* Makefile

CC=gcc

CFLAGS=-I.

stocks: main.o

$(CC) -o stocks main.o

* C Source Code

//Author: Calob Horton

//Class: CIS 241-02

//Date: July 29, 2021

//Version: 1.1

#include <stdio.h>

#include <stdlib.h>

//structure to hold the values in each record in text file

struct record{

double ratio;

int month, day, year, putVol, callVol, optVol;

};

//then structures for years, months, quarters for comparisons

struct year{

double maxRatio, minRatio, avgRatio, avgCall, avgPut, avgOpt;

int minCall, maxCall, minPut, maxPut, minOpt, maxOpt;

};

struct month{

double maxRatio, minRatio, avgRatio, avgCall, avgPut, avgOpt;

int minCall, maxCall, minPut, maxPut, minOpt, maxOpt;

};

struct quarter{

double maxRatio, minRatio, avgRatio, avgCall, avgPut, avgOpt;

int minCall, maxCall, minPut, maxPut, minOpt, maxOpt;

};

//array of record structures to hold values from file

struct record records[2330];

//arrays of year, month, quarter to hold respective values

struct year years[10];

struct month months[12];

struct quarter quarters[4];

//list of function prototypes to be used later

void avgRatio(int choice);

void minRatio(int choice);

void maxRatio(int choice);

void avgCall(int choice);

void minCall(int choice);

void maxCall(int choice);

void avgPut(int choice);

void minPut(int choice);

void maxPut(int choice);

void avgOpt(int choice);

void minOpt(int choice);

void maxOpt(int choice);

void displayYears();

void displayMonths();

void displayQuarters();

void displayConclusion();

int main(void){

FILE \*f;

char junk[100];//, user;

int i = 0, user;

if ((f = fopen("SPY241Project.txt", "r")) == NULL){

printf("Oops! File not found! Please try again.");

} else{

//skips first line

fgets(junk, 100, f);

//then loops and adds each line's data to appropriate spot

for (; i < 2330; ++i){

fscanf(f, "%d/%d/%d,%lf,%d,%d,%d", &records[i].month,

&records[i].day, &records[i].year, &records[i].ratio,

&records[i].putVol, &records[i].callVol,

&records[i].optVol);

}

}

//loops for void functions

for (int i = 0; i < 3; i++){

avgRatio(i), minRatio(i), maxRatio(i),avgCall(i), minCall(i),

maxCall(i), avgPut(i), minPut(i), maxPut(i), avgOpt(i),

minOpt(i), maxOpt(i);

}

//prompts for user input to keep console clear

printf("Enter 1 for Years, 2 for Months, 3 for Quarters, 4 for "

"Conclusions or 0 to Quit: ");

//loops while user != 0 to continue printing to console

while (scanf("%d", &user) != 0) {

system("clear");

if (user == 1) {

displayYears();

printf("Please enter next selection: ");

} else if (user == 2) {

displayMonths();

printf("Please enter next selection: ");

} else if (user == 3) {

displayQuarters();

printf("Please enter next selection: ");

} else if (user == 4){

displayConclusion();

printf("Please enter next selection: ");

} else if (user == 0) {

break;

} else {

printf("Oops! Try 1 for Years, 2 for Months, 3 for Quarters,"

"4 for Conclusions, or 0 to quit!");

}

}

fclose(f);

}

void avgRatio(int choice){

double totRatio = 0, q1 = 0, q2 = 0, q3 = 0, q4 = 0;

int k = 0, a = 0, b = 0, c = 0, d = 0;

//switch statement for which average ratio to find

switch (choice){

//years

case 0:

for (int i = 0; i < 10; i++){

for (int j = 0; j < 2330; j++){

if (records[j].year == (i + 10)){

totRatio += records[j].ratio;

k++;

}

}

//calculates and assigns year[i]'s average ratio

totRatio /= k;

years[i].avgRatio = totRatio;

totRatio = 0, k = 0;

}

break;

//months

case 1:

for (int i = 0; i < 12; i++){

for (int j = 0; j < 2330; j++){

if (records[j].month == (i + 1)){

totRatio += records[j].ratio;

k++;

}

}

//calculates and assigns month[i]'s average ratio

totRatio /= k;

months[i].avgRatio = totRatio;

totRatio = 0, k = 0;

}

break;

//quarters

case 2:

for (int j = 0; j < 2330; j++){

if (records[j].month >= 1 && records[j].month <= 3){

q1 += records[j].ratio;

a++;

} else if (records[j].month >= 4 && records[j].month <= 6){

q2 += records[j].ratio;

b++;

} else if (records[j].month >= 7 && records[j].month <= 9){

q3 += records[j].ratio;

c++;

} else {

q4 += records[j].ratio;

d++;

}

}

//calculates and adds average to proper quarter

quarters[0].avgRatio = q1 / a;

quarters[1].avgRatio = q2 / b;

quarters[2].avgRatio = q3 / c;

quarters[3].avgRatio = q4 / d;

break;

default:

break;

}

}

void minRatio(int choice){

double minimum = records[0].ratio, minq1 = records[0].ratio,

minq2 = records[0].ratio, minq3 = records[0].ratio,

minq4 = records[0].ratio;

//switch for which minRatio to find

switch (choice) {

//years

case 0:

for (int i = 0; i < 10; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].year == (i + 10)) {

if (records[j].ratio < minimum) {

minimum = records[j].ratio;

}

}

}

//adds minimum to minRatio for year[i]

years[i].minRatio = minimum;

minimum = records[0].ratio;

}

break;

//months

case 1:

for (int i = 0; i < 12; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].month == (i + 1)) {

if (records[j].ratio < minimum) {

minimum = records[j].ratio;

}

}

}

//adds minimum to minRatio for month[i]

months[i].minRatio = minimum;

minimum = records[0].ratio;

}

break;

//quarters

case 2:

for (int j = 0; j < 2330; j++){

if (records[j].month >= 1 && records[j].month <= 3){

if (records[j].ratio < minq1){

minq1 = records[j].ratio;

}

} else if (records[j].month >= 4 && records[j].month <= 6){

if (records[j].ratio < minq2){

minq2 = records[j].ratio;

}

} else if (records[j].month >= 7 && records[j].month <= 9){

if (records[j].ratio < minq3){

minq3 = records[j].ratio;

}

} else {

if (records[j].ratio < minq4){

minq4 = records[j].ratio;

}

}

}

//adds minimum to minRatio for proper quarter

quarters[0].minRatio = minq1;

quarters[1].minRatio = minq2;

quarters[2].minRatio = minq3;

quarters[3].minRatio = minq4;

break;

default:

break;

}

}

void maxRatio(int choice){

double maximum = records[0].ratio, maxq1 = records[0].ratio,

maxq2 = records[0].ratio, maxq3 = records[0].ratio,

maxq4 = records[0].ratio;

//switch statement for which maxRatio to calculate

switch (choice) {

//years

case 0:

for (int i = 0; i < 10; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].year == (i + 10)) {

if (records[j].ratio > maximum) {

maximum = records[j].ratio;

}

}

}

//adds maxRatio to years[i]

years[i].maxRatio = maximum;

maximum = records[0].ratio;

}

break;

//months

case 1:

for (int i = 0; i < 12; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].month == (i + 1)) {

if (records[j].ratio > maximum) {

maximum = records[j].ratio;

}

}

}

//adds maxRatio to months[i]

months[i].maxRatio = maximum;

maximum = records[0].ratio;

}

break;

//quarters

case 2:

for (int j = 0; j < 2330; j++){

if (records[j].month >= 1 && records[j].month <= 3){

if (records[j].ratio > maxq1){

maxq1 = records[j].ratio;

}

} else if (records[j].month >= 4 && records[j].month <= 6){

if (records[j].ratio > maxq2){

maxq2 = records[j].ratio;

}

} else if (records[j].month >= 7 && records[j].month <= 9){

if (records[j].ratio > maxq3){

maxq3 = records[j].ratio;

}

} else {

if (records[j].ratio > maxq4){

maxq4 = records[j].ratio;

}

}

}

//adds maxRatio to proper quarter

quarters[0].maxRatio = maxq1;

quarters[1].maxRatio = maxq2;

quarters[2].maxRatio = maxq3;

quarters[3].maxRatio = maxq4;

break;

default:

break;

}

}

void avgCall(int choice){

double totCall = 0, q1 = 0, q2 = 0, q3 = 0, q4 = 0;

int k = 0, a = 0, b = 0, c = 0, d = 0;

//switch statement for which avgCall to calculate

switch (choice){

//years

case 0:

for (int i = 0; i < 10; i++){

for (int j = 0; j < 2330; j++){

if (records[j].year == (i + 10)){

totCall += records[j].callVol;

k++;

}

}

//calculates and adds average to year[i]

totCall /= k;

years[i].avgCall = totCall;

totCall = 0, k = 0;

}

break;

//months

case 1:

for (int i = 0; i < 12; i++){

for (int j = 0; j < 2330; j++){

if (records[j].month == (i + 1)){

totCall += records[j].callVol;

k++;

}

}

//calculates and adds average to month[i]

totCall /= k;

months[i].avgCall = totCall;

totCall = 0, k = 0;

}

break;

//quarters

case 2:

for (int j = 0; j < 2330; j++){

if (records[j].month >= 1 && records[j].month <= 3){

q1 += records[j].callVol;

a++;

} else if (records[j].month >= 4 && records[j].month <= 6){

q2 += records[j].callVol;

b++;

} else if (records[j].month >= 7 && records[j].month <= 9){

q3 += records[j].callVol;

c++;

} else {

q4 += records[j].callVol;

d++;

}

}

//calculates and adds avgCall to proper quarter

quarters[0].avgCall = q1 / a;

quarters[1].avgCall = q2 / b;

quarters[2].avgCall = q3 / c;

quarters[3].avgCall = q4 / d;

break;

default:

break;

}

}

void minCall(int choice){

int minimum = records[0].callVol, minq1 = records[0].callVol,

minq2 = records[0].callVol, minq3 = records[0].callVol,

minq4 = records[0].callVol;

//switch statement for which minCall to calculate

switch (choice) {

//years

case 0:

for (int i = 0; i < 10; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].year == (i + 10)) {

if (records[j].callVol < minimum) {

minimum = records[j].callVol;

}

}

}

//adds minimum to year[i]'s minCall

years[i].minCall = minimum;

minimum = records[0].callVol;

}

break;

//months

case 1:

for (int i = 0; i < 12; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].month == (i + 1)) {

if (records[j].callVol < minimum) {

minimum = records[j].callVol;

}

}

}

//adds minimum to month[i]'s minCall

months[i].minCall = minimum;

minimum = records[0].callVol;

}

break;

//quarters

case 2:

for (int j = 0; j < 2330; j++){

if (records[j].month >= 1 && records[j].month <= 3){

if (records[j].callVol < minq1){

minq1 = records[j].callVol;

}

} else if (records[j].month >= 4 && records[j].month <= 6){

if (records[j].callVol < minq2){

minq2 = records[j].callVol;

}

} else if (records[j].month >= 7 && records[j].month <= 9){

if (records[j].callVol < minq3){

minq3 = records[j].callVol;

}

} else {

if (records[j].callVol < minq4){

minq4 = records[j].callVol;

}

}

}

//adds minimum to proper quarter

quarters[0].minCall = minq1;

quarters[1].minCall = minq2;

quarters[2].minCall = minq3;

quarters[3].minCall = minq4;

break;

default:

break;

}

}

void maxCall(int choice){

int maximum = records[0].callVol, maxq1 = records[0].callVol,

maxq2 = records[0].callVol, maxq3 = records[0].callVol,

maxq4 = records[0].callVol;

//switch statement for which maxCall to calculate

switch (choice) {

//years

case 0:

for (int i = 0; i < 10; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].year == (i + 10)) {

if (records[j].callVol > maximum) {

maximum = records[j].callVol;

}

}

}

//adds maximum to year[i]'s maxCall

years[i].maxCall = maximum;

maximum = records[0].callVol;

}

break;

//months

case 1:

for (int i = 0; i < 12; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].month == (i + 1)) {

if (records[j].callVol > maximum) {

maximum = records[j].callVol;

}

}

}

//adds maximum to month[i]'s maxCall

months[i].maxCall = maximum;

maximum = records[0].callVol;

}

break;

//quarters

case 2:

for (int j = 0; j < 2330; j++){

if (records[j].month >= 1 && records[j].month <= 3){

if (records[j].callVol > maxq1){

maxq1 = records[j].callVol;

}

} else if (records[j].month >= 4 && records[j].month <= 6){

if (records[j].callVol > maxq2){

maxq2 = records[j].callVol;

}

} else if (records[j].month >= 7 && records[j].month <= 9){

if (records[j].callVol > maxq3){

maxq3 = records[j].callVol;

}

} else {

if (records[j].callVol > maxq4){

maxq4 = records[j].callVol;

}

}

}

//adds maximum to proper quarter maxCall

quarters[0].maxCall = maxq1;

quarters[1].maxCall = maxq2;

quarters[2].maxCall = maxq3;

quarters[3].maxCall = maxq4;

break;

default:

break;

}

}

void avgPut(int choice){

double totPut = 0, q1 = 0, q2 = 0, q3 = 0, q4 = 0;

int k = 0, a = 0, b = 0, c = 0, d = 0;

//switch statement for which avgPut to calculate

switch (choice){

//years

case 0:

for (int i = 0; i < 10; i++){

for (int j = 0; j < 2330; j++){

if (records[j].year == (i + 10)){

totPut += records[j].putVol;

k++;

}

}

//calculates and adds avgPut to proper year[i]'s avgPut

totPut /= k;

years[i].avgPut = totPut;

totPut = 0, k = 0;

}

break;

//months

case 1:

for (int i = 0; i < 12; i++){

for (int j = 0; j < 2330; j++){

if (records[j].month == (i + 1)){

totPut += records[j].putVol;

k++;

}

}

//calculates and adds avgPut to proper month[i] avgPut

totPut /= k;

months[i].avgPut = totPut;

totPut = 0, k = 0;

}

break;

//quarters

case 2:

for (int j = 0; j < 2330; j++){

if (records[j].month >= 1 && records[j].month <= 3){

q1 += records[j].putVol;

a++;

} else if (records[j].month >= 4 && records[j].month <= 6){

q2 += records[j].putVol;

b++;

} else if (records[j].month >= 7 && records[j].month <= 9){

q3 += records[j].putVol;

c++;

} else {

q4 += records[j].putVol;

d++;

}

}

//calculates and adds avgPut to proper quarter

quarters[0].avgPut = q1 / a;

quarters[1].avgPut = q2 / b;

quarters[2].avgPut = q3 / c;

quarters[3].avgPut = q4 / d;

break;

default:

break;

}

}

void minPut(int choice){

int minimum = records[0].putVol, minq1 = records[0].putVol,

minq2 = records[0].putVol, minq3 = records[0].putVol,

minq4 = records[0].putVol;

//switch statement for which minPut to calculate

switch (choice) {

//years

case 0:

for (int i = 0; i < 10; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].year == (i + 10)) {

if (records[j].putVol < minimum) {

minimum = records[j].putVol;

}

}

}

//adds minimum to proper year[i]'s minPut

years[i].minPut = minimum;

minimum = records[0].putVol;

}

break;

//months

case 1:

for (int i = 0; i < 12; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].month == (i + 1)) {

if (records[j].putVol < minimum) {

minimum = records[j].putVol;

}

}

}

//adds minimum to proper month[i]'s minPut

months[i].minPut = minimum;

minimum = records[0].putVol;

}

break;

//quarters

case 2:

for (int j = 0; j < 2330; j++){

if (records[j].month >= 1 && records[j].month <= 3){

if (records[j].putVol < minq1){

minq1 = records[j].putVol;

}

} else if (records[j].month >= 4 && records[j].month <= 6){

if (records[j].putVol < minq2){

minq2 = records[j].putVol;

}

} else if (records[j].month >= 7 && records[j].month <= 9){

if (records[j].putVol < minq3){

minq3 = records[j].putVol;

}

} else {

if (records[j].putVol < minq4){

minq4 = records[j].putVol;

}

}

}

//adds minimum to proper quarter

quarters[0].minPut = minq1;

quarters[1].minPut = minq2;

quarters[2].minPut = minq3;

quarters[3].minPut = minq4;

break;

default:

break;

}

}

void maxPut(int choice){

int maximum = records[0].putVol, maxq1 = records[0].putVol,

maxq2 = records[0].putVol, maxq3 = records[0].putVol,

maxq4 = records[0].putVol;

//switch statement for which maxPut to find

switch (choice) {

//years

case 0:

for (int i = 0; i < 10; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].year == (i + 10)) {

if (records[j].putVol > maximum) {

maximum = records[j].putVol;

}

}

}

//adds maximum to year[i]'s maxPut

years[i].maxPut = maximum;

maximum = records[0].putVol;

}

break;

//months

case 1:

for (int i = 0; i < 12; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].month == (i + 1)) {

if (records[j].putVol > maximum) {

maximum = records[j].putVol;

}

}

}

//adds maximum to month[i]'s maxPut

months[i].maxPut = maximum;

maximum = records[0].putVol;

}

break;

//quarters

case 2:

for (int j = 0; j < 2330; j++){

if (records[j].month >= 1 && records[j].month <= 3){

if (records[j].putVol > maxq1){

maxq1 = records[j].putVol;

}

} else if (records[j].month >= 4 && records[j].month <= 6){

if (records[j].putVol > maxq2){

maxq2 = records[j].putVol;

}

} else if (records[j].month >= 7 && records[j].month <= 9){

if (records[j].putVol > maxq3){

maxq3 = records[j].putVol;

}

} else {

if (records[j].putVol > maxq4){

maxq4 = records[j].putVol;

}

}

}

//adds maximum to proper quarter's maxPut

quarters[0].maxPut = maxq1;

quarters[1].maxPut = maxq2;

quarters[2].maxPut = maxq3;

quarters[3].maxPut = maxq4;

break;

default:

break;

}

}

void avgOpt(int choice){

double totOpt = 0, q1 = 0, q2 = 0, q3 = 0, q4 = 0;

int k = 0, a = 0, b = 0, c = 0, d = 0;

//switch statement for which avgOpt to find

switch (choice){

//years

case 0:

for (int i = 0; i < 10; i++){

for (int j = 0; j < 2330; j++){

if (records[j].year == (i + 10)){

totOpt += records[j].optVol;

k++;

}

}

//calculates and adds average to year[i]'s avgOpt

totOpt /= k;

years[i].avgOpt = totOpt;

totOpt = 0, k = 0;

}

break;

//months

case 1:

for (int i = 0; i < 12; i++){

for (int j = 0; j < 2330; j++){

if (records[j].month == (i + 1)){

totOpt += records[j].optVol;

k++;

}

}

//calculates and adds average to month[i]'s avgOpt

totOpt /= k;

months[i].avgOpt = totOpt;

totOpt = 0, k = 0;

}

break;

//quarters

case 2:

for (int j = 0; j < 2330; j++){

if (records[j].month >= 1 && records[j].month <= 3){

q1 += records[j].optVol;

a++;

} else if (records[j].month >= 4 && records[j].month <= 6){

q2 += records[j].optVol;

b++;

} else if (records[j].month >= 7 && records[j].month <= 9){

q3 += records[j].optVol;

c++;

} else {

q4 += records[j].optVol;

d++;

}

}

//calculates and adds average to proper quarter's avgOpt

quarters[0].avgOpt = q1 / a;

quarters[1].avgOpt = q2 / b;

quarters[2].avgOpt = q3 / c;

quarters[3].avgOpt = q4 / d;

break;

default:

break;

}

}

void minOpt(int choice){

int minimum = records[0].optVol, minq1 = records[0].optVol,

minq2 = records[0].optVol, minq3 = records[0].optVol,

minq4 = records[0].optVol;

//switch statement for which minOpt to find

switch (choice) {

//years

case 0:

for (int i = 0; i < 10; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].year == (i + 10)) {

if (records[j].optVol < minimum) {

minimum = records[j].optVol;

}

}

}

//adds minimum to year[i]'s minOpt

years[i].minOpt = minimum;

minimum = records[0].optVol;

}

break;

//months

case 1:

for (int i = 0; i < 12; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].month == (i + 1)) {

if (records[j].optVol < minimum) {

minimum = records[j].optVol;

}

}

}

//adds minimum to month[i]'s minOpt

months[i].minOpt = minimum;

minimum = records[0].optVol;

}

break;

//quarters

case 2:

for (int j = 0; j < 2330; j++){

if (records[j].month >= 1 && records[j].month <= 3){

if (records[j].optVol < minq1){

minq1 = records[j].optVol;

}

} else if (records[j].month >= 4 && records[j].month <= 6){

if (records[j].optVol < minq2){

minq2 = records[j].optVol;

}

} else if (records[j].month >= 7 && records[j].month <= 9){

if (records[j].optVol < minq3){

minq3 = records[j].optVol;

}

} else {

if (records[j].optVol < minq4){

minq4 = records[j].optVol;

}

}

}

//adds minimum to proper quarter's minOpt

quarters[0].minOpt = minq1;

quarters[1].minOpt = minq2;

quarters[2].minOpt = minq3;

quarters[3].minOpt = minq4;

break;

default:

break;

}

}

void maxOpt(int choice){

int maximum = records[0].optVol, maxq1 = records[0].optVol,

maxq2 = records[0].optVol, maxq3 = records[0].optVol,

maxq4 = records[0].optVol;

//switch statement for which maxOpt to find

switch (choice) {

//years

case 0:

for (int i = 0; i < 10; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].year == (i + 10)) {

if (records[j].optVol > maximum) {

maximum = records[j].optVol;

}

}

}

//adds maximum to year[i]'s maxOpt

years[i].maxOpt = maximum;

maximum = records[0].optVol;

}

break;

//months

case 1:

for (int i = 0; i < 12; i++) {

for (int j = 1; j < 2330; ++j) {

if (records[j].month == (i + 1)) {

if (records[j].optVol > maximum) {

maximum = records[j].optVol;

}

}

}

//adds maximum to month[i]'s maxOpt

months[i].maxOpt = maximum;

maximum = records[0].optVol;

}

break;

//quarters

case 2:

for (int j = 0; j < 2330; j++){

if (records[j].month >= 1 && records[j].month <= 3){

if (records[j].optVol > maxq1){

maxq1 = records[j].optVol;

}

} else if (records[j].month >= 4 && records[j].month <= 6){

if (records[j].optVol > maxq2){

maxq2 = records[j].optVol;

}

} else if (records[j].month >= 7 && records[j].month <= 9){

if (records[j].optVol > maxq3){

maxq3 = records[j].optVol;

}

} else {

if (records[j].optVol > maxq4){

maxq4 = records[j].optVol;

}

}

}

//adds maximum to proper quarter's maxOpt

quarters[0].maxOpt = maxq1;

quarters[1].maxOpt = maxq2;

quarters[2].maxOpt = maxq3;

quarters[3].maxOpt = maxq4;

break;

default:

break;

}

}

void displayYears(){

printf("Year: Average, Minimum, Maximum Ratios || "

"Average, Minimum, Maximum Put || Average,"

" Minimum, Maximum Call || Average, Minimum,"

" Maximum Options\n"

"===============================================||============="

"================================||============================"

"==================||=========================================="

"======\n");

for (int i = 0; i < 10; i++){

printf("20%d:%12.2lf%11.2lf%11.2lf%23.2lf%14d%13d%20.2lf%14d%13d%"

"21.2lf%14d%13d\n", (i + 10), years[i].avgRatio,

years[i].minRatio, years[i].maxRatio, years[i].avgPut,

years[i].minPut, years[i].maxPut, years[i].avgCall,

years[i].minCall, years[i].maxCall, years[i].avgOpt,

years[i].minOpt, years[i].maxOpt);

}

}

void displayMonths(){

printf("Month: Average, Minimum, Maximum Ratios || "

"Average, Minimum, Maximum Put || Average,"

" Minimum, Maximum Call || Average, Minimum,"

" Maximum Options\n"

"================================================||============="

"================================||==========================="

"===================||========================================="

"=======\n");

for (int i = 0; i < 12; i++){

printf("%2d:%15.2lf%11.2lf%11.2lf%23.2lf%14d%13d%20.2lf%14d%13d%"

"21.2lf%14d%13d\n", (i + 1), months[i].avgRatio,

months[i].minRatio, months[i].maxRatio, months[i].avgPut,

months[i].minPut, months[i].maxPut, months[i].avgCall,

months[i].minCall, months[i].maxCall, months[i].avgOpt,

months[i].minOpt, months[i].maxOpt);

}

}

void displayQuarters(){

printf("Quarter: Average, Minimum, Maximum Ratios || "

"Average, Minimum, Maximum Put || Average,"

" Minimum, Maximum Call || Average, Minimum,"

" Maximum Options\n"

"==================================================||============="

"================================||==========================="

"===================||========================================="

"=======\n");

for (int i = 0; i < 4; i++){

printf("Q%d:%17.2lf%11.2lf%11.2lf%23.2lf%14d%13d%20.2lf%14d%13d%"

"21.2lf%14d%13d\n", (i + 1), quarters[i].avgRatio,

quarters[i].minRatio, quarters[i].maxRatio, quarters[i].avgPut,

quarters[i].minPut, quarters[i].maxPut, quarters[i].avgCall,

quarters[i].minCall, quarters[i].maxCall, quarters[i].avgOpt,

quarters[i].minOpt, quarters[i].maxOpt);

}

}

void displayConclusion(){

printf("Investopedia defines bearish sentiment as a time when investors are\n"

"buying more puts than calls. Bullish sentiment is when the opposite\n"

"is true: when investors buy more calls than puts. A put/call ratio\n"

"greater than 0.7 and exceeding 1.0 \"suggests that bearish sentiment\n"

"is building in the market...\" and \"investors are speculating that\n"

"the market will move lower.\" A ratio that is between (or below)\n"

"0.5 to 0.7 \"is considered a bullish indicator.\"\n\nFrom the data we "

"can determine that no years, quarters, nor months\nwere considered "

"bullish under Investopedia's definitions. In fact,\nthe lowest daily "

"ratio of 0.81 still does not fall under the given\ndefinition of bullish. "

"They are actually all bearish in what the\nwebsite describes as a "

"\"extreme\" way because of how much higher\nthe average ratio is "

"compared to the 0.5 to 0.7 range of typical\nbullish ratios. There are "

"a few daily ratio outliers but\nfor the most part they stay "

"relatively stable.\n\nI also found it interesting to see how 2010-2013 "

"were years\nwith relatively low options volumes. This shows a trend "

"of\nwariness and apprehensiveness after the 2008 Recession. The data "

"also\nshows that after 2013 investors were more willing to dabble "

"in\nthe stock market again, even if the put/call ratio shows that "

"the\nmajority of them were buying puts instead of calls.\n\nFinally, "

"there seem to also be times of the year where the\nstock market is "

"slower than others. Specifically, March, April,\nMay, and July have "

"lower options volumes than other months.\nThese months make up Spring "

"and the middle of Summer. This\nmakes sense when compared to quarterly "

"averages: Q2 (Spring) and\nthe middle of Q3 (Summer, specifically July) "

"are noticeably lower,\nas well.\n");

}

* Screenshots of Output

A picture containing text

Description automatically generated

Figure - Years table

Text

Description automatically generated with low confidence

Figure - Months table

Table

Description automatically generated with medium confidence

Figure - Quarters table

Text

Description automatically generated

Figure - Conclusions