GroceryTracker.cpp Documentation

Project 3

Timothy Celestino 02/20/2025

For this project the instructions required the code to have the following 4 attributes:

1) Create a data file, with the naming convention frequency.dat, for backing up your accumulated data

2) The code must include at least one class with public and private sections.

3) we recommend using Maps

4) Optional challenge of making sure we have menu option 4 available so users can exit the program.

I will begin by explaining my design. I started by creating a class, I chose to use “ItemFrequency”. I used “ItemFrequency” to encapsulate the data and operations related to item frequencies from the txt Document.

For private members like on line 13 “map<string, int> itemCounts;” this map is used for efficient lookups and storage. It’s a core data structure that stores item names aka strings as keys and their corresponding frequencies aka integers as values. On line 14 “string inputFileName;” stores the name of the input file where information is read from and on line 15 string “outputFileName;” stores the name of the output file aka “backupfile” to which item frequency data is written to. On line 18 there is “LoadData()” it’s a function that opens the input file, reads the information within and stores the number of times each item appears within the “itemCounts” map. On line 35 “backupData()” opens the output file and then writes the information that is held within the “itemCounts” map into the output file.

For public members on line 51, I created a Constructor that initializes the “inputFileName” and “outputFileName” members. It also calls “loadData()” to read data from the input file and “backupData()” to create a backup of the data. If either of these operations fail, the program exits . Next on line 61 the “getItemFrequency(const string& item) const” takes an item name as input and returns its frequency. It uses “itemCounts.find()” to efficiently look up the item in the map. If the item is not found, it returns 0. On line 72 “printAllFrequencies() const:” is used to iterate through the “itemCounts” map and prints each item and its frequency. On line 79 “printHistogram() const:” generates a text-based histogram of item frequencies that iterates through the “itemCounts” map and prints each item name, followed by the number of asterisks equal to the item's frequency. The const keyword indicates that this function does not modify the object's state. On line 95 the “displayMenu():” function presents a menu to the user, allowing them to choose from different operations (find item frequency, print all frequencies, print histogram and exit). It handles user input, validates it, and calls the appropriate functions based on the user's choice. And finally the “main()” function creates an instance of the “ItemFrequency” class, then calls the “displayMenu()” function to start the program's interactive menu.

As for my thought process and usage of C++ I utilized an Object-oriented approach. I used the “ItemFrequency” class to encapsulate the data and operations related to item frequencies, which hides the data within the class and only exposes methods that are required for functionality, making the code organized and maintainable. I used data structures because they provide efficient key-value storage and lookups which are essential for quickly accessing item frequencies. I included “ifstream” and “ofstream” for reading and writing to files because it allowed the program to load data from the txt file and create a backup of the data in the output file which was required for this project. This is also where I include components of the optional error handling challenge because cases where the file cannot open and instances of the user wanting to exit are now managed. “iostream” and “iomanip” are used for console input/output and formatting the output for the histogram here is another error handling aspect that verifies the user input for correct values. The code includes error handling for file opening, the data type of the user input and if the menu option exists. I used the “<algorithm>” library because I used the “max()” function to find the maximum length of the item names for formatting the histogram. I used the “<limits>” library to get the maximum value of “streamsize” which is used to clear the input buffer. I used the const so that objects’ states aren’t unintendedly modified and this helped to simplify my code. Finally, my use of the “this->” pointer is used to explicitly call the member functions of the object.