

Corner Grocer

The Corner Grocer Item Tracking Program was developed to analyze daily grocery purchase records and determine how frequently each item was purchased. The program reads a text file containing item names listed in chronological order throughout the business day. It processes this data to calculate purchase frequencies and provides an interactive menu that allows the user to search for specific items, view all item frequencies, and display a text-based histogram. In addition to this, the program automatically generates a backup data file named frequency.dat at startup. This file stores each item and its corresponding purchase count to preserve the accumulated data for future reference.

Class Design & Structure

The program was designed using object-oriented principles. A single class named GroceryTracker was created to encapsulate all functionality related to file processing and frequency tracking. The class contains one private data member: `map<string, int> itemCounts`. This map stores each grocery item as a key & its frequency as the associated value. Using a map was an intentional design choice because it efficiently stores key value pairs and automatically handles unique keys. When reading items from the file, if an item does not yet exist in the map, it is automatically created with a value of zero then incremented. This makes frequency tracking both clean and efficient.

This class contains the following public member functions: `LoadFromFile()` which reads input data & builds frequency counts, `WriteBackupFile()` which creates the `frequency.dat` backup file, `GetItemFrequency()` which returns the count for a specific item, `PrintAllFrequencies()` which displays all items & counts, and `PrintHistogram()` which displays a visual representation using asterisks. Separating these responsibilities into member functions improves readability, maintainability, and organization of the code.

The screenshot shows the Microsoft Visual Studio IDE interface. The code editor displays a C++ file named 'CornerGrocerTracker.cpp' containing the implementation of a 'GroceryTracker' class. The class has private members for a map of item counts and a static method to convert strings to lowercase. It also includes methods for loading from a file, writing to a backup file, and getting item frequencies. The Solution Explorer window shows a single project named 'CornerGrocerTracker' with files like 'HeaderFiles', 'Resource Files', and 'Source Files' (containing 'CornerGrocerTracker.cpp'). The Properties window is visible on the right. The Output window at the bottom shows a message about symbol loading.

```
class GroceryTracker {
private:
    //Map to store each item & its frequency count
    map<string, int> itemCounts;

    // Converts a string to Lowercase so searches can be case insensitive
    static string ToLower(string str) {
        for (char& c : str) {
            c = static_cast<char>(tolower(static_cast<unsigned char>(c)));
        }
        return str;
    }

public:
    // Loads item data from the input file
    void LoadFromFile(const string& fileName);

    //Writes all item frequencies to a backup file
    void WriteBackupFile(const string& fileName) const;

    //Returns how many times a specific item appears
    int GetItemFrequency(const string& item) const;
};
```

File Handling & Data Processing

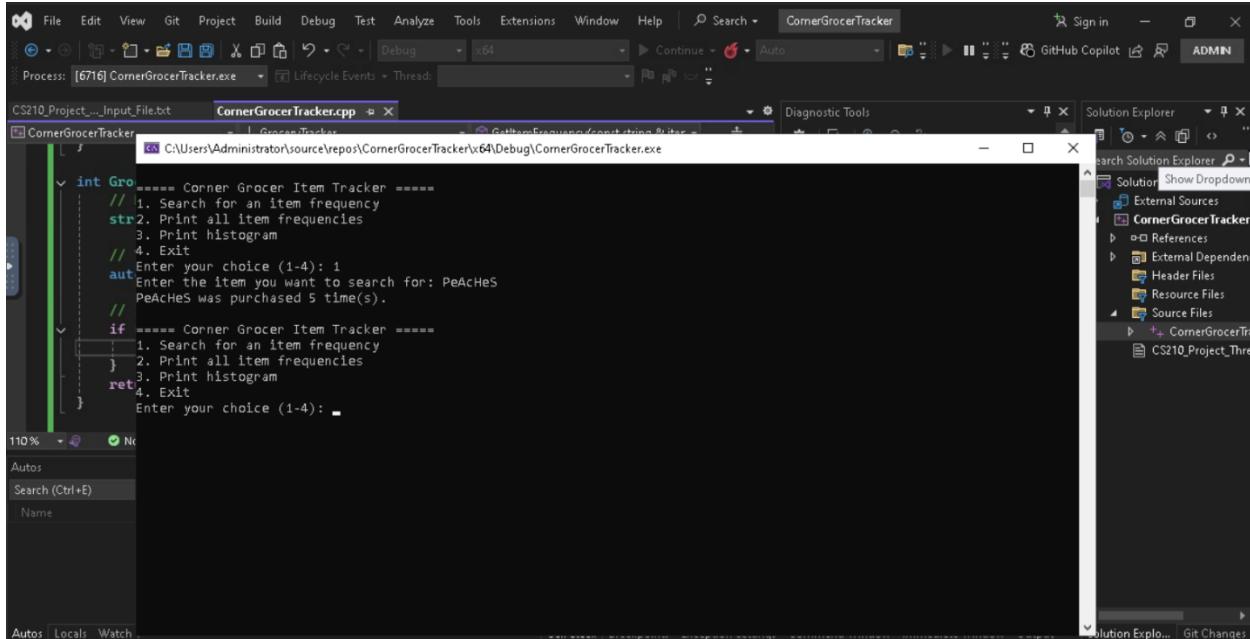
The program begins by calling `LoadFromFile()`, which attempts to open the file `CS210_Project_Three_Input_File.txt`. File validation is performed using `is_open()` to ensure the file loads successfully before processing. Items are read using a `while (inFile >> item)` loop, which continues until the end of the file is reached. Each item is normalized to lowercase before being stored in the map to ensure consistent frequency tracking. Immediately after loading the file, the program calls `WriteBackupFile()` to create `frequency.dat`. This function writes each item & its frequency to the backup file without requiring user input. This ensures data persistence and satisfies the assignment requirement for automatic backup creation.

The screenshot shows a terminal window displaying the contents of a file named 'frequency.dat'. The file contains a list of items and their frequencies, separated by spaces. The items listed are apples, beets, broccoli, cantaloupe, cauliflower, celery, cranberries, cucumbers, garlic, limes, onions, peaches, pears, peas, potatoes, pumpkins, radishes, spinach, yams, and zucchini. Each item is followed by its corresponding frequency value.

```
apples 4
beets 3
broccoli 7
cantaloupe 2
cauliflower 6
celery 6
cranberries 10
cucumbers 9
garlic 8
limes 1
onions 4
peaches 5
pears 1
peas 8
potatoes 5
pumpkins 2
radishes 3
spinach 5
yams 5
zucchini 10
```

Case insensitive Search Enhancement

To improve usability, the program implements case-insensitive searching. Without this enhancement, searching for “Peaches”, “peaches”, or “PEACHES” would produce different results. A helper function named ToLower() was created to convert strings to lowercase. This function is used when reading items from the input file, and also when processing user search input. By standardizing all stored data and search terms to lowercase, the program ensures accurate & consistent matching regardless of how the user types the item name.



```
int Groc ===== Corner Grocer Item Tracker =====
    // 1. Search for an item frequency
    str2. Print all item frequencies
        3. Print histogram
    // 4. Exit
    aut Enter your choice (1-4): 1
    Enter the item you want to search for: PeAches
    PeAches was purchased 5 time(s).
    //
    if ===== Corner Grocer Item Tracker =====
        1. Search for an item frequency
    } 2. Print all item frequencies
        3. Print histogram
    ret4. Exit
    Enter your choice (1-4):
```

Menu System & Input Validation

The program includes an interactive menu that allows the user to select from four options. 1. Search for an item frequency. 2. Print all item frequencies. 3. Print a histogram. 4. Exit the program. The menu runs inside a loop that continues until the user selects option 4. Basic input validation is implemented using `cin.fail()` to prevent the program from crashing if a user enters invalid input (such as letters instead of numbers). If invalid input is detected the input stream is cleared and the user is prompted again. This improves readability and user experience.

```
==== Corner Grocer Item Tracker ====
1. Search for an item frequency
2. Print all item frequencies
3. Print histogram
4. Exit
Enter your choice (1-4): q
Invalid input. Please enter a number 1-4.

==== Corner Grocer Item Tracker ====
1. Search for an item frequency
2. Print all item frequencies
3. Print histogram
4. Exit
Enter your choice (1-4):
```

Histogram Representation

The histogram feature visually represents item frequency by printing one asterisk (*) per record purchase. This provides a quick and intuitive visual comparison of purchase trends, helping the store determine how to reorganize its produce section effectively.

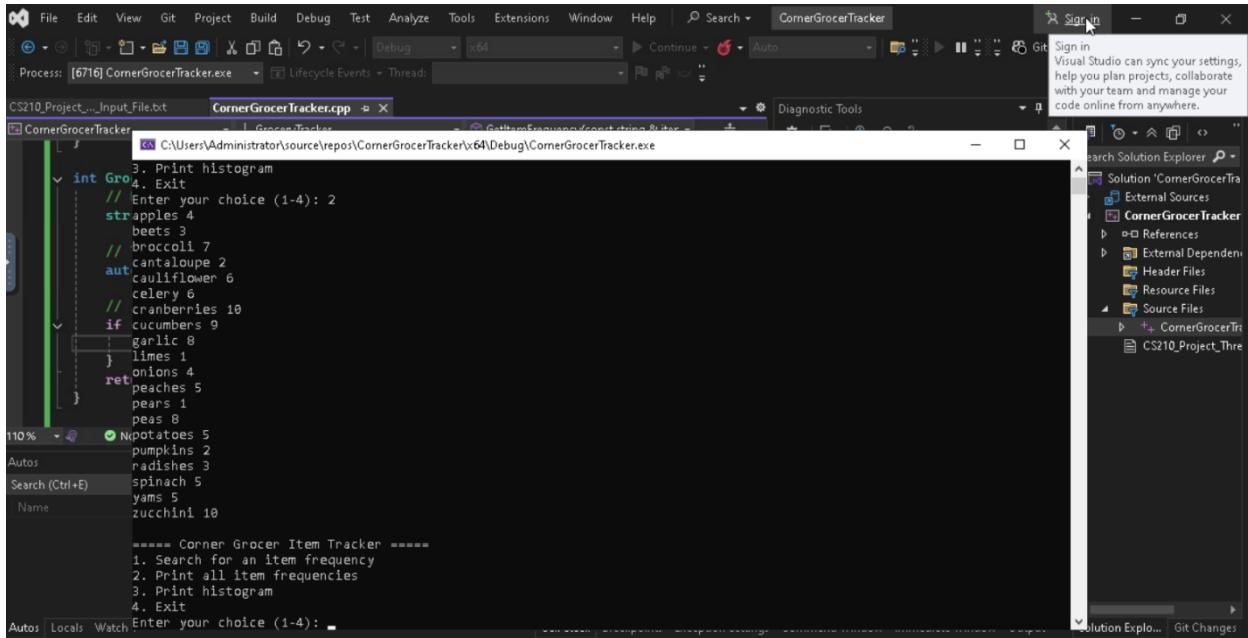
```
==== Corner Grocer Item Tracker ====
3. Print histogram
4. Exit
// Enter your choice (1-4): 3
str
apples ****
beets **
broccoli *****
aut
cantaloupe **
cauliflower *****
celery *****
cranberries *****
if cucumbers *****
garlic *****
limes *
onions ***
ret
peaches *
pears *
peas *****
potatoes *****
pumpkins **
radishes ***
spinach *****
yams *****
zucchini *****

===== Corner Grocer Item Tracker =====
1. Search for an item frequency
2. Print all item frequencies
3. Print histogram
4. Exit
Enter your choice (1-4):
```

Program Execution & Testing

The program was tested using multiple search inputs, including variations in capitalization,

to verify that case-insensitive searching functions correctly. All menu options were tested to ensure accurate frequency reporting and correct histogram output. The backup file frequency.dat was also verified to confirm that it contains the correct item counts immediately upon program execution.



The screenshot shows the Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar. The title bar says "CornerGrocerTracker". The main window displays the code for "CornerGrocerTracker.cpp" and the output window shows the program's execution. The output window content is as follows:

```
CS210_Project_Thre_Input_File.txt CornerGrocerTracker.cpp
C:\Users\Administrator\source\repos\CornerGrocerTracker\x64\Debug\CornerGrocerTracker.exe
3. Print histogram
4. Exit
Enter your choice (1-4): 2
apples 4
beets 3
broccoli 7
cantaloupe 2
autocauliflower 6
celery 6
cranberries 10
if cucumbers 9
garlic 8
limes 1
onions 4
retpeaches 5
pears 1
peas 8
potatoes 5
pumpkins 2
radishes 3
spinach 5
yams 5
zucchini 10

===== Corner Grocer Item Tracker =====
1. Search for an item frequency
2. Print all item frequencies
3. Print histogram
4. Exit
Enter your choice (1-4):
```

The Solution Explorer on the right shows the project structure with files like "CornerGrocerTracker.h", "CornerGrocerTracker.cpp", and "CS210_Project_Thre.h".

In conclusion, the Corner Grocer Item Tracking Program successfully meets all functional requirements outlined. It demonstrates proper use of file input and output, object oriented design, associative containers (maps), user interaction, input validation, and data normalization. The program is organized, readable, and maintainable, following industry standard best practices such as inline comments, clear naming conventions, and structured logic flow.

References

- Vahid, F., & Lysecky, R. (2019). CS 210: Programming languages. zyBooks.
<http://www.zybooks.com>