Justin Phillips, 10/15/23

For ChadaTech and Corner Grocer

This design document outlines and selectively highlights the functionality of a program designed for ChadaTech and Corner Grocer which analyzes an input shopping list and returns the frequency of its items in an output file as well as presents a user-friendly interface to display the information from the list.

End usage:

A screenshot of a computer

Description automatically generated

The program must be ran using a C++ compiler. I recommend using Visual Studio 2019. This is where the source was developed. The program is confused by non-integer inputs on this screen which I could not fix.

The functions here are explained by the menu. I f 1) is selected the user can search the grocery list for a particular item. If an item is not found, then it will inform the user. If the item is found, then it will represent the frequency to the user.

Similarly, 2 and 3 will display the full list of unique items and their corresponding frequencies, the difference between them being that 3 will display a histogram for the end user.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Developing the program

In main.cpp is where I began to write my file. I started with a generic inclusion of iostream, a native C++ library which allows me to read user input and present output to the console. I also included the string native library and declared ‘using namespace std’. These tend to be practices I employ at the inception of each file in order to make things easier later on. Many C++ functions are in namespace standard, and declaring that as my active namespace allows me to skip specifying std::function() whenever I call a function.

I also declared globals at the top of my main.cpp which I wanted to be easily modifiable by an end user, such as the input file (this shipped as “GroceryList.txt”), histogram\_char\_string, a string that holds the histogram display ‘block’, be that an asterisk or one of the many characters from ASCII that are just as well suited (i.e. 𺆼), and the desired output file, which is set to frequency.dat in accordance to the specification documents.

I then did another set of generic practices for my main() declaration – I declared it as an integer return type function and wrapped the meat and potatoes of the code in a while true loop, which ensures that my loop will be running until it is either interrupted by the user exiting the program or an error occurs.

A screen shot of a computer program

Description automatically generated

To handle the functionality of the menu, I declared a switch statement which gets an input in as the switch’s director, then depending on the numerical input sends the program into a particular directory. In accordance with the given specifications, I outlined four cases and a default case, which handles incorrect values. Finally, I put a pause statement that executes whenever the switch is exited, which occurs every case due to the break statements. The pause gives an opportunity for the end user to read any given case’s output. 4 is the case chosen to exit the program altogether by interrupting the loop and returning the program’s exit code as 0.

A screen shot of a computer program

Description automatically generated

I then created a header and an empty file to handle the menu library class which would hold all my functions, being 1) frequencyBySearch, 2) frequencyList, 3) histogram, all of which were named to be memorable and be descriptive. My MenuFunctions header just declared these functions with arguments for the readfile in and the writefile out wherever it was appropriate, with histogram having its own argument for the display character string global. In menuFunctions.cpp, I have the bulk of my code. All the functions above are defined here to be called by main later. I wrote an initialize function that I could call when I wanted rather than the declaration of the new MenuFunctions class in main(), which I found cleaner for me. My initialization would process the input document and record the unique grocery item keys and their frequencies of occurrence to the output document. Checkers whenever a document is opened are present throughout each function, as well as closers.

Despite attempts to rectify it, I did have issues with recasting a c-inned variable that selected for the menu, and as a result a non-integer value in the menu will result in an infinitely looping menu display. I then remembered cin.clear(), which will allow me to reset the input next time I have a cin variable, however this did not work.

I took advantage of the map object to create an array of paired strings and ints for the key and frequency. I also created a set to store all the unique keys. The frequency of a key could be increased whenever that key was identified and incremented while the set held each unique key.

A screen shot of a computer program

Description automatically generated

I also made a for loop to iterate for all items in the key, from the start of the key to the end of the key, with the iteration counter increasing every iteration, and used this to print all my entries.

A computer code on a black background

Description automatically generated

This enter process is similar between the histogram and normal frequency displays.

frequencyList() merely reads and prints each line of the original output file from initialization() because the printed data is identical to the data stored in this case, and is therefore redundant.

A screen shot of a computer program

Description automatically generated

frequencyBySearch is more interesting than the other functions because I had to seek out the frequency of specific items by query. I was able to add case insensitivity here fairly easily by making the input and the version of the key taken from the source data both lowercase.

A screen shot of a computer program

Description automatically generated

Finally, histogram() has a for loop to print the histogram character as many times as the frequency is.

A screen shot of a computer code

Description automatically generated