

# 1 Introduction

This is the first of what I call Integration Stage labs. An Integration Stage lab is where we take several concepts that have been recently done and combine them into a more substantial program. Projects will be doing the same thing but are more involved than these.

**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 Lab06 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 Lab06 folder up into a single zip file and submit it.

# 2 Exercise

In this lab you will be writing only one program, which is an extension of a program you wrote in a previous lab. The program will read in stock data from a text file and print out the data along with some descriptive statistics about the data. In addition, you will be doing the calculations inside functions. All the main will do is take the input of the filename, 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 6 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 a stock is either increasing in value or flat. In other words, the longest stretch of days that the stock 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 a stock is either decreasing in value or flat. In other words, the longest stretch of days that the stock 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 hundred 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 check if the file exists and if not exit the program with a 1 error code. If the file exists it will 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.

**getDayCount** Takes in a string parameter of the filename and returns the total number of days in the data file.

**printStockDataChart** Takes in a string parameter of the filename 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 a string parameter of the filename and returns the average closing value of the stock.

**CalculateUpDayCount** Takes in a string parameter of the filename and returns the number of days the stock went up in value.

**CalculateDownDayCount** Takes in a string parameter of the filename and returns the number of days the stock went down in value.

**CalculateFlatDayCount** Takes in a string parameter of the filename and returns the number of days the stock did not change in value.

**getClimbData** Takes in a string parameter of the filename, 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 a stock is either increasing in value or flat. In other words, the longest stretch of days that the stock did not decrease in value. This function will not return any information through a return type.

**getFallData** Takes in a string parameter of the filename, 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 a stock is either decreasing in value or flat. In other words, the longest stretch of days that the stock did not increase in value. This function will not return any information through a return type.

**getMaxDayIncrease** Takes in a string parameter of the filename, 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 a string parameter of the filename, 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.

Note that each of these functions takes in the filename of the file the user selected. It is also fairly obvious that each of these functions will open the file, read through it, process the information needed for the purpose of the function, and finally close the file. It would be more efficient to read the data into the program once and store it into some structure that we can read multiple times without needing to reread from the file, for example using parallel arrays or vectors or a more sophisticated approach of a single array or vector holding a struct or object that stores the data for each day. We will investigate an approach like this in the future but for this exercise we will not be using an internal storage structure to store the data, we will reread the file data each time.

One (optional) alternative approach that you are allowed to do is the following. In the descriptions above each of the functions is taking the filename of the file, opening, processing, and finally closing the file. There is really no reason to open and close the file so often. Instead, you can have the main program open the file before it calls these functions, have each function bring in an ifstream parameter in place of the filename, this will need to be passed by reference. Each function will reset the file reading pointer back to the beginning of the file and then read it to process the data. Finally, the main will close the file at the end of the program.