

Procedural Decomposition:**Simple Stock Portfolio Tracker**

For this project you will modify and extend your implementation for Project 2 to accommodate a design based upon a good procedural decomposition of the assigned problem. The input file will have precisely the same format as for the previous project. The output file will be slightly modified to reflect some additional requirements.

Sample input data:

Here is a sample input file, named `StockData.txt`, for the program. The first line specifies the name of the customer, and the second specifies the customer's account number. The third line specifies the report date. Each of the remaining lines specify the name of a particular stock, the number of shares the customer holds, and the initial and final prices for the stock. These values are separated by single tab characters.

```
Customer Name  John Q Public
Account #      432-10023-4311-89
Report Date    08/05/2002

MSFT          650   57.32  58.14  55.03  55.17
IBM           100   53.02  54.27  53.02  54.00
APPL          200   17.45  17.45  13.09  13.09
```

There is no limit on the number of lines of stock data that may be supplied; your program must terminate properly when the input stream fails. There will be no sentinel line. The input values are guaranteed to be syntactically and logically correct. The customer name, account number and date are just string data. For formatting purposes, you may assume that the customer name will be no more than 40 characters long, and the account number and date will be no longer than the ones shown above.

Calculations:

For each of the given stocks, you will still calculate all of the values specified for the previous project. In addition, you will also calculate a measure of the volatility of each stock's performance. For the purposes of this assignment, we will measure the volatility as the total range of the stock's price movement as a percentage of the opening price of the stock. More specifically, the volatility involves the sum of the absolute values of three quantities: the difference between the high and low prices, the difference between the high price and the smaller of the initial and closing prices, and the difference between the low price and the larger of the initial and closing prices. In the event the initial price is zero, we will arbitrarily define the volatility to also be zero; however that situation will not arise in actual data.

The comments on representing monetary values given for Project 1 still apply.

Sample output:

Here is a sample output file, named `Summary.txt`, which corresponds to the input data given above:

```
Programmer:  William D McQuain
CS 1044 Simple Portfolio Tracker

Customer:    John Q Public   432-10023-4311-89
Date:        08/05/2002
Stock        Shares      Open      Close      Change      % Change  Volatility
-----
MSFT         650       57.32     55.17     -1397.50    -0.038    0.15
IBM          100       53.02     54.00       98.00      0.018     0.07
APPL         200       17.45     13.09     -872.00    -0.250    0.75
-----
Summary                                46050.00  43878.50  -2171.50   -0.047

Winners:     1
Losers:      2
```

The format of the output file is identical to that for Project 2, except that there is an additional column showing the volatility of each stock; you must report the volatility to two decimal places.

If you have read the description of how the Curator scores your program in the *Student Guide*, you know that is important that you use the same spelling and capitalization for all the labels shown above. The horizontal spacing does not effect scoring unless you combine things that should be separate or separate things that should be combined. You are free to experiment with the horizontal layout but you should try to align the columns neatly.

Procedural decomposition:

As always, you should practice incremental development. In this case, you have already done much of the low-level design work in completing the previous project. You should also be able to recycle much of the C++ source code you wrote for the previous project.

The primary new design issue is that you must take advantage of the opportunity to divide the assignment into independent procedures. Your implementation must reflect this by making good use of C++ functions. Specific minimal requirements are given in the Evaluation section that follows; be sure you follow them.

Evaluation:

Everything that you have been told about testing in class applies here. Do not waste submissions to the Curator in testing your program! There is no point in submitting your program until you have verified that it produces correct results on the sample data files that are provided. If you waste all of your submissions because you have not tested your program adequately then you will receive a low score on this assignment. You will not be given extra submissions.

Your submitted program will be assigned a score, out of 100, based upon the runtime testing performed by the Curator System. We will also be evaluating your submission of this program for documentation style and a few good coding practices. This will result in a deduction (ideally zero) that will be applied to your score from the Curator to yield your final score for this project.

Read the *Programming Standards* page on the CS 1044 website for general guidelines. You should comment your code in the same manner as the code given for the first two programming assignments. In particular:

- You should have a header comment identifying yourself, and describing what the program does.
- Every constant and variable you declare should have a comment explaining its logical significance in the program.
- Every major block of code should have a comment describing its purpose.
- Adopt a consistent indentation style and stick to it.

Your implementation must also meet the following requirements:

- Choose descriptive identifiers when you declare a variable or constant. Avoid choosing identifiers that are entirely lower-case.
- Use named constants instead of literal constants when the constant has logical significance.
- Use C++ streams for input and output, not C-style constructs.
- Use C++ string variables to hold character data, not C-style character pointers or arrays.
- Use C++ functions intelligently and appropriately. In particular, you must implement and use at least 4 functions in addition to `main()`. At least one of these functions must be `void`; at least one must return a value via a `return` statement, and at least one must use a reference parameter to return a value. It is acceptable that all function calls are from `main()` to other functions.
- Use the appropriate protocol when passing each parameter to a function. In particular, do not pass any parameter by reference unless it is being used to communicate a changed value from the called function to its caller.

Important: since the primary functionality of this program was already present in the previous project, you must meet the functional decomposition requirements given above in order to receive full credit. A program that uses only one function, `main()`, will be assigned a final score of zero. Programs that use fewer than the specified number of functions will receive substantial penalties.

Understand that the list of requirements here is not a complete repetition of the *Programming Standards* page on the course website. It is possible that requirements listed there will be applied, even if they are not listed here.

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 submission link 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.