BlackRock Stock Analyzer

Anjana Niranjanan

Requirements

- 50 for Functioning of Source Code and Program Complexity
  - Complexity of the program
  - Mandatory implementation of long-term data storage.
  - Unique features and implementations.
  - Program Functions without errors.
  - Code well documented(comments explaining the functioning source code for the classes and methods)
- 50 Write up and YouTube Video
  - YouTube video showcasing:
    - A PowerPoint presentation containing written descriptions of the program, GUI, program flow, and data structures
    - A demo of program running

50 for Functioning of Source Code and Program Complexity (Requirements)

- **Complexity of the program**
  - Uses custom class with multiple functions and a structured menu loop.
  - Combines vectors, file I/O, and data formatting for real-world analysis.
- **Mandatory implementation of long-term data storage.**
  - Loads stock data from a csv file using file input (ifstream).
  - Stores data in memory using a vector for later analysis and graphing.
- **Code well documented(comments explaining the functioning source code for the classes and methods)**
  - Each function has simple comments explaining what it does.
  - Key logic like loops, conditions, and file parsing are clearly labeled.
- **Unique features and implementations.**
  - ASCII line graph adds a visual way to view stock trends in the terminal.
  - Uses real BlackRock stock data, making it more practical and unique.
- **Program Functions without errors.**
  - All menu options were tested and work as expected with no crashes.
  - Program handles bad input or missing data safely and clearly.

main.cpp file

BlackRock Stock Analyzer App
main

BlackRock Stock Analyzer App 〉 My Mac Finished running BlackRo

BLK | StockAnalyzer | main | StockAnalyzer | BlackRock Stock Analyzer App

BlackRock Stock Analyzer App 〉 BlackRock Stock Analyzer App 〉 main 〉 No Selection

Find goodb

```cpp
// main.cpp
#include <iostream>
#include <limits>
#include <unistd.h>   // for getcwd
#include <climits>    // for PATH_MAX
#include "StockAnalyzer.h"

using namespace std;

// Displays the main menu options to the user
void showMenu() {
    cout << "\n---- BlackRock Stock Analyzer ----\n";
    cout << "1. Load Data\n";
    cout << "2. Average Closing Price\n";
    cout << "3. Show Line Graph\n";
    cout << "4. Search Closing Price by Date\n";
    cout << "5. Exit\n";
    cout << "Choose an option: ";
}

int main() {
    StockAnalyzer analyzer;
    int option;
    string filename = "BLK.csv"; // CSV file containing stock data

    do {
        showMenu();
        cin >> option;

        switch (option) {
            case 1:
                analyzer.loadData(filename);
                break;
            case 2:
                analyzer.printAverageClosingPrice();
                break;
            case 3:
                analyzer.printClosingPriceGraph();
                break;
            case 4: {
                string date;
                cout << "Enter date (YYYY-MM-DD): ";
                cin >> date;
                analyzer.searchByDate(date);
                break;
            }
            case 5:
                cout << "Thanks for using BlackRock Stock Analyzer!" << endl;
                break;
            default:
                cout << "Invalid option. Try again." << endl;
                break;
        }
    } while (option != 5);

    return 0;
}
```

main.cpp

- Acts as the main entry point for the program
- Displays a terminal-based menu with numbered options
- Uses a while (true) loop to keep the menu running until user exits
- Handles menu choices with a switch statement
- Connects user actions to the correct StockAnalyzer class methods
- Lets users:
    - Load stock data from a CSV
    - Calculate average closing price
    - Display an ASCII line graph of prices
    - Search closing price by date
    - Keeps logic simple and readable using only class functions and standard I/O

StockAnalyzer.cpp file

BlackRock Stock Analyzer App
BlackRock Stock Analyzer App

BlackRock Stock Analyzer App 〉 My Mac Finished running BlackRock Stock Analyzer App

BLK | StockAnalyzer | main | StockAnalyzer | BlackRock Stock Analyzer App

BlackRock Stock Analyzer App 〉 BlackRock Stock Analyzer App 〉 StockAnalyzer 〉 No Selection

```cpp
// StockAnalyzer.cpp
#include "StockAnalyzer.h"
#include <iomanip>
#include <cmath>
#include <unistd.h>
#include <limits.h>

// Helper function to trim whitespace and control characters
string trim(const string& str) {
    size_t first = str.find_first_not_of(" \r\n\t");
    if (first == string::npos) return "";
    size_t last = str.find_last_not_of(" \r\n\t");
    return str.substr(first, (last - first + 1));
}

void StockAnalyzer::loadData(string filename) {
    entries.clear();
    ifstream file(filename);
    if (!file.is_open()) {
        cout << "Failed to open file: " << filename << endl;
        return;
    }

    string line;
    getline(file, line); // Skip header

    while (getline(file, line)) {
        size_t commaPos = line.find(',');
        if (commaPos != string::npos) {
            string date = trim(line.substr(0, commaPos));
            string closeStr = trim(line.substr(commaPos + 1));
            double close = stod(closeStr);
            entries.push_back({ date, close });
        }
    }
    file.close();

    char cwd[PATH_MAX];
    getcwd(cwd, sizeof(cwd));
    cout << "Current Working Directory: " << cwd << endl;
    cout << "Loaded " << entries.size() << " stock entries.\n";
}

void StockAnalyzer::printAverageClosingPrice() {
    if (entries.empty()) {
        cout << "No data loaded.\n";
        return;
    }
    double sum = 0;
    for (auto& e : entries)
        sum += e.close;
    double average = sum / entries.size();
    cout << fixed << setprecision(3);
    cout << "Average Closing Price: $" << average << endl;
}

void StockAnalyzer::printClosingPriceGraph() {
    if (entries.empty()) {
        cout << "No data loaded.\n";
        return;
    }

    double maxClose = 0;
    for (auto& e : entries)
```

BlackRock Stock Analyzer App
main

BlackRock Stock Analyzer App 〉 My Mac Finished running Blackl

BLK | StockAnalyzer | main | StockAnalyzer | BlackRock Stock Analyzer App

BlackRock Stock Analyzer App 〉 BlackRock Stock Analyzer App 〉 StockAnalyzer 〉 No Selection

```cpp
void StockAnalyzer::printClosingPriceGraph() {
    double maxClose = 0;
    for (auto& e : entries)
        if (e.close > maxClose)
            maxClose = e.close;

    cout << "\n--- Closing Price Line Graph ---\n";
    for (auto& e : entries) {
        int barLength = static_cast<int>((e.close / maxClose) * 50);
        cout << e.date << ": ";
        for (int i = 0; i < barLength; ++i)
            cout << "|";
        cout << " $" << fixed << setprecision(2) << e.close << endl;
    }
}

void StockAnalyzer::searchByDate(string targetDate) {
    if (entries.empty()) {
        cout << "No data loaded.\n";
        return;
    }
    bool found = false;
    for (auto& e : entries) {
        if (trim(e.date) == trim(targetDate)) {
            cout << "Closing price on " << e.date << ": $" << fixed << setprecision(2) << e.close << endl;
            found = true;
            break;
        }
    }
    if (!found) {
        cout << "Date not found." << endl;
    }
}
```

StockAnalyzer.cpp

- Implements the core functionality of the StockAnalyzer class
- Loads stock data from a .csv file using ifstream and stores it in a vector
- Cleans and parses each line using getline(), substr(), and stod()
- Calculates the average closing price by looping through the vector
- Displays an ASCII line graph showing the relative closing prices
- Performs a linear search to find and display the closing price by a specific date
- Each function is modular and handles one task (loading, graphing, etc.)
- Includes printed feedback to help users know if data was loaded or if errors occurred

StockAnalyzer.h file

BlackRock Stock Analyzer App
main

BlackRock Stock Analyzer App 〉 My Mac Finished running BlackRock Stock Analyzer App

BLK | C+ StockAnalyzer | C+ main | h StockAnalyzer | BlackRock Stock Analyzer App

BlackRock Stock Analyzer App 〉 BlackRock Stock Analyzer App 〉 h StockAnalyzer 〉 No Selection

```
// StockAnalyzer.h
#ifndef STOCK_ANALYZER_H
#define STOCK_ANALYZER_H

#include <iostream>
#include <vector>
#include <string>
#include <fstream>
using namespace std;

// Represents one record of stock data with date and closing price
struct StockEntry {
    string date;      // Date in YYYY-MM-DD format
    double close;     // Closing price of the stock on that date
};

// Handles loading, analyzing, and displaying stock market data
class StockAnalyzer {
private:
    vector<StockEntry> entries;  // Stores all loaded stock entries

public:
    // Loads stock data from a CSV file
    void loadData(string filename);

    // Prints the average closing price across all entries
    void printAverageClosingPrice();

    // Prints a scaled terminal line graph of closing prices
    void printClosingPriceGraph();

    // Searches and prints the closing price for a given date
    void searchByDate(string targetDate);
};

#endif // STOCK_ANALYZER_H
```

StockAnalyzer.h

- Defines the structure and interface for the stock analysis program
- Contains a struct StockEntry with string date and double close fields
- Declares the StockAnalyzer class and its public methods:
  - loadData(), printAverageClosingPrice(), printClosingPriceGraph(), searchByDate()
- Stores all loaded stock data in a private vector<StockEntry>
- Acts as a blueprint, keeping the class definition clean and separate from the logic
- Helps organize the code for better readability and reusability

Program Flow Chart

Start → Show Menu

↓

[1] Load Data → read CSV file → back to menu

[2] Avg Price → calculate & print → back to menu

[3] Line Graph → draw ASCII graph → back to menu

[4] Search → linear search & print → back

[0] Exit → ends program

demo time

Challenges I faced:

- Figuring out how to scale the line graph cleanly was difficult
- Learning to parse CSV data in C++ took some trial and error
- Proud of how smooth the menu and search ended up working

Things I learned:

- How to manage file input
- How to build real apps using class-based structure
- How to comment and document my code clearly

thank you!