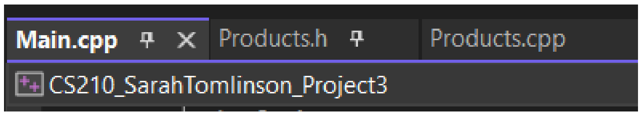
To keep the main function as simple as possible I used a "Products" class with a private map and public functions to run the program. To further follow best practices, this is done across three files "Main.cpp", "Products.h", and "Products.cpp". Products.h contains the class and function declarations, while Products.cpp contains the class definitions and main functionality of the entire program.

The class has a private map and 5 public functions to read the file, write a data file, print the menu to the user, run the main menu and get user choice, and a function to help with formatting output to the user. I felt it was important to implement this as a class so that additional product lists and maps can be created without needing to alter the first map. Multiple output files could also be created easily if necessary to track all the data separately. The program is coded in such a way that if different files needed to be read or input by the user the program could easily be updated to accept a file name to open from the user or only needing to change the name of the file in one location (Main).

I also made sure to check for user input errors as well to keep the program running as intended even if the user makes a mistake. If the user enters an incorrect input at the menu phase the program will output a relevant warning message and reprint the menu to the user asking for their input again. If the user wishes to get the count for a specific item, it needs to be typed exactly as it is from original document. Instead of including this error in the try/catch exception handling, I decided to use the "count" function of maps. This searches for the user's input in the map keys and if it doesn't exist simply advises the user that their requested item isn't in the list and reprints the map asking for another menu choice. Below are photos detailing the items I mentioned here as well as the overall structure of the code.

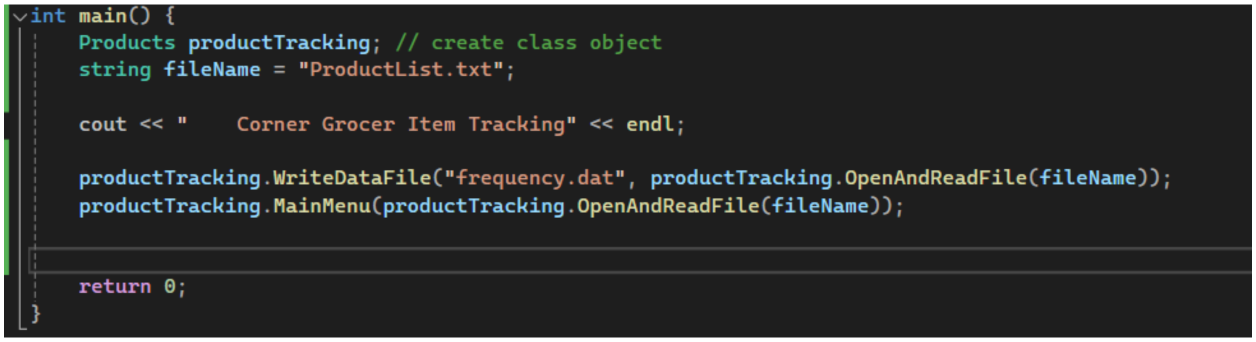
Multiple Files Screenshot



Main.cpp Products.h Products.cpp

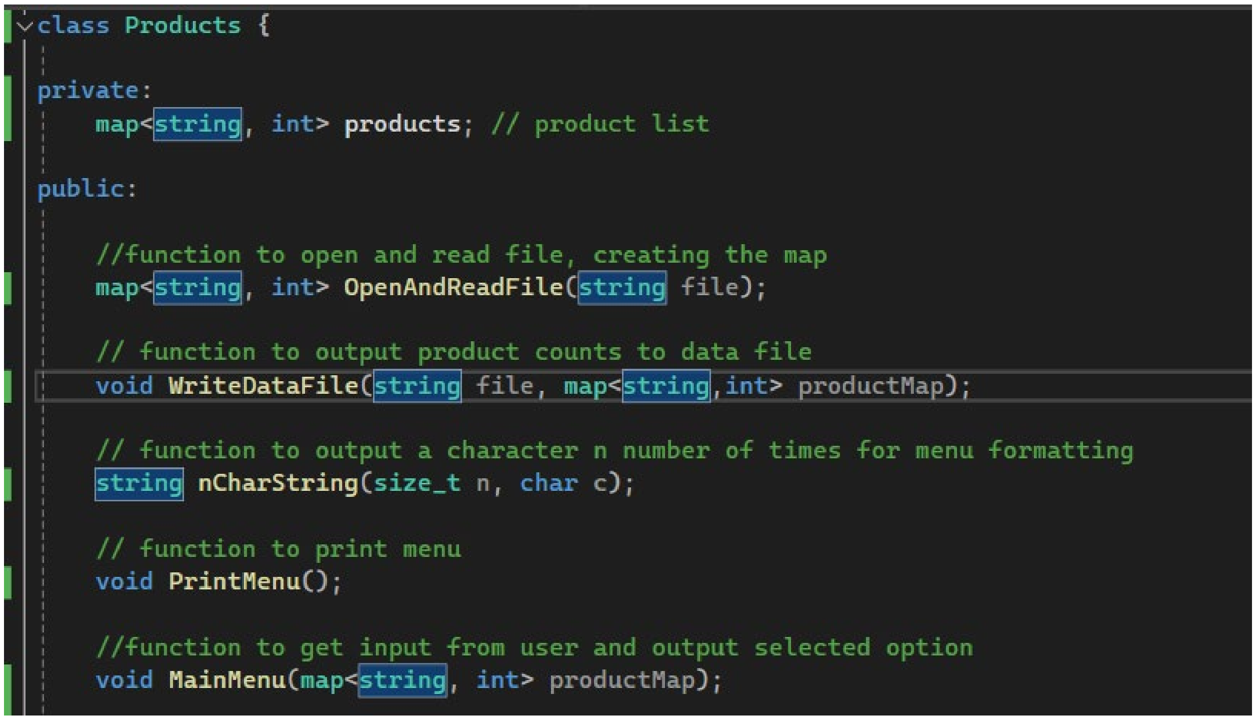
CS210_SarahTomlinson_Project3

Main Function



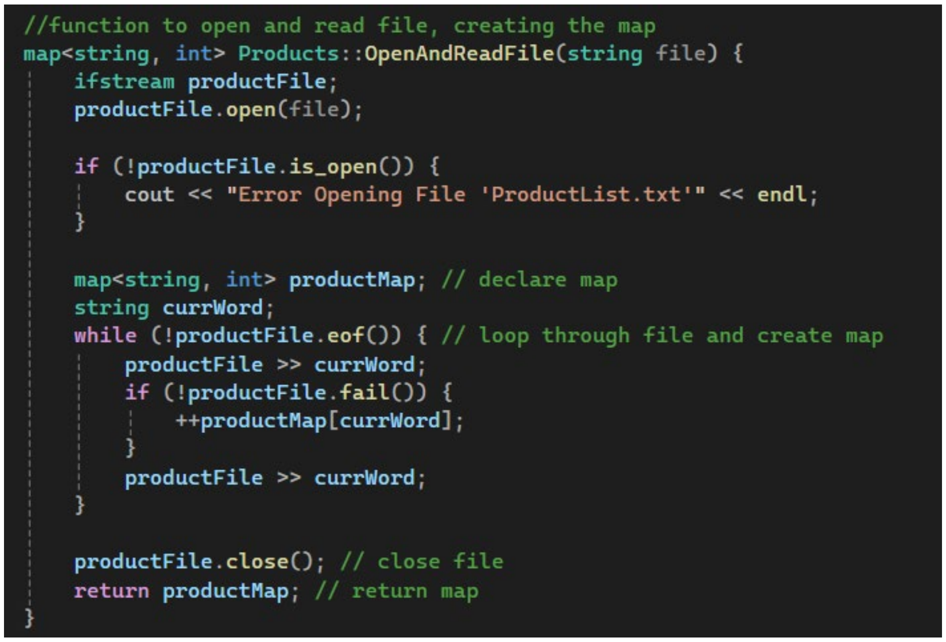
```
int main() {
    Products productTracking; // create class object
    string fileName = "ProductList.txt";

    cout << "    Corner Grocer Item Tracking" << endl;

    productTracking.WriteDataFile("frequency.dat", productTracking.OpenAndReadFile(fileName));
    productTracking.MainMenu(productTracking.OpenAndReadFile(fileName));


    return 0;
}
```

Class Products



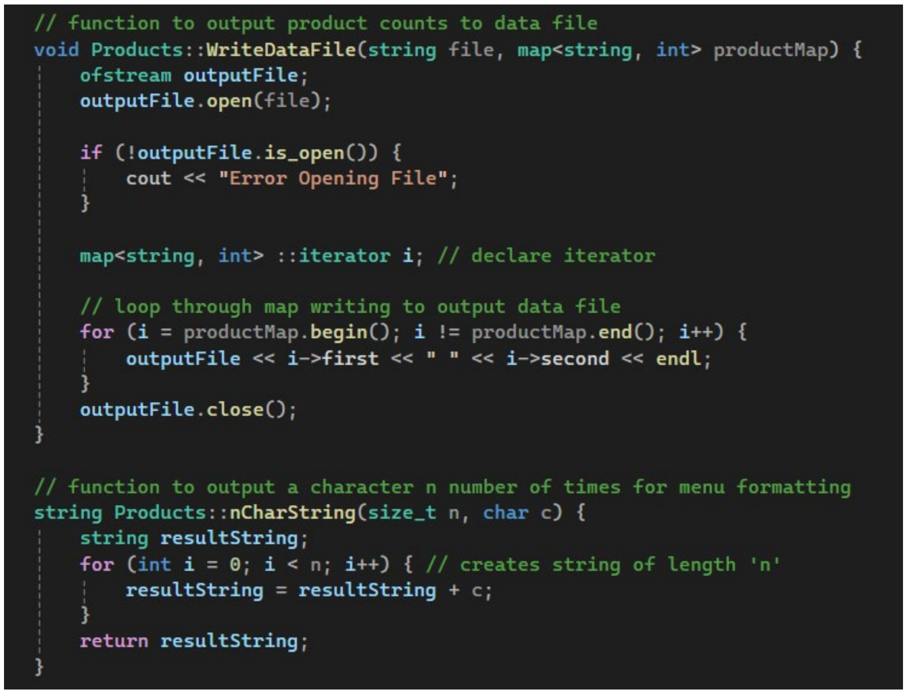
```
class Products {

private:
    map<string, int> products; // product list

public:

    //function to open and read file, creating the map
    map<string, int> OpenAndReadFile(string file);

    // function to output product counts to data file
    void WriteDataFile(string file, map<string,int> productMap);

    // function to output a character n number of times for menu formatting
    string nCharString(size_t n, char c);

    // function to print menu
    void PrintMenu();

    //function to get input from user and output selected option
    void MainMenu(map<string, int> productMap);
```

Function to open and read file



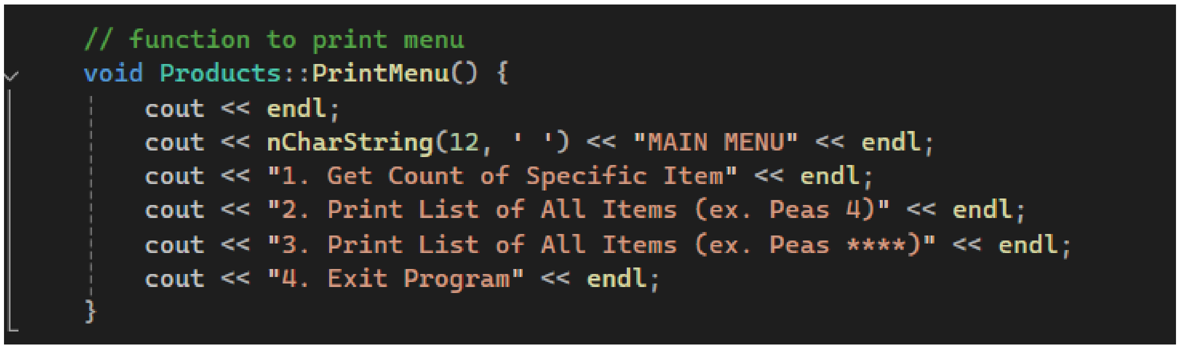
```cpp
//function to open and read file, creating the map
map<string, int> Products::OpenAndReadFile(string file) {
    ifstream productFile;
    productFile.open(file);

    if (!productFile.is_open()) {
        cout << "Error Opening File 'ProductList.txt'" << endl;
    }

    map<string, int> productMap; // declare map
    string currWord;
    while (!productFile.eof()) { // loop through file and create map
        productFile >> currWord;
        if (!productFile.fail()) {
            ++productMap[currWord];
        }
        productFile >> currWord;
    }

    productFile.close(); // close file
    return productMap; // return map
}
```

Functions to Write Data file and format menu



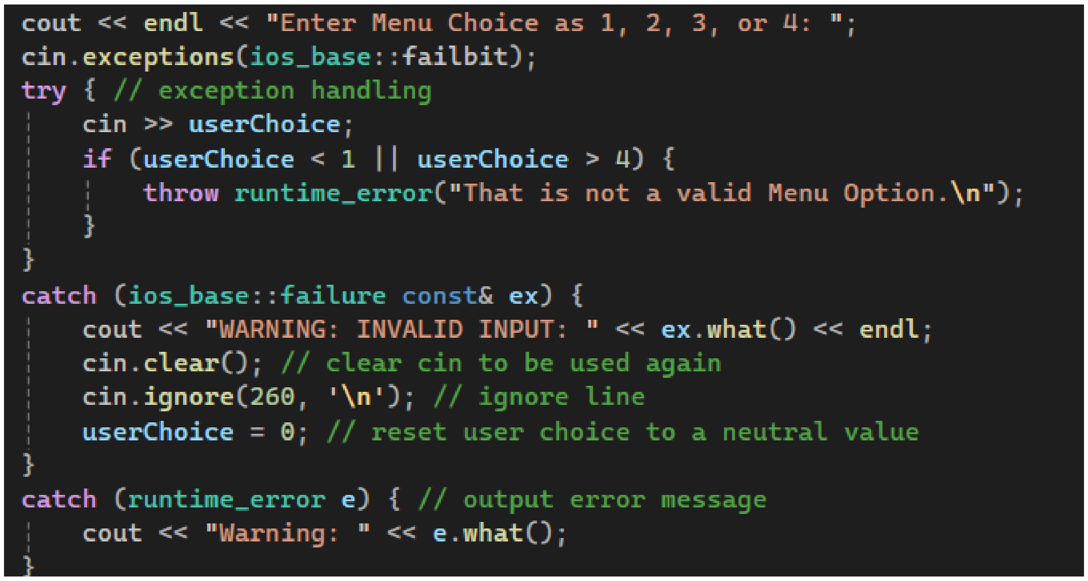
```cpp
// function to output product counts to data file
void Products::WriteDataFile(string file, map<string, int> productMap) {
    ofstream outputFile;
    outputFile.open(file);

    if (!outputFile.is_open()) {
        cout << "Error Opening File";
    }

    map<string, int> ::iterator i; // declare iterator

    // loop through map writing to output data file
    for (i = productMap.begin(); i != productMap.end(); i++) {
        outputFile << i->first << " " << i->second << endl;
    }
    outputFile.close();
}

// function to output a character n number of times for menu formatting
string Products::nCharString(size_t n, char c) {
    string resultString;
    for (int i = 0; i < n; i++) { // creates string of length 'n'
        resultString = resultString + c;
    }
    return resultString;
}
```

Function to print menu to user



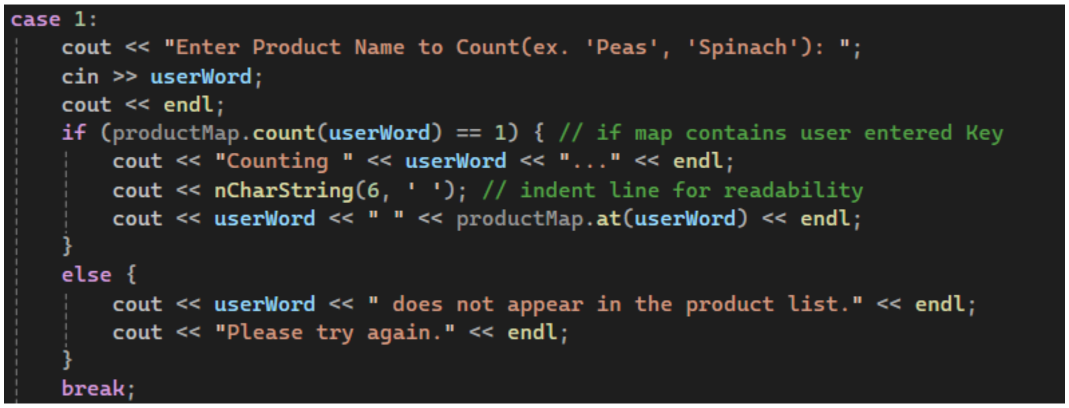
```
// function to print menu
void Products::PrintMenu() {
    cout << endl;
    cout << nCharString(12, ' ') << "MAIN MENU" << endl;
    cout << "1. Get Count of Specific Item" << endl;
    cout << "2. Print List of All Items (ex. Peas 4)" << endl;
    cout << "3. Print List of All Items (ex. Peas ****)" << endl;
    cout << "4. Exit Program" << endl;
}
```

Exception Handling



```
cout << endl << "Enter Menu Choice as 1, 2, 3, or 4: ";
cin.exceptions(ios_base::failbit);
try { // exception handling
    cin >> userChoice;
    if (userChoice < 1 || userChoice > 4) {
        throw runtime_error("That is not a valid Menu Option.\n");
    }
}
catch (ios_base::failure const& ex) {
    cout << "WARNING: INVALID INPUT: " << ex.what() << endl;
    cin.clear(); // clear cin to be used again
    cin.ignore(260, '\n'); // ignore line
    userChoice = 0; // reset user choice to a neutral value
}
catch (runtime_error e) { // output error message
    cout << "Warning: " << e.what();
}
```

Check that map contains user input key



```
case 1:
    cout << "Enter Product Name to Count(ex. 'Peas', 'Spinach'): ";
    cin >> userWord;
    cout << endl;
    if (productMap.count(userWord) == 1) { // if map contains user entered Key
        cout << "Counting " << userWord << "..." << endl;
        cout << nCharString(6, ' '); // indent line for readability
        cout << userWord << " " << productMap.at(userWord) << endl;
    }
    else {
        cout << userWord << " does not appear in the product list." << endl;
        cout << "Please try again." << endl;
    }
    break;
```