Joseph Cassello Jr.

CS210 – Corner Grocer

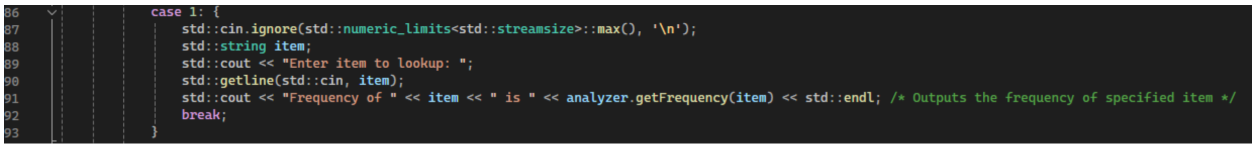
06 / 23 / 2024

Overview:

The Corner Grocer program I designed serves as an item-tracking tool aimed at optimizing the layout of their produce section based on purchase records. By utilizing C++, I was able to create a program that employs a menu driven interface facilitated by the "FrequencyAnalyzer" class

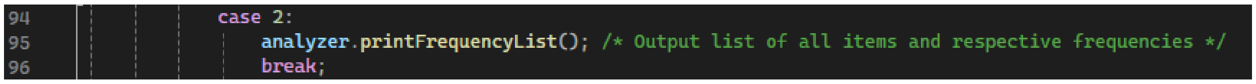
Main Functionality and User Interface

The first option the interface presents to the user is to get the frequency of a specific item. A user can input an item's name to retrieve its frequency from the purchase records.



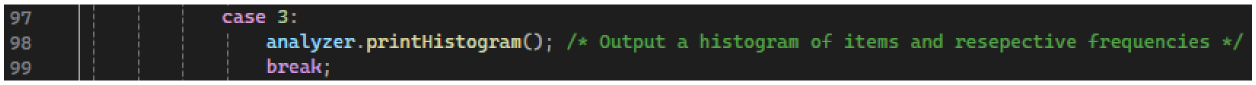
```cpp
            case 1: {
                std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
                std::string item;
                std::cout << "Enter item to lookup: ";
                std::getline(std::cin, item);
                std::cout << "Frequency of " << item << " is " << analyzer.getFrequency(item) << std::endl; /* Outputs the frequency of specified item */
                break;
            }
```

The second option outputs a list of all purchased items along with their respective frequencies. Data is read from the file "CS210_Project_Three_Input_file.txt."



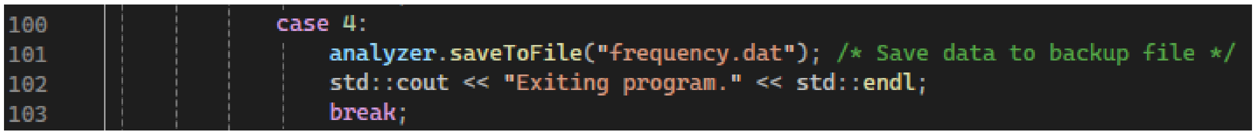
```cpp
            case 2:
                analyzer.printFrequencyList(); /* Output list of all items and respective frequencies */
                break;
```

The third option allows users to visualize item frequencies through a histogram representation, where each item's frequency is depicted as asterisks instead.



```cpp
            case 3:
                analyzer.printHistogram(); /* Output a histogram of items and resepective frequencies */
                break;
```

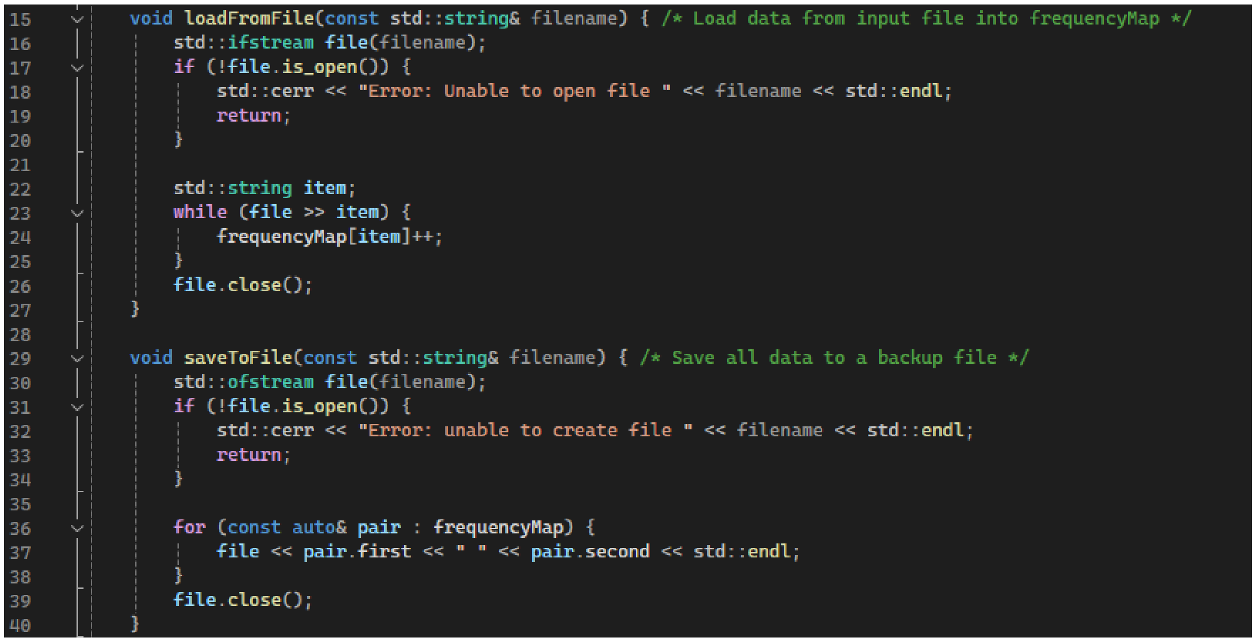
The fourth and final option exits the program, but before doing so, all data is saved as a backup to "frequency.dat."



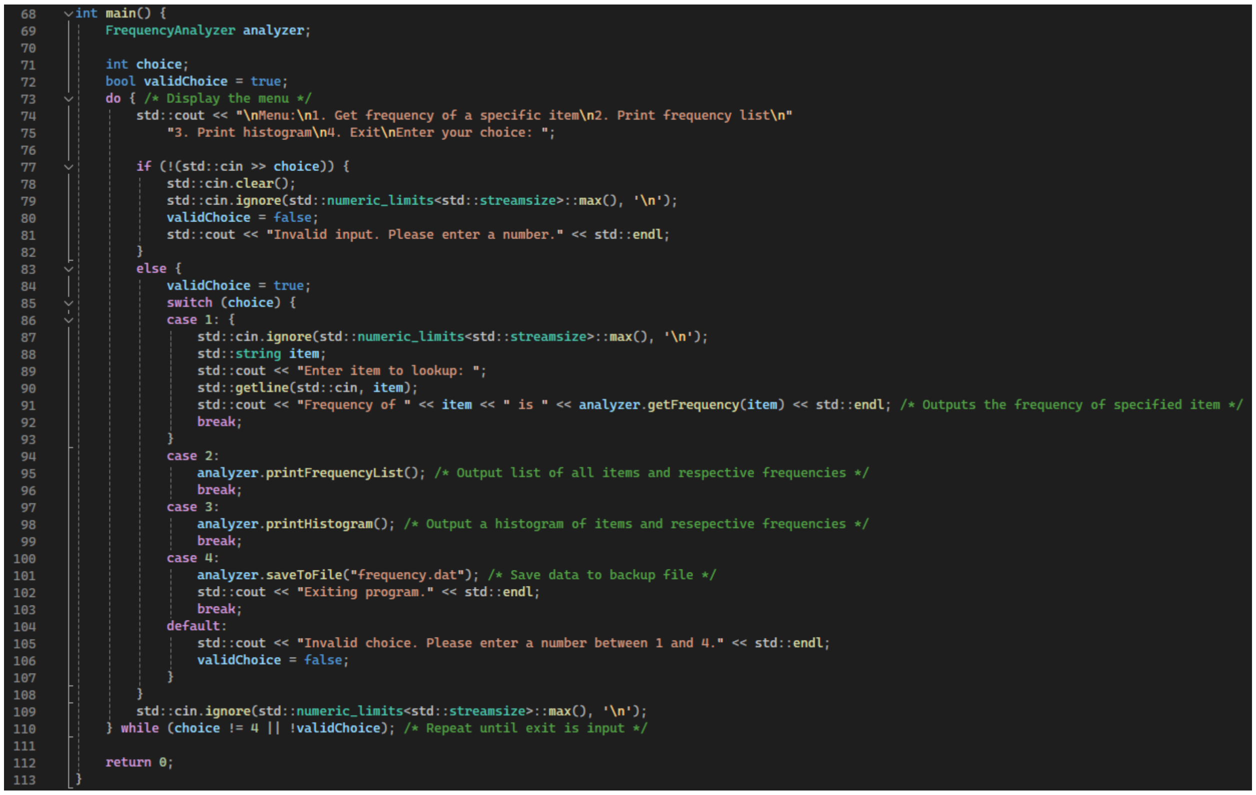
```
            case 4:
                analyzer.saveToFile("frequency.dat"); /* Save data to backup file */
                std::cout << "Exiting program." << std::endl;
                break;
```

Error Handling:

In my code I incorporated error handling mechanisms to ensure a smooth experience for the user. I used standard C++ input functions to validate user input, ensuring that only numeric choices between 1 and 4 are accepted. I also created error messages that display when the program encounters issues with opening or creating files.



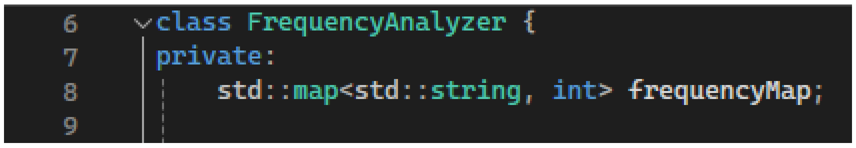
```
    void loadFromFile(const std::string& filename) { /* Load data from input file into frequencyMap */
        std::ifstream file(filename);
        if (!file.is_open()) {
            std::cerr << "Error: Unable to open file " << filename << std::endl;
            return;
        }

        std::string item;
        while (file >> item) {
            frequencyMap[item]++;
        }
        file.close();
    }

    void saveToFile(const std::string& filename) { /* Save all data to a backup file */
        std::ofstream file(filename);
        if (!file.is_open()) {
            std::cerr << "Error: unable to create file " << filename << std::endl;
            return;
        }

        for (const auto& pair : frequencyMap) {
            file << pair.first << " " << pair.second << std::endl;
        }
        file.close();
    }
```



```cpp
int main() {
    FrequencyAnalyzer analyzer;

    int choice;
    bool validChoice = true;
    do { /* Display the menu */
        std::cout << "\nMenu:\n1. Get frequency of a specific item\n2. Print frequency list\n"
            "3. Print histogram\n4. Exit\nEnter your choice: ";

        if (!(std::cin >> choice)) {
            std::cin.clear();
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
            validChoice = false;
            std::cout << "Invalid input. Please enter a number." << std::endl;
        }
        else {
            validChoice = true;
            switch (choice) {
            case 1: {
                std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
                std::string item;
                std::cout << "Enter item to lookup: ";
                std::getline(std::cin, item);
                std::cout << "Frequency of " << item << " is " << analyzer.getFrequency(item) << std::endl; /* Outputs the frequency of specified item */
                break;
            }
            case 2:
                analyzer.printFrequencyList(); /* Output list of all items and respective frequencies */
                break;
            case 3:
                analyzer.printHistogram(); /* Output a histogram of items and resepective frequencies */
                break;
            case 4:
                analyzer.saveToFile("frequency.dat"); /* Save data to backup file */
                std::cout << "Exiting program." << std::endl;
                break;
            default:
                std::cout << "Invalid choice. Please enter a number between 1 and 4." << std::endl;
                validChoice = false;
            }
        }
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    } while (choice != 4 || !validChoice); /* Repeat until exit is input */

    return 0;
}
```

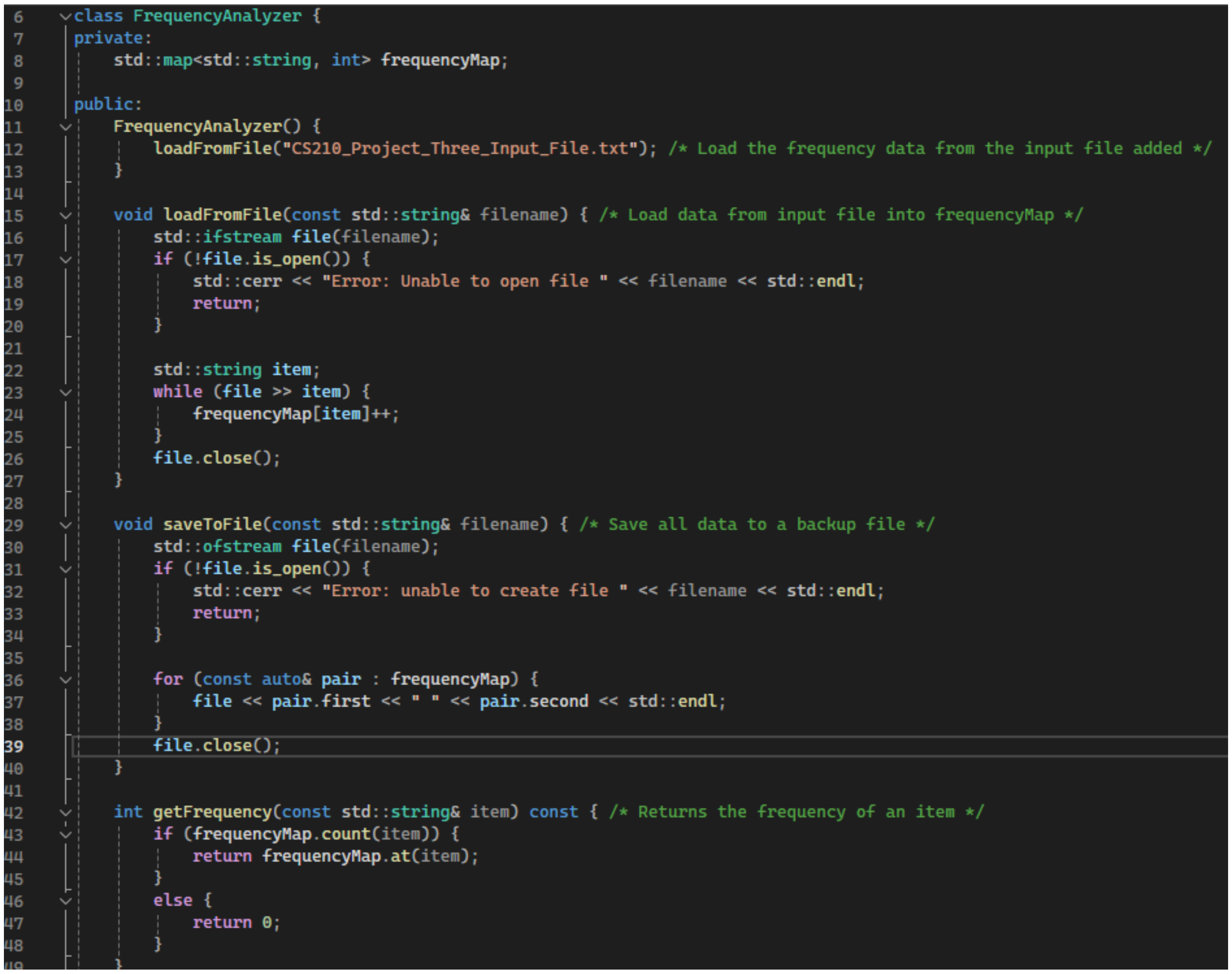
Class Structure:

My class 'FrequenceAnalyzer,' manages and analyzes purchase records for the Corner Grocer. It includes a private member ‘frequencyMap’, which stores item names and their purchase frequencies. The class has a constructor that loads initial data from a specified file into 'frequencyMap.'

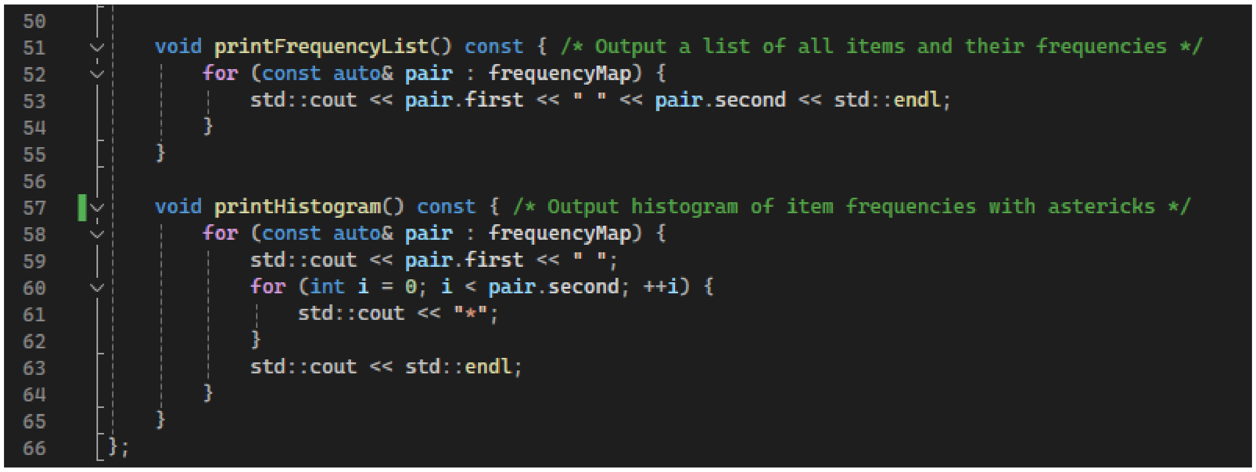


```cpp
class FrequencyAnalyzer {
private:
    std::map<std::string, int> frequencyMap;

```

Public member functions include 'loadFromFile()' to read additional data, ‘saveToFile()’ to back up the data, ‘getFrequency()’ to retrieve item frequencies, 'printFrequencyList()' to list all items and tehri frequencies, and 'printHistogram()' to visualize frequencies with asterisks.



```cpp
class FrequencyAnalyzer {
private:
    std::map<std::string, int> frequencyMap;

public:
    FrequencyAnalyzer() {
        loadFromFile("CS210_Project_Three_Input_File.txt"); /* Load the frequency data from the input file added */
    }

    void loadFromFile(const std::string& filename) { /* Load data from input file into frequencyMap */
        std::ifstream file(filename);
        if (!file.is_open()) {
            std::cerr << "Error: Unable to open file " << filename << std::endl;
            return;
        }

        std::string item;
        while (file >> item) {
            frequencyMap[item]++;
        }
        file.close();
    }

    void saveToFile(const std::string& filename) { /* Save all data to a backup file */
        std::ofstream file(filename);
        if (!file.is_open()) {
            std::cerr << "Error: unable to create file " << filename << std::endl;
            return;
        }

        for (const auto& pair : frequencyMap) {
            file << pair.first << " " << pair.second << std::endl;
        }
        file.close();
    }

    int getFrequency(const std::string& item) const { /* Returns the frequency of an item */
        if (frequencyMap.count(item)) {
            return frequencyMap.at(item);
        }
        else {
            return 0;
        }
    }
```



```cpp

    void printFrequencyList() const { /* Output a list of all items and their frequencies */
        for (const auto& pair : frequencyMap) {
            std::cout << pair.first << " " << pair.second << std::endl;
        }
    }

    void printHistogram() const { /* Output histogram of item frequencies with astericks */
        for (const auto& pair : frequencyMap) {
            std::cout << pair.first << " ";
            for (int i = 0; i < pair.second; ++i) {
                std::cout << "*";
            }
            std::cout << std::endl;
        }
    }
};
```