

1 Introduction

This Integration Stage lab is an alteration of the last one, where you will first load the file data into a system of vectors and do your calculations by accessing the data in the vectors.

Remember to follow the coding and documentation standards for the class listed on the MyClasses pages.

When you are ready to submit your work create a folder called Lab08 in that folder have separate folders for each project, one folder per project. Put all the code files needed for that project in its respective folder. Do not include the files that the IDE creates, I just want the code files. Zip the entire Lab08 folder up into a single zip file and submit it.

2 Exercise

This program will read in stock data from a text file and load it into three parallel vectors. One is an vector of strings that will hold the day's date, one is a vector of doubles that will hold the opening cost for each day, and the third is another vector of doubles that will hold the closing cost for each day. As with parallel arrays, the i^{th} row of each represents the i^{th} day's date, opening and closing costs respectively.

As with the last exercise this program will print out the data along with some descriptive statistics about the data. In addition, you will be doing the calculations inside functions. The main will take in the filename from the user, create the vector storage arrays and call a function to load the data into the vectors. It will then call the needed functions, and display the results.

As before you will be analyzing stock data taken from a free stock data website run by yahoo. The files you will be using are included with the lab 7 files and are IBM.txt, INTC.txt, and MSFT.txt. Each of these contains the opening and closing stock prices for that stock along with the date of that day. The IBM.txt file contains the stock prices for IBM from January 3, 1962 to February 8, 2021. The INTC.txt file contains the stock prices for Intel from March 18, 1980 to February 8, 2021. The MSFT.txt file contains the stock prices for Microsoft from March 14, 1986 to February 8, 2021. Obviously, I downloaded these on February 8, 2021.

You can assume that the files all have the same format. The files are tab delimited, which means that there is a tab character between entries on a line and each line is a separate day. There are no other characters between the entries and there is never more than one day on a line. The first line is a header, if you were going to load this into a spreadsheet. Each line after that is the data for a day on the market. It starts with a date, then the opening price, then the closing price. You are not to alter this file in any way. The beginning of the file will look like the following.

| Date | Open | Close |
|------------|----------|----------|
| 1980-03-18 | 0.325521 | 0.322917 |
| 1980-03-19 | 0.330729 | 0.330729 |

```
1980-03-20  0.330729  0.329427
1980-03-21  0.322917  0.317708
1980-03-24  0.316406  0.311198
```

The file ends with the last recorded day, as in the example below.

```
2021-02-03  57.889999  57.68
2021-02-04  57.610001  58.790001
2021-02-05  59  58.18
2021-02-08  58.380001  59.16
```

You will write a program that will ask the user to input a filename. The program will then,

1. Display each day's date, opening cost, closing cost, a + or - if the day had an increase or decrease respectively and then the amount of increase or decrease. If the day is flat, opening and closing price the same the day is marked with — instead of an amount change.
2. At the end the program will display a set of descriptive statistics for the stock.
 - Number of days in the data set.
 - Average closing price.
 - Number of days the stock went up.
 - Number of days the stock went down.
 - Number of days the stock was flat. (no increase or decrease)
 - The longest climb, displaying the number of days in the climb and the date range of the climb. A climb is when consecutive days of the closing cost of the stock is either increasing in value or flat. In other words, the longest stretch of days that the stock closing cost did not decrease in value.
 - The longest fall, displaying the number of days in the fall and the date range of the fall. A fall is when consecutive days of the closing cost of the stock is either decreasing in value or flat. In other words, the longest stretch of days that the stock closing cost did not increase in value.
 - The largest increase in value in a day, both the amount and the date it happened.
 - The largest decrease in value in a day, both the amount and the date it happened.

A run of the program is below. The three vertical dots represent several thousand lines that were removed so that the output did not go on for pages, your program will print out every day.

```
Input the filename: IBM.txt
  Date      Opening    Closing      Change
1962-01-03   7.626667   7.693333   +   0.066666
1962-01-04   7.693333   7.616667   -   0.076666
```

| | | | | |
|------------|------------|------------|-----|----------|
| 1962-01-05 | 7.606667 | 7.466667 | - | 0.140000 |
| 1962-01-08 | 7.460000 | 7.326667 | - | 0.133333 |
| 1962-01-09 | 7.360000 | 7.413333 | + | 0.053333 |
| 1962-01-10 | 7.426667 | 7.426667 | --- | |
| 1962-01-11 | 7.446667 | 7.506667 | + | 0.060000 |
| 1962-01-12 | 7.520000 | 7.520000 | --- | |
| 1962-01-15 | 7.546667 | 7.553333 | + | 0.006666 |
| 1962-01-16 | 7.546667 | 7.473333 | - | 0.073334 |
| . | | | | |
| . | | | | |
| . | | | | |
| 2021-01-29 | 120.220001 | 119.110001 | - | 1.110000 |
| 2021-02-01 | 119.900002 | 120.540001 | + | 0.639999 |
| 2021-02-02 | 119.360001 | 119.440002 | + | 0.080001 |
| 2021-02-03 | 119.040001 | 119.120003 | + | 0.080002 |
| 2021-02-04 | 119.910004 | 121.019997 | + | 1.109993 |
| 2021-02-05 | 121.000000 | 121.790001 | + | 0.790001 |
| 2021-02-08 | 122.620003 | 123.610001 | + | 0.989998 |

```

Stock Statistics for file IBM.txt
=====
Number of days in data set: 14877
Average closing stock price: $61.041122
Number of days stock went up: 7161
Number of days stock went down: 7199
Number of days stock stayed flat: 517
Longest climb: 13 Days from 2004-10-28 to 2004-11-15
Longest fall: 15 Days from 2005-03-31 to 2005-04-20
Largest Day Increase: 10.875000 on 2001-01-03
Largest Day Decrease: 11.125000 on 2000-10-12

```

In your program the main will ask the user for the filename to be processed. It will then call the loading function. The loading function will check if the file exists and if not exit the program with a 1 error code. If the file exists it will open the file and load all of the data into the vectors. It will then call the following functions and display the statistics as above. The main is to display the statistics, not the functions. The only function that is to display anything to the console is the `printStockDataChart` function. All functions must be below the main in this exercise and have just their prototypes above the main.

LoadData Takes in the filename and the vectors to be populated. It will open the file and load the data into the vectors.

printStockDataChart Takes in the vectors and prints out a chart of dates, opening, closing, and change as in the example above. Make sure the columns line up nicely with the decimal points aligned and 6 decimal places for each of the floating point values.

CalculateAverageClosingValue Takes in the vectors and returns the average closing value of the stock.

CalculateUpDayCount Takes in the vectors and returns the number of days the stock went up in value.

CalculateDownDayCount Takes in the vectors and returns the number of days the stock went down in value.

CalculateFlatDayCount Takes in the vectors and returns the number of days the stock did not change in value.

getClimbData Takes in the vectors, it will also have three output reference parameters, one integer for the number of days of the longest climb, and two strings that will hold the start and end dates respectively of the climb. A climb is when consecutive days of the closing cost of the stock is either increasing in value or flat. In other words, the longest stretch of days that the stock closing cost did not decrease in value. This function will not return any information through a return type.

getFallData Takes in the vectors, it will also have three output reference parameters, one integer for the number of days of the longest fall, and two strings that will hold the start and end dates respectively of the fall. A fall is when consecutive days of the closing cost of the stock is either decreasing in value or flat. In other words, the longest stretch of days that the stock closing cost did not increase in value. This function will not return any information through a return type.

getMaxDayIncrease Takes in the vectors, it will also have two output reference parameters, a double for the amount of increase on the maximum increase day and a string holding the date it happened. This function will not return any information through a return type.

getMaxDayDecrease Takes in the vectors, it will also have two output reference parameters, a double for the amount of decrease on the maximum decrease day and a string holding the date it happened. This function will not return any information through a return type.

An alternative approach that still uses parallel vectors is to combine the two numeric vectors into a vector version of a two-dimensional array. As a two-dimensional array is an array of arrays, a vector can store vectors. So if we crate a vector is vectors we can do the storage of the numeric data in one structure instead of two. If you would like to take this approach you may.