

/\*

Edward Alvarado

CIS247C

ATM application

03/03/2021

\*/

// bring in our libararies

#include <iostream>

#include <conio.h>

#include <string>

#include <fstream> // read/write to files

#include <ctime> // time(0)

#include <iomanip> //setprecision( )

using namespace std;

// prototypes

void deposit(double\* ptrBalance);

void withdrawal(double\* ptrBalance, float dailyLimit); // overloaded method - this version does not take withdrawal amount

void withdrawal(double\* ptrBalance, float dailyLimit, float amount); // overloaded method that takes withdrawal amount

/// Entry point to the application

int main()

{

//create constant values -- cannot be changed

const int EXIT\_VALUE = 5;

const float DAILY\_LIMIT = 400.0f;

const string FILENAME = "Account.txt";

// create balance variable

double balance = 0.0;

// look for the starting balance; otherwise generage a random starting balance

ifstream iFile(FILENAME.c\_str());

if (iFile.is\_open())

{

// did the file open? if so, read the balance

iFile >> balance;

iFile.close();

}

else

{

// if the file did not open or does not exist, create a

// random number for the starting balance

srand(time(0));

const int MIN = 1000;

const int MAX = 10000;

balance = rand() % (MAX - MIN + 1) + MIN;

}

cout << fixed << setprecision(2) << "Starting Balance: $" << balance << endl;

// let's create a pointer and set it to the balance variable location

double\* ptrBalance = &balance; // & means "address of"

//pause before we clear the screen

cout << "\nPress any key to continue...";

\_getch();

//create loop variable BEFORE the loop

short choice = 0;

// start the application loop

do

{

//show the menu

system("cls"); // clears the console screen -- for MAC, use system("clear");

cout << "Menu\n" << endl;

cout << "1) Deposit " << endl;

cout << "2) Withdrawl " << endl;

cout << "3) Check Balance " << endl;

cout << "4) Quick $40" << endl;

cout << "5) Exit " << endl;

//get user input

cout << "\nEnter your choice: ";

cin >> choice;

//run code based on the user's choice

switch (choice)

{

case 1:

deposit(ptrBalance); // passing a pointer so only four bytes have to go accross the system bus!

break;

case 2:

withdrawal(ptrBalance, DAILY\_LIMIT); // passing four byte pointer!

break;

case 3:

// show the balance

cout << fixed << setprecision(2) << "\nCurrent Balance: $" << balance << endl;

break;

case 4:

// get a quick $40

withdrawal(ptrBalance, DAILY\_LIMIT, 40.0f);

break;

case 5:

cout << "\nGoodbye" << endl;

break;

default:

cout << "\nError. Please select from the menu." << endl;

}

// pause

cout << "\nPress any key to continue...";

\_getch();

} while (choice != EXIT\_VALUE);

// now that the application is over, write the new balance to the file

ofstream oFile(FILENAME.c\_str());

oFile << balance << endl;

oFile.close();

return 0;

}

/// Make a deposit

void deposit(double\* ptrBalance)

{

//get deposit and validate it

float deposit = 0.0f;

do

{

cout << "\nEnter deposit amount: ";

cin >> deposit;

if (cin.fail()) // did they give us a character instead of a number?

{

cin.clear(); //Clears fail state

cin.ignore(INT16\_MAX, '\n'); // clears keyboard buffer

cout << "\nError. Please us numbers only. \n" << endl;

deposit = -1; // set deposit to a "bad" number

continue; // restart the loop

}

else if (deposit < 0.0f) //check for negative number

cout << "\nError. Invalid deposit amount, \n" << endl;

} while (deposit < 0.0f);

// how do we get teh double value located at the pointer

// Dereference it using an asterisk!

\*ptrBalance += deposit; // same as : \*ptrBalance = \*ptrBalance + deposit;

cout << fixed << setprecision(2) << "\nCurrent ptrBalance: $" << \*ptrBalance << endl; // notice the asterisk

}

/// Make a withdrawal

void withdrawal(double\* ptrBalance, float dailyLimit)

{

//get the withdrawal (you should validate this input)

float amount = 0.0f;

cout << "\nEnter withdrawal amount: ";

cin >> amount;

// call the overloaded method version that takes

// the balance, dailyLimit, and withdrawal amount

withdrawal(ptrBalance, dailyLimit, amount);

}

/// Make a withdrawal - this overload accepts balance, dailyLimit, and withdrawal amountd

void withdrawal(double\* ptrBalance, float dailyLimit, float amount)

{

// take away money from the account and show the balance

if (amount > dailyLimit)

{

cout << "\nError. Amount exceeds daily limit." << endl;

}

else if (amount > \*ptrBalance) // notice the asterisk to dereference the pointer!

{

cout << "\nError. Insufficient funds." << endl;

}

else

{

\*ptrBalance -= amount; // same as: \*ptrBalance = \*ptrBalance - amount

cout << "\nHere is your cash: $" << amount << endl;

}

cout << fixed << setprecision(2) << "\nCurrent Balance: $" << \*ptrBalance << endl;

}