Program Assignment 8

Michael Charara

Main.cpp

/\*

Programming Assignment 8

Michael Charara

(Stock Market) Write a program to help a local stock trading company automate its systems. The company invests only in the stock market. At the end of each trading day, the company would like to generate and post the listing of its stocks so that investors can see how their holdings performed that day. We assume that the company invests in, say, 10 different stocks. The desired output is to produce two listings, one sorted by stock symbol and another sorted by percent gain from highest to lowest.

The input data is provided in a file in the following format:

symbol openingPrice closingPrice todayHigh todayLow

prevClose volume

For example, the sample data is:

MSMT 112.50 115.75 116.50 111.75 113.50 6723823

CBA 67.50 75.50 78.75 67.50 65.75 378233

-

-

-

The first line indicates that the stock symbol is MSMT, today’s opening

price was 112.50, the closing price was 115.75, today’s high price was

116.50, today’s low price was 111.75, yesterday’s closing price was

113.50, and the number of shares currently being held is 6723823.

The listing sorted by stock symbols must be of the following form:

The listing sorted by stock symbols must be of the following form:

\*\*\*\*\*\*\*\*\* First Investor's Heaven \*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\* Financial Report \*\*\*\*\*\*\*\*\*\*

Stock           Today                 Previous Percent

Symbol  Open    Close   High    Low     Close   Gain    Volume

------  -----   -----   -----   ----- -------- -------  ------

ABC     123.45  130.95  132.00  125.00  120.50  8.67%   10000

AOLK    80.00   75.00   82.00   74.00   83.00   -9.64%  5000

CSCO    100.00  102.00  105.00  98.00   101.00  0.99%   25000

IBD     68.00   71.00   72.00   67.00   75.00   -5.33%  15000

MSET    120.00  140.00  145.00  140.00  115.00  21.74%  30920

Closing Assets: $9628300.00

-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*

Develop this programming exercise in two steps. In the first step (part a), design and implement a stock object. In the second step (part b), design and implement an object to maintain a list of stocks.

a. (Stock Object) Design and implement the stock object. Call the class that captures the various characteristics of a stock object stockType.

The main components of a stock are the stock symbol, stock price, and number of shares. Moreover, we need to output the opening price, closing price, high price, low price, previous price, and the percent gain/loss for the day. These are also all the characteristics of a stock. Therefore, the stock object should store all this

information. Perform the following operations on each stock object:

    i. Set the stock information.

    ii. Print the stock information.

    iii. Show the different prices.

    iv. Calculate and print the percent gain/loss.

    v. Show the number of shares.

        a.1. The natural ordering of the stock list is by stock symbol. Overload the relational operators to compare two stock objects by their symbols.

        a.2. Overload the insertion operator, <<, for easy output.

        a.3. Because the data is stored in a file, overload the stream extraction operator, >>, for easy input.

        For example, suppose infile is an ifstream object and the input file was opened using the object infile. Further suppose that myStock is a stock object. Then, the statement infile >> myStock; reads the data from the input file and stores it in the object myStock. (Note that this statement reads and stores the data in the relevant components

        of myStock.)

b. Now that you have designed and implemented the class stockType to implement a stock object in a program, it is time to create a list of stock objects.

Let us call the class to implement a list of stock objects stockListType.

The class stockListType must be derived from the class listType, which you designed and implemented in the previous exercise. However,the class stockListType is a very specific class, designed to create a list of stock objects. Therefore, the class stockListType is no longer a template.

Add and/or overwrite the operations of the class listType to implement the necessary operations on a stock list.

The following statement derives the class stockListType from the class listType.

class stockListType: public listType<stockType>

{

    member list

};

The member variables to hold the list elements, the length of the list, and the maximum size of the list were declared as protected in the class listType. Therefore, these members can be directly accessed in the

class stockListType. Because the company also requires you to produce the list ordered by the percent gain/loss, you need to sort the stock list by this component. However, you are not to physically sort the list by the component percent gain/loss. Instead, you will provide a logical ordering with respect to this component.

To do so, add a member variable, an array, to hold the indices of the stock list ordered by the component percent gain/loss. Call this array sortIndicesGainLoss. When printing the list ordered by the component percent gain/loss, use the array sortIndicesGainLoss to print the list. The elements of the array sortIndicesGainLoss will tell which component of the stock list to print next.

c. Write a program that uses these two classes to automate the company’s analysis of stock data.

\*/

// header files

#include <iostream>//include for input-output stream library

#include <fstream>//include for file stream library

#include <iomanip>//include for io manipulation library

#include <vector>//include for vector library

#include<string> //include for string library

#include<algorithm>//include for algorithm library

#include "stockType.h"//header file for stock Type objects declaration

#include "stockList.h" //header file for stcok List objects declaration

using namespace std;

const int a = 5;

int flag =0; // Flag for file checking

void getData(stockList& list) //Function to get data from file and save it to stock type object

{

   ifstream infile;

   string fileName;

   string symbol;

   double OpenPrice;

   double ClosePrice;

   double tHigh;

   double tLow;

   double yClose;

   int       shares;

   stockType temp;

   cout<<"Please enter the stock data file name"<<endl;

   getline(cin,fileName);// Gets file name from user

   infile.open(fileName);// Opens file

   if (!infile) // returns error if file entered does not exist

   {

       cout << "The file you entered does not exist" << endl;

       flag=5;

   }

   else

   {

       infile >> symbol;

       while (infile) // Move data from file to stock inturn stock list

       {

           infile >> OpenPrice >> ClosePrice >> tHigh >> tLow >> yClose >> shares;

           temp.setStockInfo(symbol,OpenPrice,ClosePrice,tHigh,tLow,yClose,shares);

           list.insert(temp);

           infile >> symbol;

       }

   }

};

int main()

{

   stockList stockList; // create stock list object

   cout << fixed << showpoint;

   cout << setprecision(2);

   getData(stockList); // copy data a from file to the stcoklist object

   if(flag==0)

   {

       stockList.sortStockList();// Sort input data

       stockList.printBySymbol();//Print by sort list by symbol

       cout << endl;

       stockList.printByGain();//Print data by sorted list by gain

   }

   return 0;

}

stockType.h

#ifndef H\_stockType

#define H\_stockType

#include <iostream>//include for input-output stream library

#include <fstream>//include for file stream library

#include <iomanip>//include for io manipulation library

#include<string> //include for string library

using namespace std;

class stockType // Declaring Stock Type Class

{

   friend ostream& operator<< (ostream&, const stockType&);

   friend ifstream& operator>> (ifstream&, stockType&);

public:

   void setStockInfo(string symbol, double openPrice,

                      double closeProce, double high,

                      double Low, double prevClose,

                      int   shares);

   string getSymbol();

   double getPercentChange();

   double getOpenPrice();

   double getClosePrice();

   double getHighPrice();

   double getLowPrice();

   double getPrevPrice();

   int getnoOfShares();

   stockType();

   stockType(string symbol, double openPrice,

              double closeProce, double high,

              double Low, double prevClose,

              int   shares);

   bool operator ==(const stockType& other) const;

   bool operator !=(const stockType& other) const;

   bool operator <=(const stockType& other) const;

   bool operator >=(const stockType& other) const;

   bool operator >(const stockType& other) const;

   bool operator <(const stockType& other) const;

private:

   string   stockSymbol;

   double   todayOpenPrice;

   double   todayClosePrice;

   double   todayHigh;

   double   todayLow;

   double   yesterdayClose;

   double   percentChange;

   int       noOfShares;

};

void stockType::setStockInfo(string symbol, double openPrice,//Setting Stock Info Method

                      double closeProce, double high,

                      double low, double prevClose,

                      int   shares)

{

       stockSymbol = symbol;

       todayOpenPrice = openPrice;

       todayClosePrice = closeProce;

       todayHigh = high;

       todayLow = low;

       yesterdayClose = prevClose;

       percentChange = (todayClosePrice - yesterdayClose) / yesterdayClose;

       noOfShares = shares;

}

string stockType::getSymbol() // Method for reterieving symbol

{

   return stockSymbol;

}

double stockType::getPercentChange()// Method for reterieving percenatge change

{

   return percentChange;

}

double stockType::getOpenPrice()// Method for reterieving open price

{

   return todayOpenPrice;

}

double stockType::getClosePrice()// Method for reterieving close price

{

   return todayClosePrice;

}

double stockType::getHighPrice()// Method for reterieving high Price

{

   return todayHigh;

}

double stockType::getLowPrice()// Method for reterieving low price

{

   return todayLow;

}

double stockType::getPrevPrice()// Method for reterieving prev price

{

   return yesterdayClose;

}

int stockType::getnoOfShares()// Method for reterieving no of shares

{

   return noOfShares;

}

stockType::stockType()// Method for setting an empty object for soritng purposes

{

    stockSymbol = "";

    todayOpenPrice = 0;

    todayClosePrice = 0;

    todayHigh = 0;

    todayLow = 0;

    yesterdayClose = 0;

    percentChange = 0;

    noOfShares = 0;

}

stockType::stockType(string symbol, double openPrice,// Method for writng data in to the object

              double closeProce, double high,

              double low, double prevClose,

              int   shares)

{

   stockSymbol = symbol;

   todayOpenPrice = openPrice;

   todayClosePrice = closeProce;

   todayHigh = high;

   todayLow = low;

   yesterdayClose = prevClose;

   percentChange = (todayClosePrice - yesterdayClose) / yesterdayClose; // Percentage change calculation

   noOfShares = shares;

}

bool stockType::operator ==(const stockType& other) const // Natural Logical Sorting Method for data sorting by Symbol

{

   return (stockSymbol == other.stockSymbol);

}

bool stockType::operator !=(const stockType& other) const

{

   return (stockSymbol != other.stockSymbol);

}

bool stockType::operator <=(const stockType& other) const

{

   return (stockSymbol <= other.stockSymbol);

}

bool stockType::operator >=(const stockType& other) const

{

   return (stockSymbol >= other.stockSymbol);

}

bool stockType::operator >(const stockType& other) const

{

   return (stockSymbol > other.stockSymbol);

}

bool stockType::operator <(const stockType& other) const

{

   return (stockSymbol < other.stockSymbol);

}

ostream& operator<<(ostream& os, const stockType& stock) // Out stream simplification

{

   os << setw(6) << stock.stockSymbol << " "

        << setw(6) << stock.todayOpenPrice << " "

       << setw(6) << stock.todayClosePrice << " "

       << setw(6) << stock.todayHigh << " "

       << setw(6) << stock.todayLow << " "

       << setw(6) << stock.yesterdayClose << " "

       << setw(8) << stock.percentChange \* 100 << "% "

       << setw(10) << stock.noOfShares;

   return os;

}

ifstream& operator>> (ifstream& inf, stockType& stock)// in stream simplification

{

   inf >> stock.stockSymbol >> stock.todayOpenPrice

       >> stock.todayClosePrice >> stock.todayHigh

       >> stock.todayLow >> stock.yesterdayClose

       >> stock.noOfShares;

   stock.percentChange = (stock.todayClosePrice - stock.yesterdayClose) / stock.yesterdayClose;

   return inf;

}

#endif

stockList.h

#ifndef H\_stockList

#define H\_stockList

#include <vector>//include for vector library

#include<string> //include for string library

#include <iostream>//include for input-output stream library

#include<algorithm>//include for algorithm library

#include "stockType.h"// Include to use stock type object

using namespace std;

class stockList // StockList class declaration

{

public:

   void printBySymbol();

   void printByGain();

   void printReports();

   void sortStockList();

   void sortByGain();

   void insert(const stockType& item);

private:

   vector<int>   indexByGain;

   vector<stockType> list; // Stock list creation using stock type object

};

void stockList::printBySymbol() // Printing Stock List by Symbol

{

    double closingBalance = 0;

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FIRST INVESTORS HEAVEN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

    cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FINANCIAL REPORT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

    cout << "\nSTOCK           TODAY                    PREVIOUS PERCENT" << endl;

    cout << "SYMBOL OPEN    CLOSE   HIGH    LOW     CLOSE     GAIN          VOLUME" << endl;

    cout << "------ -----   -----   -----   -----   -------- -------     ------" << endl;

    int x=list.size();

    int y=0;

    while (y<x)

    {

        cout << list[y] << endl;

        closingBalance += list[y].getClosePrice() \* list[y].getnoOfShares();

        y++;

    }

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

    cout << "CLOSING ASSETS: $" << closingBalance << endl;

    cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

void stockList::printByGain() // Printing Stock List by Gain

{

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    FIRST INVESTORS HEAVEN    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

    cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    FINANCIAL REPORT    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

    cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*       GAIN/LOSS       \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

    cout << "\nSTOCK           TODAY                    PREVIOUS PERCENT" << endl;

    cout << "SYMBOL OPEN    CLOSE   HIGH    LOW     CLOSE     GAIN          VOLUME" << endl;

    cout << "------ -----   -----   -----   -----   -------- -------     ------" << endl;

    int x=list.size();

        int y=0;

        while (y<x)

        {

        cout << list[indexByGain[y]]<<"\n";

        y++;

        }

        cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

void stockList::sortStockList() // Sorting stock list by Symbol

{

    sort(list.begin(), list.end());

    sortByGain();

}

void stockList::insert(const stockType& item)// ethod for inserting stock type object in stock list

{

    list.push\_back(item);

}

void stockList::sortByGain() // Sorting stock List By Gain

{

    bool \*temp;

    temp = new bool[list.size()];

    int maxIndex;

    int l=list.size();

    int y=0;

    while(y<l)

    {

        temp[y] = false;

        y++;

    }

    l=list.size();

    int r=0;

    while(r<l)

    {

        l=list.size();

        int j=0;

        while(j<l)

        {

            if (!temp[j])

            {

                maxIndex = j;

                break;

            }

            j++;

        }

        l=list.size();

        j=maxIndex+1;

        while(j<l)

        {

            if (!temp[j])

            {

                if (list[maxIndex].getPercentChange() < list[j].getPercentChange())

                    maxIndex = j;

            }

            j++;

        }

        indexByGain.push\_back(maxIndex);

        temp[maxIndex] = true;

        r++;

    }

    delete [] temp;

}

#endif

Data.txt

ABC 123.45 130.95 132.00 125.00 120.50

AOLK 80.00 75.00 82.00 74.00 83.00

CSCO 100.00 102.00 105.00 98.00 101.00

IBD 68.00 71.00 72.00 67.00 75.00

MSET 120.00 140.00 145.00 140.00 115.00