Corner Grocer Project Documentation

Author: Joshua Dunphy

Date: 6/18/2024

This was a fun project! I created this for my company, Chada Tech, and I was assigned to do it for our client Corner Grocer. It is designed with C++ and implements three different tracking options from their data. The three different tracking options are:

Option 1: Search for the frequency of times an item is listed in the data. This is done with my function getItemFrequency(). The second option, lists all the items and the corresponding number of times they are bought. This is done using my function listItems(). This function uses a linked list to store the items and update their values in nodes accordingly, to then print them all with their corresponding values. Lastly, for the third option I was required to make a bar chart for the data to represent a visual on the data so we can see how many items are sold with a visual element. This was done using my listHistogram() function, where I used a map<string, int>. The map used the string value to store the item, as well the corresponding number of times it appeared with int, this is a key-value pair. Later, the asterisks were printed per key via the value that it corresponded to.

All of this was possible by using File I/O, where I used file input to process the data received to display the file. I also used file output, to create a backup, this is done first with the Backup() function. My backup function also checks if the file has already been backed up, to avoid unnecessary concurrent file creation.

The logic of the file is structured accordingly. It is separated with three files:

1. Header file: GroceryTracker.h

This file is the blueprint for the program, and contains all the declarations of functions and variables used in the program.

1. Definitions file: Functions.cpp

This file is the definition holder. It defines all the functions declared in the header file, which will be used by the main function.

1. Main driver: Main.cpp

This is the main file of the program, the compiler reads this file and executes the program accordingly, line-by-line. This is responsible for executing the program, and calling all the functions.

I also ensured good programming practices. Such as naming conventions, proper inline comments, and as proper handling memory leaks and edge cases throughout the program. The best part? This program includes consistent looping. So, it will always prompt a user for input and then re-prompt even if incorrect input is used. You can easily navigate the menu this way, as well as continue to go through all selections necessary to get the most out of the data.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

These pictures are screenshots of my menu. You can clearly see that all options are working accordingly, and that it is re-looping and displaying the correct data, when prompted!