
        // If so...
        NameOfLeader = nextName; // remember this name and
        ScoreOfLeader = nextScore; // score
    }

    SumOfScores = SumOfScores + nextScore; // Add in this score
    NumberOfPlayers = NumberOfPlayers + 1; // Count this player

    In.ignore(INT_MAX, '\n'); // Advance to beginning of next input line
    In >> nextName >> nextScore; // Try to read another name and score
}

In.close(); // Tell the OS you're done with the input file.

// Now calculate the average score.
double AverageScore;
if ( NumberOfPlayers > 0 ) { // Check to be sure it's safe to divide.

    // We convert the integers to decimal values before dividing:
    AverageScore = double(SumOfScores) / double(NumberOfPlayers);
}
else {
    AverageScore = 0.0;
}

ofstream Out( reportFileName.c_str() ); // Attach an input stream to the
// input file.

Out << fixed << showpoint;

// Write an identification header to the output file:
Out << "Programmer: "
    << "<put your name here>" // Change this to YOUR name.
    << endl; // That's an "ell" not a "one".

Out << "CS 1044 Summer I 2001 Project 1"
    << endl
    << endl;
```

```

// Write a summary of the results computed above:
if ( NumberOfPlayers > 0 ) {
    Out << "Winner:          " << setw(15) << NameOfLeader << endl;
    Out << "Winning score:     " << setw(6) << ScoreOfLeader << endl;
    Out << "Number of players:  " << setw(6) << NumberOfPlayers << endl;
    Out << "Average score:       " << setw(6) << setprecision(1)
                                << AverageScore << endl;
}
else {
    Out << "There were no players." << endl;
}

Out.close(); // Tell the OS you're done with the output file.
return 0;    // Traditional to return 0 on success.
}

```

Sample input and corresponding output:

Here is a sample input file for the program. The first line contains column labels and is ignored by the program. Each remaining line contains a name and a score, separated by one or more spaces. The name will be a string of characters, and will not contain any embedded spaces or tabs. The score will be a positive integer and will be last data item on the line.

| Player | Score |
|---------|-------|
| Watson | 68 |
| Trevino | 67 |
| Niklaus | 68 |
| Kite | 72 |
| Miller | 70 |

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. It begins by identifying the programmer (you) and the specific project. We will always require that information at the beginning of your output file. The remainder of 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.

There will be four lines of output, each specifying a labeled value computed by the program. The output must include the name of the player with the low score, that player's score, the total number of players reported, and the average of all the scores. The average score must be printed with precision 1; i.e., showing one digit after the decimal point. The labels must be precisely as shown in the sample output. The output should be aligned for easy readability.

```

Programmer: <put your name here>
CS 1044 Summer I 2001 Project 1

Winner:          Trevino
Winning score:     67
Number of players:  5
Average score:     69.0

```

Additional samples of input and correct output are available on the course website.

How to create and test the program:

First of all, you must either work in the CS Department Lab in McB 118, or have obtained Microsoft Visual C++ version 6.0 and installed it on your computer. Read through Appendix 1 of the CS 1044 course notes for a quick introduction to using the MS 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 CS 1044 Projects. Inside that, create another folder named Project1. 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.

Now, start the Visual C++ IDE and type in the given program. (Again, Appendix 1 in the course notes 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. It doesn't matter what you call the source file, or the project workspace, but something descriptive like P1 or Project1 will make it easier to keep things straight.

Once you've typed in the program, you should follow the example in Appendix 1 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 Appendix 1 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...

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 "Data.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". Be sure to pick the right folder and type in the right name for the file when you save it. 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.

Once you've got an input data file you can test your program. Just run the program (see Appendix 1). A black 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* can be found at: <http://ei.cs.vt.edu/~eags/Curator.html>

The submission client can be found at: <http://spasm.cs.vt.edu:8080/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.