

# 1 Introduction

These exercises are all dealing with text file processing. Each exercise should be its own separate project.

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

## 1.1 Submitting Your Work

When you are ready to submit your work create a folder called Lab04 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 Lab04 folder up into a single zip file and submit it.

# 2 Exercises

These exercises deal with text file processing. One nice thing about text files is that it is a little easier to debug your logic errors with then as opposed to binary files. As we have discussed in class, test your programs as you code. With files I first write the portion that reads the files into the program and print it out to the console before I do any more processing. This lets me know if the data is being read in correctly and if the end of the file is being processed the way you need. After I know my input data is sound, I do the math.

1. This program does one of the many functions that most word processors, like Word and LibreOffice Writer, do, which is to count the number of lines, words, and characters in a document. Yours will take an input filename from the user, go through the file and count the number of lines, words, and characters in the file and print out those counts to the console screen.

You may find it convenient to read through the file several times, possibly using different methods to extract the data. One way to reset the file pointer back to the beginning is to close the file and then open it back up. Another way is to use seekg as we did in the class examples. The seekg function works on ifstream as well as the more general fstream objects. One thing to note is that if the file pointer is on the EOF position the seekg will not respond. This is because the fstream and ifstream classes set a particular bit of information for hitting the end of the file (it is actually a bit). Similar bits are set for errors, etc. To use seekg you need to clear these out with the clear function. So if your file variable name is file, the command file.clear() will clear these and then you can use seekg.

The Lab04Files.zip file contains text files book.txt, book2.txt, and Cypher.txt for you to use to test your program. The Cypher.txt is a short single paragraph on the Caesar cipher. The book.txt file is a translation of Homer's Iliad and book2.txt is a translation of Homer's Odyssey. Don't worry, the copyright ran out on those several millennia ago.

2. This program is to take a dictionary file and find all the words that have THE as part of the word. For example, absinthe, mouthed, nither, etc. The program will take a filename from the user for a dictionary file. Dictionary files contain one word per line. The program will then count all the words that contain THE as a substring. The count should be case-insensitive, that is if the word has The, tHe, thE, THe, etc. in it then it should be counted.

You will probably want to use the find command in some way. In many programming languages a failed search for a substring results in the return of a  $-1$  position. In C++, due to possibly different sizes of integer data types comparing an output of find to  $-1$  can lead to unpredictable results. There is a cross platform alternative. In C++ there is a constant in the string library, `string::npos`, which is returned by find if the substring is not there. This constant can be used for comparison to see if the search failed or if the substring is present in the string.

The Lab04Files.zip file contains a text file words.txt for you to use to test your program. This file is a dictionary file used for spell checkers on Linux (and a few other systems). It is the file used by the Hunspell package that is incorporated into LibreOffice, OpenOffice.org, Mozilla Firefox & Thunderbird, Google Chrome, and it is also used by proprietary software packages, like macOS, InDesign, memoQ, Opera and SDL Trados.

3. In this program 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 4 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 will write a program that will ask the user to input a filename, read in the data and display a count of the number of up days. An up day is when the stock price went up that day, so when the closing price is greater than the opening price.

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. 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 are not to alter this file in any way. So you will need to skip over the first line and the dates, and extract the numerical data to complete the calculations and display the results.