

A Basic C++ Program:**Investment Tracking**

For this project, you will implement a complete, simple C++ program. The code you were given and completed for the first two projects will contain many clues as to how this should be done. The notes and lectures will contain the rest.

You will be given data about three investment accounts, including the account name, current balance, rate of return, and a brief description of the account. You will compute a summary that specifies the total of the initial balances, the total increase in value according to the given rates of return, the total of the final balances of the accounts, and the overall rate of return.

Part of the assignment is to determine how to perform these calculations. You should certainly already understand how to compute a percentage return on an initial amount. The rest of the calculations are pretty obvious. In any case, it will be a violation of the Honor Code to post any formulas relating to these calculations on the course forum or to provide those formulas to any other student. You may, of course, get help from your instructor or the course TAs.

There is one important data representation issue in this project. As you will soon learn, computer hardware almost never stores decimal values exactly, and so any calculations done with decimal values are likely to be slightly wrong. To avoid that, your program is required to store all monetary amounts internally using integer variables.

You may either store dollars and cents separately, or store a total amount in cents, but the latter is much simpler. Failure to do this will result in a deduction when the TAs evaluate your implementation, and also may result in a deduction when the Curator scores your submission.

Note that the requirement doesn't apply to anything but monetary amounts. In particular, you are not required to store percentage rates using integers; in fact, that would make your implementation much more difficult.

Sample input and corresponding output:

Here is a sample input file for the program, named "AccountInfo.txt". The first two lines form the top of a table of data. Each of the remaining lines specifies the name, current balance, rate of return, and a description of an investment account.

Account	Balance	Rate	Description
WACHOVIACD	3152.19	0.028	24 month CD
Dreyfus	10522.72	0.063	money market account
Checking	970.60	0.032	savings account

The input file is guaranteed to conform to this description. The balances will always be nonnegative decimal numbers, as will the rates of return. The account names will never contain spaces or tabs (which does have something to do with how you will read them). The descriptions may contain any characters, including whitespace, and there is no given bound on how long the descriptions may be. The data values are separated by spaces, not by tabs.

Here is a sample output file for the program, which must be named "AccountReport.txt". It begins with two lines identifying the programmer (you) and the specific project, followed by a blank line. 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.

There are two lines forming the header for a table of results, as shown. The next line reports the computed results: the initial total balance, the total gain, and the total final balance. That is followed by a blank line, and then a line that reports the overall rate of return, which must be reported with precision 3 as shown. If you have read the *Student Guide to the Curator*, you already know that all the fixed text must be precisely as shown in the sample output. The output should be aligned for easy readability.

```
Programmer: <your name here>
CS 1044: Account Information

Initial Balance          Gain          Final Balance
-----
          14645.51          782.24          15427.75

Overall return:      0.053
```

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

Suggested implementation plan:

You should design your solution piece by piece. Begin by identifying the major tasks that have to be done, and then add detail for each of those tasks. Your implementation should be developed in the same manner. Here's a suggested order of implementation.

First, implement the input code to read the table header, and the data for the first account. Write output code to print them out to verify your input logic is correct. This will require declaring the appropriate stream variables and setting up the file streams, and declaring and using variables to store the input values. Test this code and make sure it produces correct results. Once you know this part works, extend it to read the data for the second and third accounts, and verify that it works. You should never have to worry about your input code again.

Second, implement the code to calculate the gain for each account, and to echo the gain to output, to be sure you're calculating the gains correctly. Of course, you'll have to check these by hand since the individual gains aren't included in the final output files.

Then add the code to calculate the necessary totals and to print them. Test this code. When this is correct, you can safely delete the unnecessary output code. Add the calculation of the overall rate of return, and output it.

By implementing the program feature by feature you let yourself concentrate on solving one small part of the problem at a time. You also should only have to test one part of it at a time as well.

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 may also evaluate 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 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.
- Note: you are explicitly forbidden to write any user-defined functions for this program. This will make the program somewhat repetitive, and physically longer than the alternative. To some extent, that's the reason for this restriction.

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* can be found at: <http://www.cs.vt.edu/curator/>

The submission client can be found at: <http://eags.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.