#include <iostream>

#include <sstream>

#include <stdio.h>

#include <conio.h>

#include <fstream>

#include <cstdlib>

\#include <stdlib.h>

#include <iomanip>

#include <Windows.h>

#include <dos.h>

#include <list>

#include <queue>

#include <stack>

#include <string>

#include <thread>

using namespace std;

#include "Data.h"

#include "Header.h"

#include <limits>

#include <opencv2/imgcodecs.hpp>

#include <opencv2/highgui.hpp>

#include <opencv2/imgproc.hpp>

using namespace cv;

videoList::videoList()

{

head = NULL;

string text;

ifstream inFile("movielist.txt", std::ios\_base::app);

getline(inFile, text);

movieSize = stoi(text);

moveNode\* newNode, \* nodePtr;

while (!inFile.eof()) {

newNode = new moveNode;

getline(inFile, text);

newNode->vidID = stoi(text);

getline(inFile, text);

newNode->vidTitle = text;

getline(inFile, text);

newNode->vidGenre = text;

getline(inFile, text);

newNode->vidProd = text;

getline(inFile, text);

newNode->vidCopy = stoi(text);

getline(inFile, text);

newNode->vidImage = text;

newNode->next = NULL;

if (!head) {

head = newNode;

}

else {

nodePtr = head;

while (nodePtr->next)

nodePtr = nodePtr->next;

nodePtr->next = newNode;

}

}

inFile.close();

}

void videoList::addVideo() {

string vidTitle, vidGenre, vidProd, vidImage;

string vidCopy1, vidID1;

int vidCopy, vidID;

cout << "\n\n\n\n\n\n\n\n";

cout << "\t|-------------------------------<< Add a New Movie >>-----------------------------|" << endl;

cout << "\t|^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^|" << endl;

cout << "\t Movie ID: ";

cout << ++movieSize;

cout << "\n\t Movie Title: ";

getline(cin >> ws, vidTitle);

cout << "\t Movie Genre: ";

getline(cin >> ws, vidGenre);

cout << "\t Movie Production: ";

getline(cin >> ws, vidProd);

cout << "\t Number of Copies: ";

getline(cin >> ws, vidCopy1);

cout << "\t Movie Image File: ";

getline(cin >> ws, vidImage);

istringstream(vidCopy1) >> vidCopy; //converting the string into int

istringstream(vidID1) >> vidID;

moveNode\* nodePtr, \* newNode;

newNode = new moveNode;

newNode->vidID = movieSize;

newNode->vidTitle = vidTitle;

newNode->vidGenre = vidGenre;

newNode->vidProd = vidProd;

newNode->vidCopy = vidCopy;

newNode->vidImage = vidImage;

if (!head) {

head = newNode;

}

else {

nodePtr = head;

while (nodePtr->next)

nodePtr = nodePtr->next;

nodePtr->next = newNode;

newNode->next = NULL;

}

}

void videoList::rentVideo(int myID) {

// Variable Declarations

Rent rent;

string choice;

int rentMovieCode;

int moviecounter = 0;

// Local Stack Declarations

stack <int> videos;

stack <int> custies;

ofstream outFile;

outFile.open("rent.txt", ios\_base::app);

custies.push(myID);

//label

a:

choice = "";

cout << endl << "Enter the movie ID you wish to rent: ";

cin >> rentMovieCode;

videos.push(rentMovieCode);

cout << "Do you want to rent more [Yes/No]: ";

cin >> choice;

// If the customer wants to rent more

if (choice == "yes" || choice == "YES" || choice == "y")

{

moviecounter++;

moveNode\* nodePtr = new moveNode();

nodePtr = head;

while (nodePtr != NULL)

{

// if there is a stored movie ID that is equal to rentMovieCode

if (rentMovieCode == nodePtr->vidID && nodePtr->vidID != 0)

{

nodePtr->vidCopy--;

cout << "The movie has successfully been rented!\n\n";

cout << "Processing...";

}

//if there are no stored movie ID that is equal to rentMovieCode

else if (nodePtr->vidID == 0) // Counter starts at 1, so no movies

{

cout << "The movie you entered is unavailable!!";

return;

}

nodePtr = nodePtr->next;

}

// go back to ‘a’ label

goto a;

}

this->vidDetails(rentMovieCode);

moveNode\* nodePtr = new moveNode();

nodePtr = head;

while (nodePtr != NULL)

{

if (rentMovieCode == nodePtr->vidID && nodePtr->vidID != 0)

{

nodePtr->vidCopy--;

cout << "The movie has successfully been rented!\n\n";

cout << "Processing...";

Sleep(2000);

}

else if (nodePtr->vidID == 0)

{

cout << "The movie you entered is unavailable!!";

Sleep(2000);

return;

}

nodePtr = nodePtr->next;

}

rent.saveRent(videos, custies);

}

void videoList::returnVideo(int myID) {

//declare Rent and Customers object

Rent rent;

Customers customer;

int returnMovieCode;

// Stack Declaration

stack <int> movies;

stack <int> custies;

// FileRead

ifstream inFile;

inFile.open("rent.txt");

string line = "";

stringstream str1;

if (inFile.is\_open()) // While the File is Open

{

while (getline(inFile, line)) {

str1.str(line);

int x;

int y;

str1 >> x; //Customer ID

str1 >> y; //Movie ID

str1.clear();

if (x == myID)

{

// Store into stack

custies.push(x);

movies.push(y);

}

}

inFile.close();

}

bool isThere = true;

while (!custies.empty() && isThere) // Check if the customer rented a video

{

if (custies.top() == myID)

{

isThere = false;

}

else

{

custies.pop();

}

}

// If Customer did not rent any video

if (custies.empty()) {

cout << "Customer did not rent any movies" << endl;

Sleep(2000);

return;

}

// Else

cout << "\nEnter the movie ID you wish to return: ";

cin >> returnMovieCode;

moveNode\* nodePtr = new moveNode();

nodePtr = head;

while (nodePtr != NULL && nodePtr->vidID != returnMovieCode) { // Node Traversal

nodePtr = nodePtr->next;

}

if (nodePtr != NULL) {

cout << "You have successfully returned a movie!\n\n" << endl;

nodePtr->vidCopy++;

this\_thread::sleep\_for(chrono::milliseconds(1500));

cout << "Processing...";

Sleep(2000);

rent.returnMovie(myID, returnMovieCode);

}

else

{

cout << "Movie does not exist in the system, thus cannot return movie!\n\n";

Sleep(2000);

}

}

void videoList::vidDetails(int myID) {

moveNode\* nodePtr;

if (head == NULL)

cout << "The List is empty";

}

else {

nodePtr = head;

while (nodePtr) {

if (nodePtr->vidID == myID) {

cout << "Movie Details" << endl;

cout << "Title: " << nodePtr->vidTitle << endl;

cout << "Movie Genre: " << nodePtr->vidGenre << endl;

cout << "Movie Production: " << nodePtr->vidProd << endl;

cout << "Movie Image Filename: ";

std::string h1 = nodePtr->vidImage;

Mat img\_h1 = imread(h1);

Mat imgResize\_h1;

resize(img\_h1, imgResize\_h1, Size(640, 480));

imshow("Image Resize", imgResize\_h1);

waitKey(0);

cout << nodePtr->vidImage << endl << endl;

}

nodePtr = nodePtr->next;

}

}

}

void videoList::vidDisplay() {

moveNode\* nodePtr;

if (head == NULL) {

cout << "The list is empty" << endl;

}

else {

nodePtr = head;

cout << "ID number " << " Title " << " Genre " << " Production " << " Copy" << endl;

while (nodePtr != NULL) {

cout << setw(5) << nodePtr->vidID;

cout << setw(39) << nodePtr->vidTitle;

cout << setw(20) << nodePtr->vidGenre;

cout << setw(50) << nodePtr->vidProd;

cout << setw(5) << nodePtr->vidCopy;

cout << endl;

nodePtr = nodePtr->next;

}

}

}

void videoList::vidAvail(int myID) {

moveNode\* nodePtr;

nodePtr = head;

if (head == NULL){

cout << "The List is empty";

}

else if (nodePtr->vidID == myID && nodePtr->vidCopy == 0)

{

cout << "The movie is not available";

}

else {

nodePtr = head;

while (nodePtr) {

if (nodePtr->vidID == myID) {

cout << "The movie title is available! and the title is: ";

cout << nodePtr->vidTitle;

cout << "\nThe number of copy available is: ";

cout << nodePtr->vidCopy;

}

nodePtr = nodePtr->next;

}

}

}

void videoList::saveMovie() {

ifstream inFile("movielist.txt");

remove("tempmovie.txt");

string line;

ofstream outFile("tempmovie.txt");

getline(inFile, line);

movieSize = stoi(line);

outFile << movieSize;

moveNode\* nodePtr;

if (head == NULL)

cout << " The List is empty!" << endl;

else {

nodePtr = head;

while (nodePtr) {

getline(inFile, line);

outFile << endl;

outFile << nodePtr->vidID << endl;

outFile << nodePtr->vidTitle << endl;

outFile << nodePtr->vidGenre << endl;

outFile << nodePtr->vidProd << endl;

outFile << nodePtr->vidCopy << endl;

outFile << nodePtr->vidImage;

nodePtr = nodePtr->next;

}

}

inFile.close();

outFile.close();

remove("movielist.txt");

char oldname[] = "tempmovie.txt";

char newname[] = "movielist.txt";

rename(oldname, newname);

}

void videoList::vidLoad(int myID)

{

moveNode\* mov = new moveNode();

moveNode\* nodePtr = head;

ifstream inFile;

inFile.open("rent.txt");

stringstream str;

string line = "";

int c;

int m;

if (inFile.peek() == inFile.eof()) {

cout << "No videos rented!";

return;

}

while (getline(inFile, line)) {

str.str(line);

str >> c; //Customer ID

str >> m; //Movie ID

if (myID == c)

{

while (nodePtr != NULL && nodePtr->vidID != m)

{

nodePtr = nodePtr->next;

}

mov = nodePtr;

cout << mov->vidID << "\t\t" << mov->vidTitle << "\n\n";

}

str.clear();

}

//inFile.close();

Sleep(2000);

}