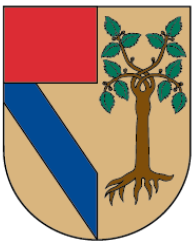


UNIVERSIDAD PANAMERICANA®

Alumnos:

Arias Cortina Carlos 0217787

Cabrera Ábrego Raúl Andrés 0213359

Villalobos Santiago Carlos Manuel. 0220791

Profesor:

Aguilar Juárez Francisco Aguilar

Proyecto:

Aplicación Cinema

Fecha de entrega:

20 de noviembre de 2019

Descripción del problema:

Conforme el paso del tiempo, la tecnología ha ido avanzando de manera impresionante, tomando un lugar radical en nuestra vida.

Todo nuestro alrededor ha ido adaptándose a este nuevo modo de vida, en donde la optimización de los procesos juega un papel fundamental, y todo es posible gracias al uso de la tecnología.

Un ejemplo son las aplicaciones digitales, que nos permiten realizar actividades desde la comodidad de la casa donde antes eran posibles solo asistiendo a un lugar en específico.

Los cines no son una excepción, ya que en nuestros tiempos ya existe la posibilidad de revisar las funciones más próximas a tu localización, la disponibilidad de boletos y hasta la facilidad de comprarlos ahí mismo, ¡sin la necesidad de ir al cine!

Este es el enfoque por el cual nuestra aplicación se dirige, y la crearemos a partir del conocimiento adquirido durante nuestra clase de Programación y de Datos.

Nuestra aplicación es diseñada para ser usada por un cine, mejorando la experiencia del cliente, tiene diversas funciones personalizadas para que el usuario pueda buscar la película deseada en el horario que se le haga más conveniente, para un mejor entendimiento de la aplicación se podrá ordenar por calificación del filme, de igual manera en orden alfabético, una vez que se haya comprado el boleto se enviará una orden de confirmación directamente al correo ya antes ingresado, comprar boletos en la aplicación ofrece muchos beneficios, como descuentos o puntos acumulados, esto va incitar a que el público introduzca su dinero en el sistema haciendo que consuman más en nuestro cine.

Explicación de la lógica de solución (diseño):

El programa inicia con un menú que a su vez se divide en 2 submenús (usuario y administrador).

En el menú de usuario, debes iniciar sesión (con) tu correo o nombre de usuario) o crearte una cuenta. Después puedes acceder desde ese menú al menú principal donde tienes las opciones de:
-Mostrar películas por rating (Estructura de Árbol)
-Mostrar películas por horarios (Estructura de Árbol)
-Mostrar alfabéticamente las películas (Lista Enlazada)
-Comprar boletos de una función específica según los asientos disponibles (Arreglos)

Después, en el menú de administrador (igual debes iniciar sesión, no puedes crear una cuenta más que otro administrador te la cree) puedes editar películas (se abre otro submenú

donde puedes mostrar, borrar y agregar películas), esto con la estructura de Listas Enlazadas.
También puedes ver todos los usuarios que hay en la plataforma.

Problem statement:

Technology has advanced in an incredible way within the time going by.

All our surroundings have been adapting to this new lifestyle, where the optimization of processes play a fundamental role thanks to the usage of technology.

For instance, digital applications let us make activities from the comfort of our own home that were only available to make by going to an specific place.

Movie theaters are no exception, since in our times there already is the possibility to check the closest showings to your location, the availability of tickets and even the ease of buying them right there, without the need to go to the movie theater!

This is the approach by which our application is directed, and we will create it from the knowledge acquired during our Programming and Structure Data class.

Our application is designed to be used by a movie theater, improving the customer experience, it has several personalized functions so that the user can search for the desired movie at the time that is most convenient for him, for a better understanding of the application you can order by rating of the film, in the same way in alphabetical order, once the ticket has been purchased a confirmation order will be sent directly to the mail that has been already entered, buying tickets in the application offers many benefits, such as discounts or accumulated points, this will encourage the public to enter their money into the system causing them to consume more in our movie theater.

Explanation of the logic behind the solution

The application begins with a menu that is divided in two submenus, the first one corresponds to the admin menu and the second one to the user menu.
Un the user's menu, you need to log in (using your email or a username) or create an account.

Next, you can have access to the main menu where you have the options of:

-Show movies by rating (using the data structure of Trees)

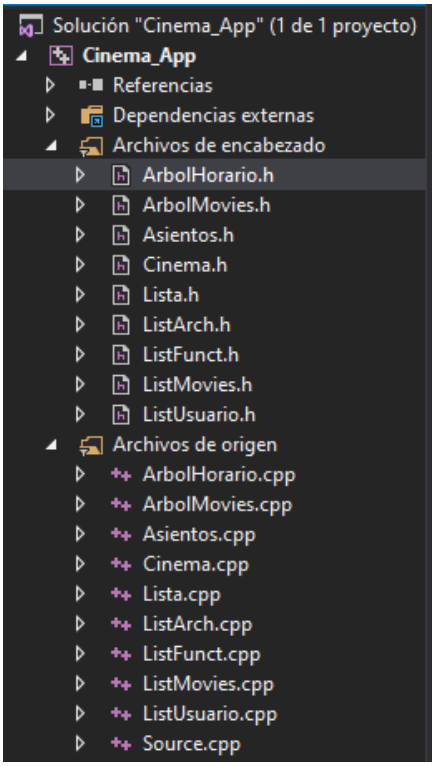
-Show movies by showtime (using the data structure of Trees)

-Show movies alphabetically (using the data structure of Linked list)

-Buy tickets for a specific movie seeing seat availability (using the data structure of arrays)

Afterwards, in the admin menu (you also have to log in, but in this case, only an admin can make someone an admin) you can edit the movies (another menu opens where you can show, delete and add movies) with the help of dynamic lists.
Last but not least, you can check how many users are registered on this platform.

Código fuente:




```
#pragma once
#include <iostream>
#include <locale>
#include <wchar.h>
using namespace std;
struct Hora
{
	string peli;
	int sal;
	string hrs;
	string nomfun;
```

```
	int HrsI;
	int HrsF;
	Hora* izq;
	Hora* der;
};
class ArbolHorario
{
public:
	ArbolHorario();
	~ArbolHorario();
	Hora* Consultar_Raiz();
	void Crear_Nodo(string nfun, int nsal, string nhrs, string npeli, int NHrsI,
int NHrsF);
	void Insertar(Hora* inicio);
	void Recorrer_In_Orden(Hora* inicio);
	void Extraer(Hora* padre, Hora* inicio, int valor, string SubArbol);
	void Intercambiar(Hora* encontrar, Hora* actual, Hora* siguiente, string
subArbol);
private:
	Hora* nodo, * aux, * raiz;
	string padreEnlace;
};
```

```
#pragma once
#include <iostream>
#include <locale>
#include <wchar.h>
using namespace std;
struct Califica
{
	string peli;
	string gene;
	float cali;
	int id;
	int num;
	Califica* izq;
	Califica* der;
};
class ArbolMovies
{
public:
	ArbolMovies();
	~ArbolMovies();
	Califica* Consultar_Raiz();
	void Crear_Nodo(string pel, string gen, float cal, int idx);
	void Insertar(Califica* inicio);
	void Recorrer_In_Orden(Califica* inicio);
	void Extraer(Califica* padre, Califica* inicio, int valor, string SubArbol);
	void Intercambiar(Califica* encontrar, Califica* actual, Califica* siguiente,
string subArbol);
private:
	Califica* nodo, * aux, * raiz;
	string padreEnlace;
};
```

```h
#pragma once
#include <iostream>
#include <locale.h>
#include <wchar.h>
#include <string>
#include <fstream>
using namespace std;
struct Seat
{
	string A1,A2,A3,A4,A5,A6,A7,A8;
	Seat* sig;
};
class Asientos
{
public:
	Asientos();
	~Asientos();
	void Mostrar_Asientos();
	void Insertar_Asientos(string xA1, string xA2, string xA3, string xA4, string
xA5, string xA6, string xA7, string xA8);
	string Cambiar_Asientos(string ext);
	void Borrar_Todo();
	string Extrae_Fin();
	bool Buscar_Asiento(string usu);
	void Reiniciar_BD(string nom);
private:
	Seat* cabecera, * fin, * nodo;

};
```

```h
#pragma once
#include "Lista.h"
#include "ListMovies.h"
#include "ListFunct.h"
#include "Asientos.h"
#include "ListArch.h"
#include "ListUsuario.h"
#include "ArbolMovies.h"
#include "ArbolHorario.h"
#include <iostream>
#include <wchar.h>
#include <locale.h>
#include <vector>
#include <fstream>
#include <string>
#include <Windows.h>
#include <stdlib.h>
#include <random>
#include <tchar.h>
#include "EASendMailObj.tlh"
using namespace std;
typedef class  ListMovies ListMovies;
typedef class Lista Lista;
```

```
typedef class ListFunct ListFunct;
typedef class Asientos Asientos;
typedef class ListArch ListArch;
typedef class ListUsuario ListUsuario;
typedef class ArbolMovies ArbolMovies;
typedef class ArbolHorario ArbolHorario;

using namespace EASendMailObjLib;

const int ConnectNormal = 0;
const int ConnectSSLAuto = 1;
const int ConnectSTARTTLS = 2;
const int ConnectDirectSSL = 3;
const int ConnectTryTLS = 4;
class Cinema
{
public:
	Cinema();
	~Cinema();
	void Imprimir(vector<string>&, int);
	void Menu();
	void Menu_Admin();
	void Validar_Admin();
	void LeeryCopiarAdmin();
	void Agregar_Admin();
	void Eliminar_Admin();
	void LeeryCopiarMovie();
	void Edit_Movie();
	void Agregar_Movie();
	void Eliminar_Movie();
	void LeeryCopiarSalaHora();
	void Agregar_Sala_Hora();
	void Eliminar_Sala_Hora();
	void LeeryCopiarUsu();
	void Menu_Usuario();
	void Iniciar_Sesion();
	void Crear_Cuenta();
	void Menu_Prin(string nomUsu);
	void Comprar_Boletos(string Usu);
	void LeeryCopiarArch();
	void CrearyCopiarFun(string nomfun);
	void LeeryCopiarFun(string nomfun);
	void Enviar_Mail(int codi,string correo,string nom,string hrs,int sala,string
asiento,string peli);
private:
	Lista* miLista;
	ListMovies* miListMovies;
	ListFunct* miListFunct;
	Asientos* miAsientos;
	ListArch* miListArch;
	ListUsuario* miListUsuario;
	ArbolMovies* miArbolMovies;
	ArbolHorario* miArbolHorario;
	int ID = 0,ID2 = 0;
	vector<string> Menu_1 = { "1) Menu de Usuario","2) Menu de Administrador","3)
Salir" };
```

```
	vector<string>Menu_Adm = { "1) Mostrar Administradores","2) Agregar un
Administrador","3) Eliminar un Administrador","4) Editar Películas","5) Mostrar
Usuarios","6) Regresar a Menu" };
	vector<string> Menu_Edit = { "1) Mostrar Películas" ,"2) Agregar
Película","3) Eliminar Película","4) Mostrar Funciones","5) Agregar Funciones","6)
Eliminar Funciones","7) Regresar a Menu de Administrador"};
	vector<string> Menu_Usu = { "1) Iniciar Sesion","2) Crear Cuenta","3)
Regresar a Menu" };
	vector<string> Menu_Pin = { "1) Mostrar Películas por Calificación","2)
Mostrar Películas por Horario","3) Mostrar Películas en Orden Alfabetico","4)
Comprar Boletos","5) Ingresar dinero en la Cuenta","6) Mostrar Estado de Cuenta","7)
Regresar a Menu de Usuario","8) Salir" };
};
```

```
#pragma once
#include <iostream>
#include <locale.h>
#include <wchar.h>
#include <string>
#include <fstream>
using namespace std;
struct Administrador
{
	string nom;
	string contra;
	Administrador* sig;
};
class Lista
{
public:
	Lista();
	~Lista();
	void Mostrar_Admin();
	void Insertar_Fin(string nuevo,string con);
	string Extrae_Prin();
	string Extrae_Inter(string ext);
	string Extrae_Fin();
	bool Buscar_Val_Admin(string usu,string con);
	bool Buscar_Admin(string usu);
	void Reiniciar_BD();
private:
	Administrador* cabecera, * fin, * nodo;
};
```

```
#pragma once
#include <iostream>
#include <locale.h>
#include <wchar.h>
#include <string>
```

```cpp
#include <fstream>
using namespace std;
struct Archivo
{
	string nomfun;
	Archivo* sig;
};
class ListArch
{
public:
	ListArch();
	~ListArch();
	void Mostrar_Archivo();
	void Insertar_Fin(string nuevo);
	string Extrae_Prin();
	string Extrae_Inter(string ext);
	string Extrae_Fin();
	bool Buscar_Archivo(string usu);
	void Reiniciar_BD();
	bool Buscar_Funcion(string fun);
private:
	Archivo* cabecera, * fin, * nodo;
};
```

```cpp
#pragma once
#include <iostream>
#include <locale.h>
#include <wchar.h>
#include <string>
#include <fstream>
using namespace std;
struct Funcion
{
	string peli;
	int sal;
	string hrs;
	string nomfun;
	int HrsI;
	int HrsF;
	int ID;
	Funcion* sig;
};
class ListFunct
{
public:
	ListFunct();
	~ListFunct();
	void Mostrar_Por_ID(string pel);
	void Mostrar_Sala_Hora();
	void Insertar_Sala_Hora(string nfun, int nsal, string nhrs,string npeli,int
NHrsI,int NHrsF,int ID);
	string Extrae_Prin();
	string Extrae_Inter(string ext,int xsal);
	string Extrae_Fin();
```

```
        void Reiniciar_BD();
        Funcion Buscar_Movie(int idx);
private:
        Funcion* cabecera, * fin, * nodo;
};
```

```
#pragma once
#include <iostream>
#include <locale.h>
#include <wchar.h>
#include <string>
#include <fstream>
using namespace std;
struct Movie
{
        string peli;
        string gene;
        float cali;
        int id;
        Movie* sig;
        Movie* sigalf;
};
class ListMovies
{
public:
        ListMovies();
        ~ListMovies();
        void Mostrar_Por_ID();
        void Mostrar_Movie();
        void Mostrar_Movie_y_Seleccionar();
        void Mostrar_Alfa();
        void Insertar_Movie(string pel, string gen,float cal,int idx);
        void Insertar_Alfa(string pel, string gen, float cal, int idx);
        string Extrae_Prin_Movie();
        string Extrae_Inter_Movie(int idx);
        string Extrae_Fin_Movie();
        string Extrae_Prin_Alfa();
        string Extrae_Inter_Alfa(int idx);
        string Extrae_Fin_Alfa();
        Movie Buscar_Movie(int idx);
        void Reiniciar_BD_Movie();
private:
        Movie* cabecera, * fin, * nodo, * aux,*ante;
        Movie* cabealfa, * finalfa, * nodoalfa;
};
```

```
#pragma once
#include <iostream>
#include <locale.h>
#include <wchar.h>
#include <string>
```

```cpp
#include <fstream>
using namespace std;
struct Usuario
{
	string correo;
	string nom;
	string contra;
	int cash;
	Usuario* sig;
};
class ListUsuario
{
public:
	ListUsuario();
	~ListUsuario();
	void Mostrar_Usuarios();
	void Insertar_Fin(string xcorr, string xnom,string xcontra,int xcash);
	string Extrae_Prin();
	string Extrae_Inter(string ext);
	string Extrae_Fin();
	bool Buscar_Val_Usuario(string usu, string con);
	bool Buscar_Usuario(string usu);
	void Reiniciar_BD();
	Usuario Buscar_Info_Usuario(string nomUsu);
	bool Cambiar_Cash(string usu, int rest);
	bool Sumar_Cash(string usu, int sum);
private:
	Usuario* cabecera, * fin, * nodo;
};
```

```cpp
#include "ArbolHorario.h"
ArbolHorario::ArbolHorario()
{
	setlocale(LC_ALL, "");
	raiz = NULL;
}
ArbolHorario::~ArbolHorario() {}
Hora* ArbolHorario::Consultar_Raiz()
{
	return raiz;
}
void ArbolHorario::Crear_Nodo(string nfun, int nsal, string nhrs, string npeli, int
NHrsI, int NHrsF)
{
	nodo = new Hora();
	nodo->nomfun = nfun;  //Paso 2
	nodo->sal = nsal;
	nodo->hrs = nhrs;
	nodo->peli = npeli;
	nodo->HrsI = NHrsI;
	nodo->HrsF = NHrsF;
	nodo->izq = NULL;
	nodo->der = NULL;
```

```
}
void ArbolHorario::Insertar(Hora* inicio)
{
	if (!raiz)                        //Paso 0
	{
		raiz = nodo;
		/*cout << "Ok. Se insertó como raíz" << endl;*/
		return;
	}
	if (nodo->HrsI == inicio->HrsI)   //Paso 1
	{
		/*cout << "Ya existe " << nodo->num << " NO se insertó " << endl;*/
		return;
	}

	if (nodo->HrsI < inicio->HrsI)   //Paso 2
	{
		if (!inicio->izq) //== NULL
		{
			inicio->izq = nodo;
			/*cout << "Ok. Se insertó a la izq de " << inicio->num <<
endl;*/
			return;
		}
		Insertar(inicio->izq);
	}

	if (nodo->HrsI > inicio->HrsI)   //Paso 3
	{
		if (!inicio->der)
		{
			inicio->der = nodo;
			/*cout << "Ok. Se insertó a la der de " << inicio->num <<
endl;*/
			return;
		}
		Insertar(inicio->der);
	}


}
void ArbolHorario::Recorrer_In_Orden(Hora* inicio)
{

	if (inicio->izq)
		Recorrer_In_Orden(inicio->izq);

	cout << "\t" << inicio->hrs << "\t" << inicio->peli << endl;

	if (inicio->der)
		Recorrer_In_Orden(inicio->der);
}

void ArbolHorario::Extraer(Hora* padre, Hora* inicio, int valor, string SubArbol)
{
	if (valor < inicio->HrsI)
	{
		Extraer(inicio, inicio->izq, valor, "izq");
```

```
                return;
        }
        if (valor > inicio->HrsI)
        {
                Extraer(inicio, inicio->der, valor, "der");
                return;
        }
        if (inicio->izq == NULL && inicio->der == NULL)
        {
                /*cout << "\tOK. Se extrajó el valor " << inicio->cali << endl <<
endl;*/
                if (SubArbol == "izq")
                        padre->izq = NULL;
                if (SubArbol == "der")
                        padre->der = NULL;
                if (raiz == inicio)
                        raiz = NULL;
                delete inicio;
                return;
        }
        if (inicio->izq != NULL)
        {
                padreEnlace = "izq";
                Intercambiar(inicio, inicio, inicio->izq, "izq");
                return;
        }
        if (inicio->der != NULL)
        {
                padreEnlace = "der";
                Intercambiar(inicio, inicio, inicio->izq, "der");
                return;
        }
}
void ArbolHorario::Intercambiar(Hora* encontrar, Hora* actual, Hora* siguiente,
string subArbol)
{
        int aux;
        if (subArbol == "izq")
        {
                if (siguiente->der == NULL)
                {
                        aux = encontrar->HrsI;
                        encontrar->HrsI = siguiente->HrsI;
                        siguiente->HrsI = aux;
                        Extraer(actual, siguiente, siguiente->HrsI, padreEnlace);
                        return;
                }
        }
        else
        {
                padreEnlace = "der";
                Intercambiar(encontrar, siguiente, siguiente->der, "izq");
                return;
        }
        if (subArbol == "der")
        {
                if (siguiente->izq == NULL)
                {
```

```
                        aux = encontrar->HrsI;
                        encontrar->HrsI = siguiente->HrsI;
                        siguiente->HrsI = aux;
                        Extraer(actual, siguiente, siguiente->HrsI, padreEnlace);
                        return;
                }
        }
        else
        {
                padreEnlace = "izq";
                Intercambiar(encontrar, siguiente, siguiente->izq, "der");
                return;
        }
}


#include "ArbolMovies.h"
ArbolMovies::ArbolMovies()
{
        setlocale(LC_ALL, "");
        raiz = NULL;
}
ArbolMovies::~ArbolMovies() {}
Califica* ArbolMovies::Consultar_Raiz()
{
        return raiz;
}
void ArbolMovies::Crear_Nodo(string pel, string gen, float cal, int idx)
{
        nodo = new Califica();
        nodo->peli = pel;
        nodo->gene = gen;
        nodo->cali = cal;
        nodo->id = idx;
        nodo->izq = NULL;
        nodo->der = NULL;
}
void ArbolMovies::Insertar(Califica* inicio)
{
        if (!raiz)                          //Paso 0
        {
                raiz = nodo;
                /*cout << "Ok. Se insertó como raíz" << endl;*/
                return;
        }
        if (nodo->cali == inicio->cali)   //Paso 1
        {
                /*cout << "Ya existe " << nodo->num << " NO se insertó " << endl;*/
                return;
        }

        if (nodo->cali < inicio->cali)   //Paso 2
        {
                if (!inicio->izq) //== NULL
                {
                        inicio->izq = nodo;
```

```
                        /*cout << "Ok. Se insertó a la izq de " << inicio->num <<
endl;*/
                        return;
                }
                Insertar(inicio->izq);
        }

        if (nodo->cali > inicio->cali)   //Paso 3
        {
                if (!inicio->der)
                {
                        inicio->der = nodo;
                        /*cout << "Ok. Se insertó a la der de " << inicio->num <<
endl;*/
                        return;
                }
                Insertar(inicio->der);
        }


}
void ArbolMovies::Recorrer_In_Orden(Califica* inicio)
{

        if (inicio->izq)
                Recorrer_In_Orden(inicio->izq);

        cout << "\t" << inicio->cali << "\t" << inicio->peli<<endl;

        if (inicio->der)
                Recorrer_In_Orden(inicio->der);
}

void ArbolMovies::Extraer(Califica* padre, Califica* inicio, int valor, string
SubArbol)
{
        if (valor < inicio->cali)
        {
                Extraer(inicio, inicio->izq, valor, "izq");
                return;
        }
        if (valor > inicio->cali)
        {
                Extraer(inicio, inicio->der, valor, "der");
                return;
        }
        if (inicio->izq == NULL && inicio->der == NULL)
        {
                /*cout << "\tOK. Se extrajó el valor " << inicio->cali << endl <<
endl;*/
                if (SubArbol == "izq")
                        padre->izq = NULL;
                if (SubArbol == "der")
                        padre->der = NULL;
                if (raiz == inicio)
                        raiz = NULL;
                delete inicio;
                return;
```

```
        }
        if (inicio->izq != NULL)
        {
                padreEnlace = "izq";
                Intercambiar(inicio, inicio, inicio->izq, "izq");
                return;
        }
        if (inicio->der != NULL)
        {
                padreEnlace = "der";
                Intercambiar(inicio, inicio, inicio->izq, "der");
                return;
        }
}
void ArbolMovies::Intercambiar(Califica* encontrar, Califica* actual, Califica*
siguiente, string subArbol)
{
        int aux;
        if (subArbol == "izq")
        {
                if (siguiente->der == NULL)
                {
                        aux = encontrar->cali;
                        encontrar->cali = siguiente->cali;
                        siguiente->cali = aux;
                        Extraer(actual, siguiente, siguiente->cali, padreEnlace);
                        return;
                }
        }
        else
        {
                padreEnlace = "der";
                Intercambiar(encontrar, siguiente, siguiente->der, "izq");
                return;
        }
        if (subArbol == "der")
        {
                if (siguiente->izq == NULL)
                {
                        aux = encontrar->cali;
                        encontrar->cali = siguiente->cali;
                        siguiente->cali = aux;
                        Extraer(actual, siguiente, siguiente->cali, padreEnlace);
                        return;
                }
        }
        else
        {
                padreEnlace = "izq";
                Intercambiar(encontrar, siguiente, siguiente->izq, "der");
                return;
        }
}



#include "Asientos.h"
```

```
Asientos::Asientos()
{
	setlocale(LC_ALL, "");
	cabecera = NULL;
	fin = NULL;
}
Asientos::~Asientos() {}
void Asientos::Mostrar_Asientos()
{
	if (!cabecera) //cabecera == NULL
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return;
	}
	nodo = cabecera;

	while (nodo)//nodo != NULL
	{
		cout << "\t| " << nodo->A1 << " | " << nodo->A2 << " | " << nodo->A3 <<
" | " << nodo->A4 << " | " << nodo->A5 << " | " << nodo->A6 << " | " << nodo->A7 <<
" | " << nodo->A8 << " |";
		cout << endl;
		nodo = nodo->sig;
	}
	cout << "\n\t_____________________________________" << endl;
	cout << "\t              PANTALLA               " << endl;
	cout << "\t_____________________________________" << endl;
}

void Asientos::Insertar_Asientos(string xA1, string xA2, string xA3, string xA4,
string xA5, string xA6, string xA7, string xA8)
{
	nodo = new Seat;   //Paso 1
	nodo->A1 = xA1;    //Paso 2
	nodo->A2 = xA2;
	nodo->A3 = xA3;
	nodo->A4 = xA4;
	nodo->A5 = xA5;
	nodo->A6 = xA6;
	nodo->A7 = xA7;
	nodo->A8 = xA8;

	nodo->sig = NULL;     //Paso 3
	if (fin)              //fin != NULL
		fin->sig = nodo;
	if (!cabecera)   //cabecera == NULL
		cabecera = nodo;
	fin = nodo;           //Paso 4
	//cout << "\tOK.Se insertó " << nuevo << endl;
}

string Asientos::Cambiar_Asientos(string ext)
{
	string extraido = "";
	Seat* anterior;
	bool bandera = false;

	if (!cabecera)
```

```
{
	cout << "\n\t La lista está vacía ** \n" << endl;
	return extraido;
}

anterior = NULL;
nodo = cabecera;

while (nodo != NULL)
{
	if (nodo->A1 == ext)
	{
		extraido = nodo->A1;
		nodo->A1 = "OC";
		bandera = true;
		break;
	}
	if (nodo->A2 == ext)
	{
		extraido = nodo->A2;
		nodo->A2 = "OC";
		bandera = true;
		break;
	}
	if (nodo->A3 == ext)
	{
		extraido = nodo->A3;
		nodo->A3 = "OC";
		bandera = true;
		break;
	}
	if (nodo->A4 == ext)
	{
		extraido = nodo->A4;
		nodo->A4 = "OC";
		bandera = true;
		break;
	}
	if (nodo->A5 == ext)
	{
		extraido = nodo->A5;
		nodo->A5 = "OC";
		bandera = true;
		break;
	}
	if (nodo->A6 == ext)
	{
		extraido = nodo->A6;
		nodo->A6 = "OC";
		bandera = true;
		break;
	}
	if (nodo->A7 == ext)
	{
		extraido = nodo->A7;
		nodo->A7 = "OC";
		bandera = true;
		break;
```

```
			}
			if (nodo->A8 == ext)
			{
				extraido = nodo->A8;
				nodo->A8 = "OC";
				bandera = true;
				break;
			}
			anterior = nodo;
			nodo = nodo->sig;
		}
		if (!bandera)
		{
			cout << "\nNo Existe " << ext << endl;
			return extraido;
		}

		return extraido;
}


bool Asientos::Buscar_Asiento(string ext)
{
		bool encontrado = false;
		if (!cabecera) //cabecera == NULL
		{
			encontrado = false;
		}
		nodo = cabecera;
		while (nodo)//nodo != NULL
		{

			if (nodo->A1 == ext)
			{
				encontrado = true;

				break;
			}
			if (nodo->A2 == ext)
			{
				encontrado = true;

				break;
			}
			if (nodo->A3 == ext)
			{
				encontrado = true;

				break;
			}
			if (nodo->A4 == ext)
			{
				encontrado = true;

				break;
			}
			if (nodo->A5 == ext)
			{
```

```
                        encontrado = true;

                        break;
                }
                if (nodo->A6 == ext)
                {
                        encontrado = true;

                        break;
                }
                if (nodo->A7 == ext)
                {
                        encontrado = true;

                        break;
                }
                if (nodo->A8 == ext)
                {
                        encontrado = true;

                        break;
                }
                nodo = nodo->sig;
        }
        return encontrado;
}
void Asientos::Reiniciar_BD(string nom)
{
        string doc = nom;
        doc.append(".csv");
        ofstream file(doc);
        nodo = cabecera;
        if (file.is_open())
        {
                file << nom << endl;
                while (nodo)//nodo != NULL
                {
                        file << nodo->A1 << ";" << nodo->A2 << ";" << nodo->A3 << ";" <<
nodo->A4 << ";" << nodo->A5 << ";" << nodo->A6 << ";" << nodo->A7 << ";" << nodo->A8
<< endl;
                        nodo = nodo->sig;
                }
        }
}
string Asientos::Extrae_Fin()
{
        Seat* anterior;
        string extraido = "";
        if (!cabecera)
        {
                cout << "\n\t La lista está vacía." << endl;
                return extraido;
        }

        anterior = NULL;
        nodo = cabecera;
        while (nodo->sig != NULL) //  Paso 1
        {
```

```
                anterior = nodo;
                nodo = nodo->sig;
        }
        extraido = nodo->A1; //  Paso 2

        if (!anterior)
        {
                cabecera = NULL;
        }
        else
        {
                anterior->sig = NULL; //  Paso 3
        }
        fin = anterior; //  Paso 4
        delete nodo; // Paso 5
        return extraido;

}
void Asientos::Borrar_Todo()
{
        string val = "a";
        do
        {
                val = Extrae_Fin();
        } while (val != "");
}
```

```
#include "Cinema.h"

Cinema::Cinema()

{

        miLista = new Lista;

        miListMovies = new ListMovies;

        miListFunct = new ListFunct;

        miAsientos = new Asientos;

        miListArch = new ListArch;

        miListUsuario = new ListUsuario;

        miArbolMovies = new ArbolMovies;

        miArbolHorario = new ArbolHorario;

        setlocale(LC_ALL, "");

}

Cinema::~Cinema() {}
```

```
void Cinema::Imprimir(vector<string>&str, int num)
{
	cout << endl;
	for (int i = 0; i < num; i++)
	{
		cout << "\t" << str[i] << "\n" << endl;
	}
	cout << "_____" << endl;
}
void Cinema::Menu()
{
	int opc,val = 0;
	do
	{
		cout << "_____Cinema_____" << endl;
		Imprimir(Menu_1, Menu_1.size());
		cout << "Digite la opión deseada (1/2/3) : ";
		cin >> opc;
		system("cls");
		if (opc > 0 && opc < 4)
			val = 1;
	} while (val == 0);
		switch (opc)
		{
		case 1:
			Menu_Usuario();
			break;
		case 2:
			Validar_Admin();
```

```
                    break;
                default:
                    exit(0);
                    break;
                }

}
void Cinema::Validar_Admin()
{
    string usuario, contra;
    bool val = false;
    int fail = 0;
    do
    {
        cout << "\n_______Validación_______" << endl<<endl;
        cout << "  Administrador : ";
        cin.ignore();
        getline(cin, usuario);
        cout << "  Contraseña : ";
        getline(cin, contra);
        system("cls");
        val = miLista->Buscar_Val_Admin(usuario, contra);
        if (val == true)
        {
            Menu_Admin();
        }
        if (val == false)
        {
            fail += 1;
```

```
		}
		if (fail == 3)
		{
			cout << "Por motivos de seguridad seras regresado al menu." << endl;
			Menu();
		}
	} while (val == false);
}
void Cinema::Menu_Admin()
{

	int opc, val = 0;
	do
	{
		cout << "_____Menu de Administrador_____" << endl;
		Imprimir(Menu_Adm, Menu_Adm.size());
		cout << "Digite la opión deseada (1/2/3/4/5/6) : ";
		cin >> opc;
		system("cls");
		if (opc > 0 && opc < 7)
			val = 1;
	} while (val == 0);
	switch (opc)
	{
	case 1:
		miLista->Mostrar_Admin();
		system("pause()");
		system("cls");
		Menu_Admin();
```

```
                break;
        case 2:
           Agregar_Admin();
                Menu_Admin();
                break;
        case 3:
                Eliminar_Admin();
                Menu_Admin();
                break;
        case 4:
                Edit_Movie();
                Menu_Admin();
                break;
        case 5:
                miListUsuario->Mostrar_Usuarios();
                system("pause()");
                system("cls");
                Menu_Admin();
                break;
        default:
                Menu();
                break;
        }

}
void Cinema::LeeryCopiarAdmin()
{
        string usuario="", contra ="";
        ifstream file;
```

```
file.open("Administrador.csv");
string line;
int i = 0;
while (file.good())
{

	getline(file, line);//2
	if (!(i == 0))
	{
		for (int n = 0; n < line.length(); n++)
		{
			contra.push_back(line[n]);


		}


	}
	if (contra != "" && usuario != "")
	{
		miLista->Insertar_Fin(usuario, contra);
		usuario = ""; contra = "";
	}
	getline(file, line, ';');//1

	for (int n = 0; n < line.length(); n++)
		{
			usuario.push_back(line[n]);
		}
	i++;
}
```

```
	/*numAdmin = i-1;*/
}
void Cinema::Agregar_Admin()
{
	string usuario, contra, subcontra;
	bool val = false;
	int cont = 0;
	do
	{
		cout << "_____Agregar_Adminstrador_____" << endl << endl <<
endl;
		cout << " Digite el nombre del administrador : ";
		cin.ignore();
		getline(cin, usuario);
		cout << " Digite la contraseña : ";
		getline(cin, contra);
		cout << " Digite de nuevo la contraseña : ";
		getline(cin, subcontra);
		system("cls");
		if (contra == subcontra)
		{
			val = true;
		}
		cont++;
		if (cont == 3)
			Menu_Admin();

	} while (val == false);
	fstream file("Administrador.csv", ios::out | ios::app);
```

```
	if (file.is_open())
	{
		file << usuario << ";" << contra << endl;
	}
	miLista->Insertar_Fin(usuario, contra);
}
void Cinema::Eliminar_Admin()
{
	string usuario;
	bool val = false;
	int cont = 0;
	do
	{
		cout << "_____Eliminar_Adminstrador_____" << endl << endl <<
endl;
		cout << " Nombre del administrador que quieres eliminar: ";
		cin.ignore();
		getline(cin, usuario);
		val = miLista->Buscar_Admin(usuario);
		system("cls");
		cont++;
		if (cont == 3)
			Menu_Admin();
	} while (val == false);
	miLista->Extrae_Inter(usuario);
	miLista->Reiniciar_BD();
}
void Cinema::LeeryCopiarMovie()
{
```

```
Califica* principio;
bool ok = false;
string peli = "", gene = "",calif = "";
float cal,fnum;
string caux;
int m = 0,num,pos = 0;
ifstream file;
file.open("List_Movies.csv");
string line;
int i = 0;
while (file.good())
{

	getline(file, line);//3 calif
	if (!(i == 0))
	{
		ID = i;
		ok = false;
		for (int n = 0; n < line.length(); n++)
		{

			calif.push_back(line[n]);
			if (line[n] == '.')
			{
				pos = n + 1;


			}
			if (pos == n)
			{
```

```
                        ok = true;
                }

        }
        if (ok == true)
        {
                for (int u = 0; u < calif.length(); u++)
                {
                        if (u == pos)
                        {
                                caux.push_back(calif[u]);
                        }
                }
        }


        string::size_type sz;
        cal = stof(calif, &sz);
        if (ok == true)
        {
                num = stoi(caux);
                fnum = (float)num / 10;
                cal = cal + fnum;
        }

}
if (peli != "" && gene != "" && calif != "")
{
```

```
                    miListMovies->Insertar_Movie(peli,gene,cal,ID);
                    miListMovies->Insertar_Alfa(peli, gene, cal, ID);
                    miArbolMovies->Crear_Nodo(peli, gene, cal, ID);
                    principio = miArbolMovies->Consultar_Raiz();
                    miArbolMovies->Insertar(principio);
                    peli = ""; gene = ""; calif = ""; caux = "";

            }
            getline(file, line, ';');//1 peli

            for (int n = 0; n < line.length(); n++)
            {
                    peli.push_back(line[n]);
            }
            getline(file, line, ';');//2 gene

            for (int n = 0; n < line.length(); n++)
            {
                    gene.push_back(line[n]);
            }
            i++;

    }
}
void Cinema::Edit_Movie()
{
    int opc, val = 0;
    string buscar;
```

```
do
{
	cout << "_____Editar Películas_____" << endl;
	Imprimir(Menu_Edit, Menu_Edit.size());
	cout << "Digite la opión deseada (1/2/3/4/5/6/7) : ";
	cin >> opc;
	system("cls");
	if (opc > 0 && opc < 8)
		val = 1;
} while (val == 0);
switch (opc)
{
case 1:
	miListMovies->Mostrar_Movie();
	system("pause()");
	system("cls");
	Edit_Movie();
	break;
case 2:
	Agregar_Movie();
	Edit_Movie();
	break;
case 3:
	Eliminar_Movie();
	Edit_Movie();
	break;
case 4: //Mostrar Funciones
	miListFunct->Mostrar_Sala_Hora();
	system("pause()");
```

```
                system("cls");
                Edit_Movie();
                break;
        case 5: //Agregar Funciones
                Agregar_Sala_Hora();
                Edit_Movie();
                break;
        case 6: //Eliminar Funciones
                Eliminar_Sala_Hora();
                Edit_Movie();
                break;
        default:
                Menu_Admin();
                break;
        }
}
void Cinema::Agregar_Movie()
{
        Califica* principio;
        string Peli, Gene;
        float Cali;

        cout << "_____Agregar_Película_____" << endl << endl << endl;
        cout << " Digite el nombre de la película : ";
        cin.ignore();
        getline(cin, Peli);
        cout << " Digite el genero de la película : ";
        getline(cin, Gene);
        cout << " Digite la calificación de la película : ";
```

```
	cin >> Cali;
	ID++;

	system("cls");


	fstream file("List_Movies.csv", ios::out | ios::app);
	if (file.is_open())
	{
		file << Peli << ";" << Gene <<";" <<  to_string(Cali) << ";" << to_string(ID) << endl;
	}
	miListMovies->Insertar_Movie(Peli, Gene,Cali,ID);
	miListMovies->Insertar_Alfa(Peli, Gene, Cali, ID);
	miArbolMovies->Crear_Nodo(Peli, Gene, Cali, ID);
	principio = miArbolMovies->Consultar_Raiz();
	miArbolMovies->Insertar(principio);
}
void Cinema::Eliminar_Movie()
{
	string val;
	int extpeli;
	Califica* principio;

	cout << "_____Eliminar_Película_____" << endl << endl << endl;
	miListMovies->Mostrar_Movie_y_Seleccionar();
	cout << "\n _____" << endl;
	cout << " Digite el número asignado a la película que quieres eliminar: ";
	cin >> extpeli;
	miListMovies->Extrae_Inter_Alfa(extpeli);
```

```
	principio = miArbolMovies->Consultar_Raiz();
	if (!principio)
		cout << "" ;
	else
		miArbolMovies->Extraer(NULL, principio, extpeli, "");

	val = miListMovies->Extrae_Inter_Movie(extpeli);
	system("cls");
	if (val != "")
	{
		cout << "Se elimino la película con el nombre de " << val << endl;

		miListMovies->Reiniciar_BD_Movie();
	}
	system("pause()");
	system("cls");


}
void Cinema::LeeryCopiarSalaHora()
{
	Hora* prin;
	string peli = "", sal = "", hrs = "", nomfun = "", ShrsI = "", ShrsF = "";
	int HrsI = 0, HrsF = 0,Sal = 0;
	bool ok = false, ok2 = false,ok3 = false;
	ifstream file;
	file.open("Funciones_Movies.csv");
	string line;
```

```
int i = 0;
while (file.good())
{

	getline(file, line);//6 NumFin
	if (!(i == 0))
	{
		ID2 = i;
		for (int n = 0; n < line.length(); n++)
		{
			ShrsF.push_back(line[n]);
			ok2 = true;
		}
		if (ok2 == true)
		{
			HrsF = stoi(ShrsF);
			ok2 = false;
		}

	}

	if (peli != "" && sal != "" && hrs != "" && nomfun != "" && ShrsI != "" && ShrsF != "")
	{


		miListFunct->Insertar_Sala_Hora(nomfun, Sal, hrs, peli, HrsI, HrsF,ID2);
		miArbolHorario->Crear_Nodo(nomfun, Sal, hrs, peli, HrsI, HrsF);
		prin = miArbolHorario->Consultar_Raiz();
```

```
            miArbolHorario->Insertar(prin);

            peli = ""; sal = ""; hrs = ""; nomfun = ""; ShrsI = ""; ShrsF = "";

}
getline(file, line, ';');//1 Funciones

for (int n = 0; n < line.length(); n++)
{
            nomfun.push_back(line[n]);
}
getline(file, line, ';');//2 Sala

for (int n = 0; n < line.length(); n++)
{
            sal.push_back(line[n]);
            ok3 = true;
}
if (ok3 == true)
{
            Sal = stoi(sal);
            ok3 = false;

}

getline(file, line, ';');//3 Hora

for (int n = 0; n < line.length(); n++)
{
```

```
                hrs.push_back(line[n]);
            }
            getline(file, line, ';');//4 Peli

            for (int n = 0; n < line.length(); n++)
            {
                peli.push_back(line[n]);
            }
            getline(file, line, ';');//5 NumHrsIni

            for (int n = 0; n < line.length(); n++)
            {
                ShrsI.push_back(line[n]);
                ok = true;
            }
            if (ok == true)
            {
                HrsI = stoi(ShrsI);
                ok = false;
            }

            i++;

        }
}
void Cinema::Agregar_Sala_Hora()
{
        Hora* prin;
        string peli , hrs , nomfun,aux ;
```

```
	int HrsI, HrsF,sala,fun;


	cout << "_____Agregar_Sala_y_Hora_____" << endl << endl << endl;
	cout << " Digite el nombre de la película : ";
	cin.ignore();
	getline(cin, peli);
	cout << " Digite el número de sala : ";
	cin >> sala;
	cout << " Digite la hora de inicio con (:) : ";
	getline(cin, hrs);
	cout << " Digite la hora de inicio número : ";//1800
	cin >> HrsI;
	cout << " Digite la hora de fin número : ";//1900
	cin >> HrsF;
	nomfun.push_back('f');
	fun = ID + rand() %1000;
	ID2++;
	aux =to_string(fun);
	for (int n = 0; n < aux.length(); n++)
	{
		nomfun.push_back(aux[n]);
	}
	system("cls");


	fstream file("Funciones_Movies.csv", ios::out | ios::app);
	if (file.is_open())
	{
		file << nomfun << ";" << to_string(sala) << ";" << hrs<< ";" << peli << ";" <<
to_string(HrsI) << ";" << to_string(HrsF) << endl;
```

```
        }
        miListFunct->Insertar_Sala_Hora(nomfun, sala, hrs, peli, HrsI, HrsF,ID2);
        miArbolHorario->Crear_Nodo(nomfun, sala, hrs, peli, HrsI, HrsF);
        prin = miArbolHorario->Consultar_Raiz();
        miArbolHorario->Insertar(prin);
}
void Cinema::Eliminar_Sala_Hora()
{
        Hora* prin;
        string hrs,val ;
        int sala;

        cout << "_____Eliminar_Sala_Hora_____" << endl << endl << endl;
        cout << " Número de la sala que se quiere eliminar : ";
        cin >> sala;
        cout << " Hora que se quiere eliminar con (:) : ";
        cin.ignore();
        getline(cin, hrs);
        /*prin = miArbolHorario->Consultar_Raiz();
        if (!prin)
                cout << "";
        else
                miArbolHorario->Extraer(NULL, prin, extpeli, "");*/



        val = miListFunct->Extrae_Inter(hrs, sala);
        system("cls");
        if (val != "")
```

```
		{
			cout << "Se elimino la película con el nombre de " << val << endl;
			miListFunct->Reiniciar_BD();
		}
		system("pause()");
		system("cls");

}
void Cinema::LeeryCopiarUsu()
{
		bool ok = false;
		string correo = "", usuario = "", contra = "",dinero = "";
		int cash;
		ifstream file;
		file.open("Usuario_Cinema.csv");
		string line;
		int i = 0;
		while (file.good())
		{

			getline(file, line);//4 Dinero
			if (!(i == 0))
			{
				for (int n = 0; n < line.length(); n++)
				{
					dinero.push_back(line[n]);

					ok = true;
				}
```

```
		if (ok == true)
		{
			cash = stoi(dinero);
			ok = false;
		}
}
if (correo != "" && usuario != ""&& contra != ""&& dinero != "")
{
		miListUsuario->Insertar_Fin(correo, usuario, contra, cash);

		usuario = ""; contra = ""; correo = ""; dinero = "";
}
getline(file, line, ';');//1 Correo

for (int n = 0; n < line.length(); n++)
{
		correo.push_back(line[n]);
}
getline(file, line, ';');//2  Nombre

for (int n = 0; n < line.length(); n++)
{
		usuario.push_back(line[n]);
}
getline(file, line, ';');//3 Contra

for (int n = 0; n < line.length(); n++)
{
		contra.push_back(line[n]);
```

```
                }

                i++;
        }
}
void Cinema::Menu_Usuario()
{
        int opc, val = 0;
        do
        {
                cout << "_____Menu del Usuario_____" << endl;
                Imprimir(Menu_Usu, Menu_Usu.size());
                cout << "Digite la opión deseada (1/2/3) : ";
                cin >> opc;
                system("cls");
                if (opc > 0 && opc < 4)
                        val = 1;
        } while (val == 0);
        switch (opc)
        {
        case 1:

                Iniciar_Sesion();
                break;
        case 2:
                Crear_Cuenta();
                Menu_Usuario();
                break;
        default:
```

```
                Menu();
                break;
        }
}
void Cinema::Iniciar_Sesion()
{
        string usuario, contra;
        bool val = false;
        int fail = 0;
        do
        {
                cout << "\n_______Iniciar_Sesion_______" << endl << endl;
                cout << "  Nombre de Usuario o Correo : ";
                cin.ignore();
                getline(cin, usuario);
                cout << "  Contraseña : ";
                getline(cin, contra);
                system("cls");
                val = miListUsuario->Buscar_Val_Usuario(usuario, contra);
                if (val == true)
                {
                        Menu_Prin(usuario);
                }
                if (val == false)
                {
                        fail += 1;
                }
                if (fail == 3)
                {
```

```
                cout << "Por motivos de seguridad seras regresado al menu." << endl;
                Menu();
            }
        } while (val == false);
}
void Cinema::Crear_Cuenta()
{
        string usuario, contra, subcontra,correo;
        bool val = false;
        int cont = 0,cash = 0;
        do
        {
            cout << "_____Crear_Cuenta_____" << endl << endl << endl;
            cout << " Digite su nombre de Usuario: ";
            cin.ignore();
            getline(cin, usuario);
            cout << " Digite su contraseña : ";
            getline(cin, contra);
            cout << " Digite de nuevo su contraseña : ";
            getline(cin, subcontra);

            if (contra == subcontra)
            {
                val = true;
            }
            if (val == false)
                system("cls");
            cont++;
            if (cont == 3)
```

```
                        Menu_Usuario();



        } while (val == false);
        cout << "\nDigite su correo electronico : ";
        getline(cin, correo);
        cout << "Digite la cantidad de dinero que quiere ingresar : ";
        cin >> cash;

        fstream file("Usuario_Cinema.csv", ios::out | ios::app);
        if (file.is_open())
        {
                file << correo << ";" << usuario << ";" << contra << ";" << to_string(cash) << endl;
        }
        miListUsuario->Insertar_Fin(correo, usuario, contra, cash);

}
void Cinema::LeeryCopiarArch()
{
        string nomfun = "";
        ifstream file;
        file.open("Archivo_Funcion.csv");
        string line;
        int i = 0;
        while (file.good())
        {

                getline(file, line);//2
                if (!(i == 0))
```

```
                {
                        for (int n = 0; n < line.length(); n++)
                        {
                                nomfun.push_back(line[n]);


                        }


                }
                if (nomfun != "")
                {

                        miListArch->Insertar_Fin(nomfun);
                        nomfun = "";
                }


                i++;
        }
}
void Cinema::Menu_Prin(string nomUsu)
{
        Califica* principio;
        Hora* prin;
        Usuario NUsu;
        int opc, val = 0,sum =0;
        do
        {
                cout << "_____Menu Principal_____" << endl;
                Imprimir(Menu_Pin, Menu_Pin.size());
                cout << "Digite la opión deseada (1/2/3/4/5/6/7/8) : ";
```

```
            cin >> opc;
            system("cls");
            if (opc > 0 && opc < 9)
                        val = 1;
} while (val == 0);
switch (opc)
{
case 1:  // Calif
            cout << "\n_________Películas_Ordenadas_por_Calificación__________" << endl;
            principio = miArbolMovies->Consultar_Raiz();
            if (!principio)
                        cout << "";
            else
               miArbolMovies->Recorrer_In_Orden(principio);

            system("pause()");
            system("cls");
            Menu_Prin(nomUsu);
            break;
case 2: //Horario
            cout << "\n_________Películas_Ordenadas_por_Horario__________" << endl;
            prin = miArbolHorario->Consultar_Raiz();
            if (!prin)
                        cout << "";
            else
                        miArbolHorario->Recorrer_In_Orden(prin);
            system("pause()");
            system("cls");
            Menu_Prin(nomUsu);
```

```
            break;
case 3: //Alfa
            miListMovies->Mostrar_Alfa();
            system("pause()");
            system("cls");
            Menu_Prin(nomUsu);
            break;
case 4: //Comprar Boleto
            Comprar_Boletos(nomUsu);
            Menu_Prin(nomUsu);
            break;
case 5: //Ingresa Money
            cout << "Digite la cantidad de dinero que quiere añadir a su cuenta : $";
            cin >> sum;
            miListUsuario->Sumar_Cash(nomUsu, sum);
            Menu_Prin(nomUsu);
            break;
case 6: //Estado de Cuenta

            NUsu = miListUsuario->Buscar_Info_Usuario(nomUsu);
            cout << "Su estado de cuenta actual es de : $" << NUsu.cash << endl;
            system("pause()");
            system("cls");
            Menu_Prin(nomUsu);
            break;
case 7: //Menu Usuario
            system("cls");
            Menu_Usuario();
```

```
                break;
        default: //Salir
                exit(0);
                break;
        }
}
void Cinema::CrearyCopiarFun(string nomfun)
{
        string A1 = "", A2 = "", A3 = "", A4 = "", A5 = "", A6 ="", A7 ="", A8 = "";
        string file;
        file = nomfun;
        file.append(".csv");
        ifstream fin;
        ofstream fout;
        fin.open("Asientos_Base.csv", ios::in);
        string line;
        int i = 0;
        fout.open(file, ios::out);
        while (fin.good())
        {

                if (i == 0)
                {
                        if (fout.is_open())
                        {
                                fout << endl;
                        }
                }
                getline(fin, line);//6 NumFin
```

```
if (!(i == 0))
{
	for (int n = 0; n < line.length(); n++)
	{
		A8.push_back(line[n]);

	}

}

if (A1 != "", A2 != "", A3 != "", A4 != "", A5 != "", A6 != "", A7 != "", A8 != "")
{
	if (fout.is_open())
	{
		fout << A1 << ";" << A2 << ";" << A3 << ";" << A4 << ";" << A5 << ";"
<< A6 << ";" << A7 << ";" << A8 << endl;

	}
	miAsientos->Insertar_Asientos(A1, A2, A3, A4, A5, A6, A7, A8);

	A1 = ""; A2 = ""; A3 = ""; A4 = ""; A5 = ""; A6 = ""; A7 = ""; A8 = "";

}
getline(fin, line, ';');//1 A

for (int n = 0; n < line.length(); n++)
{
    A1.push_back(line[n]);
}
```

```
getline(fin, line, ';');//2 A

for (int n = 0; n < line.length(); n++)
{
	A2.push_back(line[n]);

}

getline(fin, line, ';');//3 A

for (int n = 0; n < line.length(); n++)
{
	A3.push_back(line[n]);
}
getline(fin, line, ';');//4 A

for (int n = 0; n < line.length(); n++)
{
	A4.push_back(line[n]);
}
getline(fin, line, ';');//5 A

for (int n = 0; n < line.length(); n++)
{
	A5.push_back(line[n]);

}
getline(fin, line, ';');//6 A
```

```
		for (int n = 0; n < line.length(); n++)
		{
			A6.push_back(line[n]);

		}
		getline(fin, line, ';');//7 A

		for (int n = 0; n < line.length(); n++)
		{
			A7.push_back(line[n]);

		}
		i++;

	}
	fin.close();
}
void Cinema::LeeryCopiarFun(string nomfun)
{
	string A1 = "", A2 = "", A3 = "", A4 = "", A5 = "", A6 = "", A7 = "", A8 = "";
	string file;
	file = nomfun;
	file.append(".csv");
	ifstream fin;
	ofstream fout;
	fin.open(file, ios::in);
	string line;
	int i = 0;
	while (fin.good())
```

```
{

        getline(fin, line);//6 NumFin
        if (!(i == 0))
        {
                for (int n = 0; n < line.length(); n++)
                {
                        A8.push_back(line[n]);


                }


        }

        if (A1 != "", A2 != "", A3 != "", A4 != "", A5 != "", A6 != "", A7 != "", A8 != "")
        {

                miAsientos->Insertar_Asientos(A1, A2, A3, A4, A5, A6, A7, A8);

                A1 = ""; A2 = ""; A3 = ""; A4 = ""; A5 = ""; A6 = ""; A7 = ""; A8 = "";


        }
        getline(fin, line, ';');//1 A

        for (int n = 0; n < line.length(); n++)
        {
                A1.push_back(line[n]);
        }
        getline(fin, line, ';');//2 A
```

```
for (int n = 0; n < line.length(); n++)
{
	A2.push_back(line[n]);

}

getline(fin, line, ';');//3 A

for (int n = 0; n < line.length(); n++)
{
	A3.push_back(line[n]);
}
getline(fin, line, ';');//4 A

for (int n = 0; n < line.length(); n++)
{
	A4.push_back(line[n]);
}
getline(fin, line, ';');//5 A

for (int n = 0; n < line.length(); n++)
{
	A5.push_back(line[n]);

}
getline(fin, line, ';');//6 A

for (int n = 0; n < line.length(); n++)
{
```

```
                A6.push_back(line[n]);


            }
            getline(fin, line, ';');//7 A

            for (int n = 0; n < line.length(); n++)
            {
                A7.push_back(line[n]);


            }
            i++;

        }
        fin.close();
}
void Cinema::Comprar_Boletos(string Usu)
{
        bool compra;
        bool encontrado;
        Movie Peli;
        Funcion Fun;
        Usuario NUsu;
        int num, hora;
        string peli;
        string asiento;
        string cambio;
        int precio = 78,codi;
        codi = 21998 + rand() % 1000;
        NUsu = miListUsuario->Buscar_Info_Usuario(Usu);
```

```
if (NUsu.cash >= precio)
{

miListMovies->Mostrar_Por_ID();
cout << "\nDigite el número asignado a la película seleccionada : ";
cin >> num;
system("cls");
Peli = miListMovies->Buscar_Movie(num);
if (Peli.peli != "")
{
        miListFunct->Mostrar_Por_ID(Peli.peli);
        cout << "\nDigite el número asignado a la hora seleccionada : ";
        cin >> hora;
        system("cls");
        Fun = miListFunct->Buscar_Movie(hora);
        encontrado = miListArch->Buscar_Funcion(Fun.nomfun);
        if (encontrado == true)
        {
                //Abrir
                LeeryCopiarFun(Fun.nomfun);
                miAsientos->Mostrar_Asientos();
                cout << "\nDigite el nombre del asiento (A1) : ";
                cin >> asiento;
                cambio = miAsientos->Cambiar_Asientos(asiento);
                if (cambio != "")
                {
                        //dinero
                        compra = miListUsuario->Cambiar_Cash(Usu, precio);
                        if (compra == true)
```

```
                                        {
                                                Enviar_Mail(codi, NUsu.correo, NUsu.nom, Fun.hrs,
Fun.sal, asiento,Peli.peli);
                                                system("cls");
                                        }


                                }
                                else
                                {
                                        Menu_Prin(Usu);
                                }
                        }
                        else
                        {
                                if (Fun.nomfun != "")
                                {
                                        //Crear
                                        miListArch->Insertar_Fin(Fun.nomfun);
                                   CrearyCopiarFun(Fun.nomfun);
                                        miAsientos->Mostrar_Asientos();
                                        cout << "\nDigite el nombre del asiento (A1) : ";
                                        cin >> asiento;
                                        cambio = miAsientos->Cambiar_Asientos(asiento);
                                        if (cambio != "")
                                        {
                                                //dinero
                                                compra = miListUsuario->Cambiar_Cash(Usu, precio);
                                                if (compra == true)
                                                {
```

```
                                        Enviar_Mail(codi, NUsu.correo, NUsu.nom,
Fun.hrs, Fun.sal, asiento,Peli.peli);
                                        system("cls");
                                }
                        }
                        else
                        {
                                Menu_Prin(Usu);
                        }
                }
                else
                {
                        Menu_Prin(Usu);
                }
        }
        else
        {
                Menu_Prin(Usu);
        }
}
else
{
                cout << "No cuenta con saldo suficiente para realizar la compra." << endl;
                system("pause()");
                system("cls");
                Menu_Prin(Usu);
}
}
```

```
void Cinema::Enviar_Mail(int codi, string correo, string nom, string hrs, int sala, string
asiento,string peli)

{
	::CoInitialize(NULL);


	IMailPtr oSmtp = NULL;

	oSmtp.CreateInstance(__uuidof(EASendMailObjLib::Mail));

	oSmtp->LicenseCode = _T("TryIt");


	// Set your gmail email address

	oSmtp->FromAddr = _T("cinema_service@outlook.com");


	// Add recipient email address

	_bstr_t Corr(correo.c_str());

	oSmtp->AddRecipientEx(_T(Corr), 0);


	string Code = to_string(codi);

	_bstr_t Codi(Code.c_str());

	_bstr_t Nom(nom.c_str());

	_bstr_t Hrs(hrs.c_str());

	string Sal = to_string(sala);

	_bstr_t Sala(Sal.c_str());

	_bstr_t Asien(asiento.c_str());

	_bstr_t Peli(peli.c_str());


	string Base = "  Hola ";

	_bstr_t Encab(Base.c_str());

	Encab += _T(Nom);

	Encab += _T(" ,te compartimos la información de tu boleto.   \n   Codigo : ");
```

```
        Encab += _T(Codi);
        Encab += _T("\n  Película : ");
        Encab += _T(Peli);
        Encab += _T("\n  Hora : ");
        Encab += _T(Hrs);
        Encab += _T("\n  Sala : ");
        Encab += _T(Sala);
        Encab += _T("\n  Asiento : ");
        Encab += _T(Asien);

        // Set email subject
        oSmtp->Subject = _T("Boleto Cinema");

        // Set email body
        oSmtp->BodyText = _T(Encab);

        // Gmail SMTP server address
        oSmtp->ServerAddr = _T("smtp.office365.com");//("smtp.gmail.com");

        // Gmail user authentication should use your
        // Gmail email address as the user name.
        // For example: your email is "gmailid@gmail.com", then the user should be
"gmailid@gmail.com"
        oSmtp->UserName = _T("cinema_service@outlook.com");//("carlosac2198@gmail.com");
        oSmtp->Password = _T("cinema4321");

        // If you want to use direct SSL 465 port,
        // Please add this line, otherwise TLS will be used.
        // oSmtp->ServerPort = 465;
```

```
	// Set 25 or 587 SMTP port
	oSmtp->ServerPort = 587;

	// detect SSL/TLS automatically
	oSmtp->ConnectType = ConnectSSLAuto;

	_tprintf(_T("Start to send email via gmail account ...\r\n"));

	if (oSmtp->SendMail() == 0)
	{
		_tprintf(_T("email was sent successfully!\r\n"));
	}
	else
	{
		_tprintf(_T("failed to send email with the following error: %s\r\n"),
			(const TCHAR*)oSmtp->GetLastErrDescription());
	}
}
```

```
#include "Lista.h"
Lista::Lista()
{
	setlocale(LC_ALL, "");
	cabecera = NULL;
	fin = NULL;
}
Lista::~Lista() {}
void Lista::Mostrar_Admin()
{
	if (!cabecera) //cabecera == NULL
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return;
	}
	nodo = cabecera;
	cout << "\n LISTA DE ADMINISTRADORES: " << endl;
	while (nodo)//nodo != NULL
```

```
	{
		cout << "\t" << nodo->nom;
		cout << endl;
		nodo = nodo->sig;
	}
}

void Lista::Insertar_Fin(string nuevo,string con)
{
	nodo = new Administrador;   //Paso 1
	nodo->nom = nuevo;    //Paso 2
	nodo->contra = con;
	nodo->sig = NULL;     //Paso 3
	if (fin)              //fin != NULL
		fin->sig = nodo;
	if (!cabecera)   //cabecera == NULL
		cabecera = nodo;
	fin = nodo;           //Paso 4
	//cout << "\tOK.Se insertó " << nuevo << endl;
}
string Lista::Extrae_Prin()
{
	string extraido = "";
	if (!cabecera) //cabecera == NULL
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return extraido;
	}
	nodo = cabecera;
	extraido = nodo->nom;
	cabecera = nodo->sig;
	delete nodo;
	return extraido;

}
string Lista::Extrae_Inter(string ext)
{
	string extraido;
	Administrador* anterior;
	bool bandera = false;

	if (!cabecera)
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return extraido;
	}

	anterior = NULL;
	nodo = cabecera;

	while (nodo != NULL)
	{
		if (nodo->nom == ext)
		{
			bandera = true;
			break;
		}
		anterior = nodo;
```

```
			nodo = nodo->sig;
		}
		if (!bandera)
		{
			cout << "\nNo Existe " << ext << endl;
			return extraido;
		}
		extraido = nodo->nom;
		if (!anterior)
			cabecera = nodo->sig;
		else
			anterior->sig = nodo->sig;

		if (nodo == fin)
			fin = anterior;

		delete nodo;
		return extraido;
}
string Lista::Extrae_Fin()
{
		Administrador* anterior;
		string extraido = "";
		if (!cabecera)
		{
			cout << "\n\t La lista está vacía." << endl;
			return extraido;
		}

		anterior = NULL;
		nodo = cabecera;
		while (nodo->sig != NULL) //  Paso 1
		{
			anterior = nodo;
			nodo = nodo->sig;
		}
		extraido = nodo->nom; //  Paso 2

		if (!anterior)
		{
			cabecera = NULL;
		}
		else
		{
			anterior->sig = NULL; //  Paso 3
		}
		fin = anterior; //  Paso 4
		delete nodo; // Paso 5
		return extraido;

}
bool Lista::Buscar_Val_Admin(string usu, string con)
{
		bool encontrado = false;
		if (!cabecera) //cabecera == NULL
		{
			 encontrado = false;
		}
```

```
	nodo = cabecera;
	while (nodo)//nodo != NULL
	{

		if (nodo->nom == usu && nodo->contra == con)
		{
			encontrado  = true;
			break;
		}
		nodo = nodo->sig;
	}
	return encontrado;
}
bool Lista::Buscar_Admin(string usu)
{
	bool encontrado = false;
	if (!cabecera) //cabecera == NULL
	{
		encontrado = false;
	}
	nodo = cabecera;
	while (nodo)//nodo != NULL
	{

		if (nodo->nom == usu)
		{
			encontrado = true;
			break;
		}
		nodo = nodo->sig;
	}
	return encontrado;
}
void Lista::Reiniciar_BD()
{
	ofstream file("Administrador.csv");
	nodo = cabecera;
	if (file.is_open())
	{
		file << "Usuario" << ";" << "Contraseña" << endl;
		while (nodo)//nodo != NULL
		{
			file << nodo->nom << ";" << nodo->contra << endl;
			nodo = nodo->sig;
		}
	}
}



#include "ListArch.h"
ListArch::ListArch()
{
	setlocale(LC_ALL, "");
	cabecera = NULL;
	fin = NULL;
}
ListArch::~ListArch() {}
```

```
void ListArch::Mostrar_Archivo()
{
	if (!cabecera) //cabecera == NULL
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return;
	}
	nodo = cabecera;
	cout << "\n LISTA DE ADMINISTRADORES: " << endl;
	while (nodo)//nodo != NULL
	{
		cout << "\t" << nodo->nomfun;
		cout << endl;
		nodo = nodo->sig;
	}
}

void ListArch::Insertar_Fin(string nuevo)
{
	nodo = new Archivo;	//Paso 1
	nodo->nomfun = nuevo;	//Paso 2
	nodo->sig = NULL;	//Paso 3
	if (fin)		//fin != NULL
		fin->sig = nodo;
	if (!cabecera)	//cabecera == NULL
		cabecera = nodo;
	fin = nodo;		//Paso 4
	//cout << "\tOK.Se insertó " << nuevo << endl;
}
string ListArch::Extrae_Prin()
{
	string extraido = "";
	if (!cabecera) //cabecera == NULL
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return extraido;
	}
	nodo = cabecera;
	extraido = nodo->nomfun;
	cabecera = nodo->sig;
	delete nodo;
	return extraido;

}
string ListArch::Extrae_Inter(string ext)
{
	string extraido;
	Archivo* anterior;
	bool bandera = false;

	if (!cabecera)
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return extraido;
	}

	anterior = NULL;
	nodo = cabecera;
```

```
	while (nodo != NULL)
	{
		if (nodo->nomfun == ext)
		{
			bandera = true;
			break;
		}
		anterior = nodo;
		nodo = nodo->sig;
	}
	if (!bandera)
	{
		cout << "\nNo Existe " << ext << endl;
		return extraido;
	}
	extraido = nodo->nomfun;
	if (!anterior)
		cabecera = nodo->sig;
	else
		anterior->sig = nodo->sig;

	if (nodo == fin)
		fin = anterior;

	delete nodo;
	return extraido;
}
string ListArch::Extrae_Fin()
{
	Archivo* anterior;
	string extraido = "";
	if (!cabecera)
	{
		cout << "\n\t La lista está vacía." << endl;
		return extraido;
	}

	anterior = NULL;
	nodo = cabecera;
	while (nodo->sig != NULL) //  Paso 1
	{
		anterior = nodo;
		nodo = nodo->sig;
	}
	extraido = nodo->nomfun; //  Paso 2

	if (!anterior)
	{
		cabecera = NULL;
	}
	else
	{
		anterior->sig = NULL; //  Paso 3
	}
	fin = anterior; //  Paso 4
	delete nodo; // Paso 5
	return extraido;
```

```
}
bool ListArch::Buscar_Archivo(string usu)
{
	bool encontrado = false;
	if (!cabecera) //cabecera == NULL
	{
		encontrado = false;
	}
	nodo = cabecera;
	while (nodo)//nodo != NULL
	{

		if (nodo->nomfun == usu)
		{
			encontrado = true;
			break;
		}
		nodo = nodo->sig;
	}
	return encontrado;
}
void ListArch::Reiniciar_BD()
{
	ofstream file("Archivo_Funcion.csv");
	nodo = cabecera;
	if (file.is_open())
	{
		file << "Función" << endl;
		while (nodo)//nodo != NULL
		{
			file << nodo->nomfun << endl;
			nodo = nodo->sig;
		}
	}
}
bool ListArch::Buscar_Funcion(string usu)
{
	bool encontrado = false;
	if (!cabecera) //cabecera == NULL
	{
		encontrado = false;
	}
	nodo = cabecera;
	while (nodo)//nodo != NULL
	{

		if (nodo->nomfun == usu)
		{
			encontrado = true;
			break;
		}
		nodo = nodo->sig;
	}
	return encontrado;
}
```

```
#include "ListFunct.h"
ListFunct::ListFunct()
{
	setlocale(LC_ALL, "");
	cabecera = NULL;
	fin = NULL;
}
ListFunct::~ListFunct() {}
void ListFunct::Mostrar_Por_ID(string pel)
{
	if (!cabecera) //cabecera == NULL
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return;
	}
	nodo = cabecera;
	cout << "\n  " << endl;
	cout << "\n     |Número| Hora |              Película                    |"
<< endl;
	while (nodo)//nodo != NULL
	{
		if (pel == nodo->peli)
		{
			cout << "\t" << nodo->ID << "\t" << nodo->hrs << "\t" << nodo-
>peli;
			cout << endl;
		}
		nodo = nodo->sig;
	}
}
void ListFunct::Mostrar_Sala_Hora()
{
	if (!cabecera) //cabecera == NULL
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return;
	}
	nodo = cabecera;
	cout << "\n  " << endl;
	cout << "\n     |Función|   Sala  | Hora |Hora Num|Hora Fin|
Película                    |" << endl;
	while (nodo)//nodo != NULL
	{
		cout << "\t" << nodo->nomfun << "\tsala " << nodo->sal << "\t" <<
nodo->hrs << "\t" <<  nodo->HrsI << "\t" << nodo->HrsF << "\t" <<nodo->peli ;
		cout << endl;
		nodo = nodo->sig;
	}
}

void ListFunct::Insertar_Sala_Hora(string nfun, int nsal, string nhrs, string npeli,
int NHrsI, int NHrsF,int idx)
{
	nodo = new Funcion;   //Paso 1
	nodo->nomfun = nfun;  //Paso 2
	nodo->sal = nsal;
	nodo->hrs = nhrs;
	nodo->peli = npeli;
```

```
	nodo->HrsI = NHrsI;
	nodo->HrsF = NHrsF;
	nodo->ID = idx;
	nodo->sig = NULL;	//Paso 3
	if (fin)		//fin != NULL
		fin->sig = nodo;
	if (!cabecera)   //cabecera == NULL
		cabecera = nodo;
	fin = nodo;		//Paso 4
	//cout << "\tOK.Se insertó " << nuevo << endl;
}
string ListFunct::Extrae_Prin()
{
	string extraido = "";
	if (!cabecera) //cabecera == NULL
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return extraido;
	}
	nodo = cabecera;
	extraido = nodo->peli;
	cabecera = nodo->sig;
	delete nodo;
	return extraido;

}
string ListFunct::Extrae_Inter(string ext,int xsal)
{
	string extraido;
	Funcion* anterior;
	bool bandera = false;

	if (!cabecera)
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return extraido;
	}

	anterior = NULL;
	nodo = cabecera;

	while (nodo != NULL)
	{
		if (nodo->hrs == ext && nodo->sal == xsal)
		{
			bandera = true;
			break;
		}
		anterior = nodo;
		nodo = nodo->sig;
	}
	if (!bandera)
	{
		cout << "\nNo Existe " << ext << endl;
		return extraido;
	}
	extraido = nodo->peli;
	if (!anterior)
```

```
                cabecera = nodo->sig;
        else
                anterior->sig = nodo->sig;

        if (nodo == fin)
                fin = anterior;

        delete nodo;
        return extraido;
}
string ListFunct::Extrae_Fin()
{
        Funcion* anterior;
        string extraido = "";
        if (!cabecera)
        {
                cout << "\n\t La lista está vacía." << endl;
                return extraido;
        }

        anterior = NULL;
        nodo = cabecera;
        while (nodo->sig != NULL) //  Paso 1
        {
                anterior = nodo;
                nodo = nodo->sig;
        }
        extraido = nodo->peli; //  Paso 2

        if (!anterior)
        {
                cabecera = NULL;
        }
        else
        {
                anterior->sig = NULL; //  Paso 3
        }
        fin = anterior; //  Paso 4
        delete nodo; // Paso 5
        return extraido;

}


void ListFunct::Reiniciar_BD()
{

        ofstream file("Funciones_Movies.csv");
        nodo = cabecera;
        if (file.is_open())
        {
                file << "Función" << ";" << "Sala" << ";" << "Hora" << ";" <<
"Película" << ";" << "Hora en Número" << ";" << "Hora de Fin" << endl;
                while (nodo)//nodo != NULL
                {
                        file << nodo->nomfun << ";" << to_string(nodo->sal) << ";" <<
nodo->hrs << ";" << nodo->peli << ";" << to_string(nodo->HrsI) << ";" <<
to_string(nodo->HrsF) << endl;
```

```
                        nodo = nodo->sig;
                }
        }
}
Funcion ListFunct::Buscar_Movie(int idx)
{
        Funcion Encont;

        bool encontrado = false;
        if (!cabecera) //cabecera == NULL
        {
                Encont.peli = "";
                Encont.nomfun = "";
                return Encont;
        }
        nodo = cabecera;
        while (nodo)//nodo != NULL
        {

                if (nodo->ID == idx)
                {
                        Encont.peli = nodo->peli;
                        Encont.hrs = nodo->hrs;
                        Encont.sal = nodo->sal;
                        Encont.nomfun = nodo->nomfun;
                        encontrado = true;
                        break;
                }
                nodo = nodo->sig;

        }
        return Encont;
}


#include "ListMovies.h"
ListMovies::ListMovies()
{
        setlocale(LC_ALL, "");
        cabecera = NULL;
        fin = NULL;
}
ListMovies::~ListMovies() {}
void ListMovies::Mostrar_Por_ID()
{
        if (!cabecera) //cabecera == NULL
        {
                cout << "\n\t La lista está vacía ** \n" << endl;
                return;
        }
        nodo = cabecera;
        cout << "\n| Número |              Película                    " << endl;
        while (nodo)//nodo != NULL
        {

                cout << "| " << nodo->id << "  |\t" << nodo->peli;
```

```
			cout << endl;
			nodo = nodo->sig;
		}
}
void ListMovies::Mostrar_Movie()
{

		int n = 0;
		string ch = "                                          ", gran = " ", med = "
";
		if (!cabecera) //cabecera == NULL
		{
			cout << "\n\t La lista está vacía ** \n" << endl;
			return;
		}
		nodo = cabecera;
		cout << "\n|              Película                    |    Genero    |
Calificación | " << endl;
		while (nodo)//nodo != NULL
		{
			n = nodo->peli.length();
			if (n < 7)
			{
				cout << "|" << nodo->peli <<ch <<"|" << nodo->gene << "\t|" <<
nodo->cali << "|";
			}
			if (n > 7 && n < 28)
			{
				cout << "|" << nodo->peli << med << "|" << nodo->gene << "\t|"
<< nodo->cali << "|";
			}
			if (n >28)
			{
				cout << "|" << nodo->peli << gran << "|" << nodo->gene << "\t|"
<< nodo->cali << "|";
			}
			cout << endl;
			nodo = nodo->sig;
		}
}
void ListMovies::Mostrar_Movie_y_Seleccionar()
{

		if (!cabecera) //cabecera == NULL
		{
			cout << "\n\t La lista está vacía ** \n" << endl;
			return;
		}
		nodo = cabecera;
		cout << "\n|    Número    |              Película                    |" <<
endl;
		while (nodo)//nodo != NULL
		{

			cout <<"|      "<<nodo->id<< "        |" << nodo->peli  << "|" ;

			cout << endl;
			nodo = nodo->sig;
```

```
	}
}
void ListMovies::Mostrar_Alfa()
{
	int n = 0;
	string ch = "                                          ", gran = " ", med = "
";
	if (!cabealfa) //cabecera == NULL
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return;
	}
	nodoalfa = cabealfa;
	cout << "\n Películas Ordenadas Alfabeticamente : " << endl;
	while (nodoalfa)//nodo != NULL
	{
		n = nodo->peli.length();
		if (n < 7)
		{
			cout << "|" << nodoalfa->peli << ch << "|" << nodoalfa->gene <<
" |";
		}
		if (n > 7 && n < 28)
		{
			cout << "|" << nodoalfa->peli << med << "|" << nodoalfa->gene <<
"|";
		}
		if (n > 28)
		{
			cout << "|" << nodoalfa->peli << gran << "|" << nodoalfa->gene
<< "|";
		}
		cout << endl;
		nodoalfa = nodoalfa->sigalf;
	}
}
void ListMovies::Insertar_Movie(string pel, string gen,float cal,int idx)
{

	nodo = new Movie;   //Paso 1
	nodo->peli = pel;    //Paso 2
	nodo->gene = gen;
	nodo->cali = cal;
	nodo->id = idx;
	nodo->sig = NULL;     //Paso 3
	if (fin)              //fin != NULL
		fin->sig = nodo;
	if (!cabecera)   //cabecera == NULL
		cabecera = nodo;
	fin = nodo;           //Paso 4
	//cout << "\tOK.Se insertó " << nuevo << endl;
}
void ListMovies::Insertar_Alfa(string pel, string gen, float cal, int idx)
{
	nodoalfa = new Movie;
	nodoalfa->peli = pel;    //Paso 2
	nodoalfa->gene = gen;
	nodoalfa->cali = cal;
```

```
	nodoalfa->id = idx;
	if (!cabealfa)
	{
		nodo->sigalf = NULL;
		cabealfa = nodo;
		finalfa = nodo;
		return;
	}
	aux = cabealfa;
	ante = NULL;
	do
	{
		if (nodoalfa->peli < aux->peli)
		{
			nodoalfa->sigalf = aux;
			if (ante)
			{
				ante->sigalf = nodoalfa;
			}
			else
			{
				cabealfa = nodoalfa;
			}
			return;
		}
		ante = aux;
		aux = aux->sigalf;
	} while (aux);
	nodoalfa->sigalf = NULL;
	finalfa->sigalf = nodoalfa;
	finalfa = nodoalfa;
}
string ListMovies::Extrae_Prin_Movie()
{
	string extraido = "";
	if (!cabecera) //cabecera == NULL
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return extraido;
	}
	nodo = cabecera;
	extraido = nodo->peli;
	cabecera = nodo->sig;
	delete nodo;
	return extraido;

}
string ListMovies::Extrae_Inter_Movie(int idx)
{
	string extraido;
	Movie* anterior;
	bool bandera = false;

	if (!cabecera)
	{
		cout << "\n\t La lista está vacía ** \n" << endl;
		return extraido;
	}
```

```
	anterior = NULL;
	nodo = cabecera;

	while (nodo != NULL)
	{
		if (nodo->id =  idx)
		{
			bandera = true;
			break;
		}
		anterior = nodo;
		nodo = nodo->sig;
	}
	if (!bandera)
	{
		cout << "\nNo Existe la opción " << idx << endl;
		return extraido;
	}
	extraido = nodo->peli;
	if (!anterior)
		cabecera = nodo->sig;
	else
		anterior->sig = nodo->sig;

	if (nodo == fin)
		fin = anterior;

	delete nodo;
	return extraido;
}
string ListMovies::Extrae_Fin_Movie()
{
	Movie* anterior;
	string extraido = "";
	if (!cabecera)
	{
		cout << "\n\t La lista está vacía." << endl;
		return extraido;
	}

	anterior = NULL;
	nodo = cabecera;
	while (nodo->sig != NULL) //  Paso 1
	{
		anterior = nodo;
		nodo = nodo->sig;
	}
	extraido = nodo->peli; //  Paso 2

	if (!anterior)
	{
		cabecera = NULL;
	}
	else
	{
		anterior->sig = NULL; //  Paso 3
	}
```

```
        fin = anterior; //  Paso 4
        delete nodo; // Paso 5
        return extraido;

}

string ListMovies::Extrae_Prin_Alfa()
{
        string extraido = "";
        if (!cabealfa) //cabecera == NULL
        {
                cout << "\n\t La lista está vacía ** \n" << endl;
                return extraido;
        }
        nodoalfa = cabealfa;
        extraido = nodoalfa->peli;
        cabecera = nodoalfa->sig;
        delete nodoalfa;
        return extraido;
}
string ListMovies::Extrae_Inter_Alfa(int idx)
{
        string extraido;
        Movie* anterior;
        bool bandera = false;

        if (!cabealfa)
        {
                cout << "\n\t La lista está vacía ** \n" << endl;
                return extraido;
        }

        anterior = NULL;
        nodoalfa = cabealfa;

        while (nodoalfa != NULL)
        {
                if (nodoalfa->id = idx)
                {
                        bandera = true;
                        break;
                }
                anterior = nodoalfa;
                nodoalfa = nodoalfa->sigalf;
        }
        if (!bandera)
        {
                cout << "\nNo Existe la opción " << idx << endl;
                return extraido;
        }
        extraido = nodoalfa->peli;
        if (!anterior)
                cabealfa = nodoalfa->sigalf;
        else
                anterior->sigalf = nodoalfa->sigalf;

        if (nodoalfa == finalfa)
                finalfa = anterior;
```

```
	delete nodoalfa;
	return extraido;
}
string ListMovies::Extrae_Fin_Alfa()
{
	Movie* anterior;
	string extraido = "";
	if (!cabealfa)
	{
		cout << "\n\t La lista está vacía." << endl;
		return extraido;
	}

	anterior = NULL;
	nodoalfa = cabealfa;
	while (nodoalfa->sigalf != NULL) //  Paso 1
	{
		anterior = nodoalfa;
		nodoalfa = nodoalfa->sigalf;
	}
	extraido = nodoalfa->peli; //  Paso 2

	if (!anterior)
	{
		cabealfa = NULL;
	}
	else
	{
		anterior->sigalf = NULL; //  Paso 3
	}
	finalfa = anterior; //  Paso 4
	delete nodoalfa; // Paso 5
	return extraido;
}

Movie ListMovies::Buscar_Movie(int idx)
{
	Movie Encont;

	bool encontrado = false;
	if (!cabecera) //cabecera == NULL
	{
		Encont.peli = "";
		return Encont;
	}
	nodo = cabecera;
	while (nodo)//nodo != NULL
	{

		if (nodo->id == idx)
		{
			Encont.peli = nodo->peli;
			Encont.gene = nodo->gene;
			Encont.cali = nodo->cali;
			encontrado = true;
			break;
		}
```

```
			nodo = nodo->sig;

		}
		return Encont;
}
void ListMovies::Reiniciar_BD_Movie()
{
		ofstream file("List_Movies.csv");
		nodo = cabecera;
		if (file.is_open())
		{
			file << "Película" << ";" << "Genero" << ";" << "Calificación" << endl;
			while (nodo)//nodo != NULL
			{
				file << nodo->peli << ";" << nodo->gene << ";" << nodo->cali <<
endl;
				nodo = nodo->sig;
			}
		}
}



#include "ListUsuario.h"
ListUsuario::ListUsuario()
{
		setlocale(LC_ALL, "");
		cabecera = NULL;
		fin = NULL;
}
ListUsuario::~ListUsuario() {}
void ListUsuario::Mostrar_Usuarios()
{
		if (!cabecera) //cabecera == NULL
		{
			cout << "\n\t La lista está vacía ** \n" << endl;
			return;
		}
		nodo = cabecera;
		cout << "\n LISTA DE USUARIOS: " << endl;
		while (nodo)//nodo != NULL
		{
			cout << "\t" << nodo->correo << "\t" << nodo->nom << "\t" << nodo-
>contra << "\t" << nodo->cash;
			cout << endl;
			nodo = nodo->sig;
		}
}

void ListUsuario::Insertar_Fin(string xcorr, string xnom, string xcontra, int xcash)
{
		nodo = new Usuario;   //Paso 1
		nodo->correo = xcorr;
		nodo->nom = xnom;    //Paso 2
		nodo->contra = xcontra;
		nodo->cash = xcash;
		nodo->sig = NULL;     //Paso 3
		if (fin)              //fin != NULL
```

```
			fin->sig = nodo;
		if (!cabecera)   //cabecera == NULL
			cabecera = nodo;
		fin = nodo;			//Paso 4
		//cout << "\tOK.Se insertó " << nuevo << endl;
}
string ListUsuario::Extrae_Prin()
{
		string extraido = "";
		if (!cabecera) //cabecera == NULL
		{
			cout << "\n\t La lista está vacía ** \n" << endl;
			return extraido;
		}
		nodo = cabecera;
		extraido = nodo->nom;
		cabecera = nodo->sig;
		delete nodo;
		return extraido;

}
string ListUsuario::Extrae_Inter(string ext)
{
		string extraido;
		Usuario* anterior;
		bool bandera = false;

		if (!cabecera)
		{
			cout << "\n\t La lista está vacía ** \n" << endl;
			return extraido;
		}

		anterior = NULL;
		nodo = cabecera;

		while (nodo != NULL)
		{
			if (nodo->nom == ext)
			{
				bandera = true;
				break;
			}
			anterior = nodo;
			nodo = nodo->sig;
		}
		if (!bandera)
		{
			cout << "\nNo Existe " << ext << endl;
			return extraido;
		}
		extraido = nodo->nom;
		if (!anterior)
			cabecera = nodo->sig;
		else
			anterior->sig = nodo->sig;

		if (nodo == fin)
```

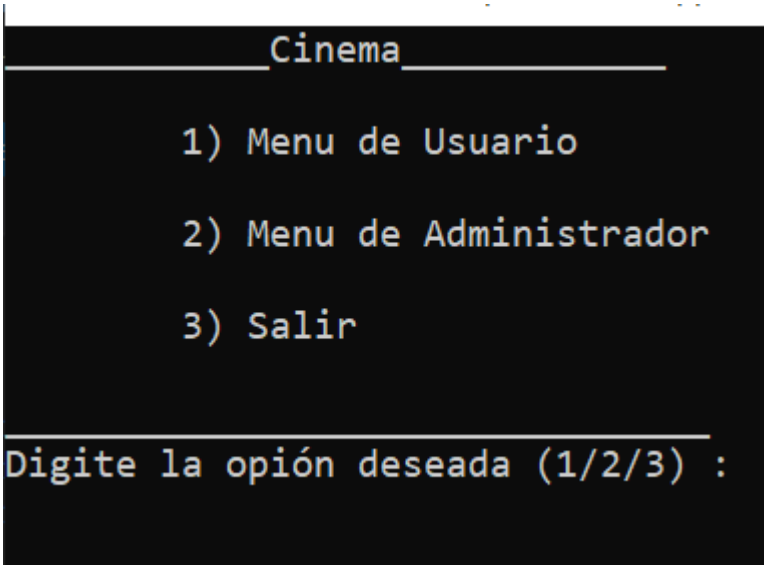
```
                fin = anterior;

        delete nodo;
        return extraido;
}
string ListUsuario::Extrae_Fin()
{
        Usuario* anterior;
        string extraido = "";
        if (!cabecera)
        {
                cout << "\n\t La lista está vacía." << endl;
                return extraido;
        }

        anterior = NULL;
        nodo = cabecera;
        while (nodo->sig != NULL) //  Paso 1
        {
                anterior = nodo;
                nodo = nodo->sig;
        }
        extraido = nodo->nom; //  Paso 2

        if (!anterior)
        {
                cabecera = NULL;
        }
        else
        {
                anterior->sig = NULL; //  Paso 3
        }
        fin = anterior; //  Paso 4
        delete nodo; // Paso 5
        return extraido;

}
bool ListUsuario::Buscar_Val_Usuario(string usu, string con)
{
        bool encontrado = false;
        if (!cabecera) //cabecera == NULL
        {
                encontrado = false;
        }
        nodo = cabecera;
        while (nodo)//nodo != NULL
        {

                if ((nodo->nom == usu || nodo->correo == usu)&& nodo->contra == con)
                {
                        encontrado = true;
                        break;
                }
                nodo = nodo->sig;
        }
        return encontrado;
}
bool ListUsuario::Buscar_Usuario(string usu)
```

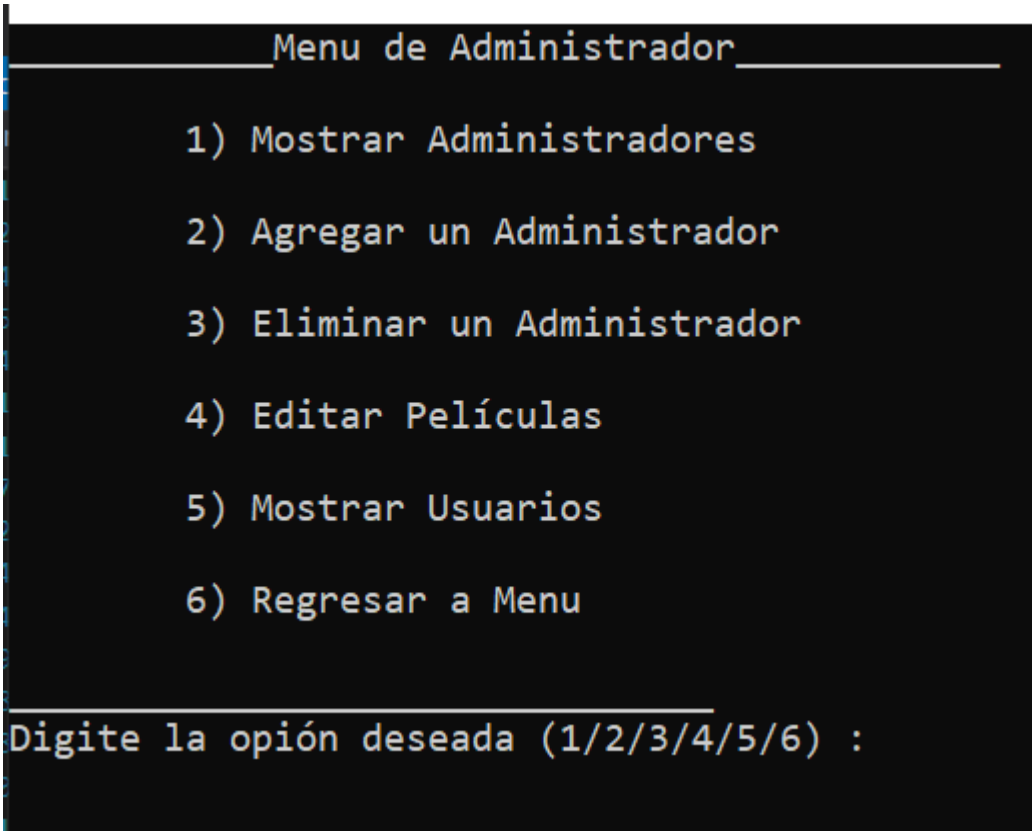
```
{
	bool encontrado = false;
	if (!cabecera) //cabecera == NULL
	{
		encontrado = false;
	}
	nodo = cabecera;
	while (nodo)//nodo != NULL
	{

		if (nodo->nom == usu)
		{
			encontrado = true;
			break;
		}
		nodo = nodo->sig;
	}
	return encontrado;
}
void ListUsuario::Reiniciar_BD()
{
	ofstream file("Administrador.csv");
	nodo = cabecera;
	if (file.is_open())
	{
		file << "Usuario" << ";" << "Contraseña" << endl;
		while (nodo)//nodo != NULL
		{
			file << nodo->nom << ";" << nodo->contra << endl;
			nodo = nodo->sig;
		}
	}
}
Usuario ListUsuario::Buscar_Info_Usuario(string Usu)
{
	Usuario Encont;

	bool encontrado = false;
	if (!cabecera) //cabecera == NULL
	{
		Encont.nom = "";
		return Encont;
	}
	nodo = cabecera;
	while (nodo)//nodo != NULL
	{

		if (nodo->nom == Usu || nodo->correo == Usu)
		{
			Encont.nom = nodo->nom;
			Encont.correo = nodo->correo;
			Encont.cash = nodo->cash;
			encontrado = true;
			break;
		}
		nodo = nodo->sig;

	}
```

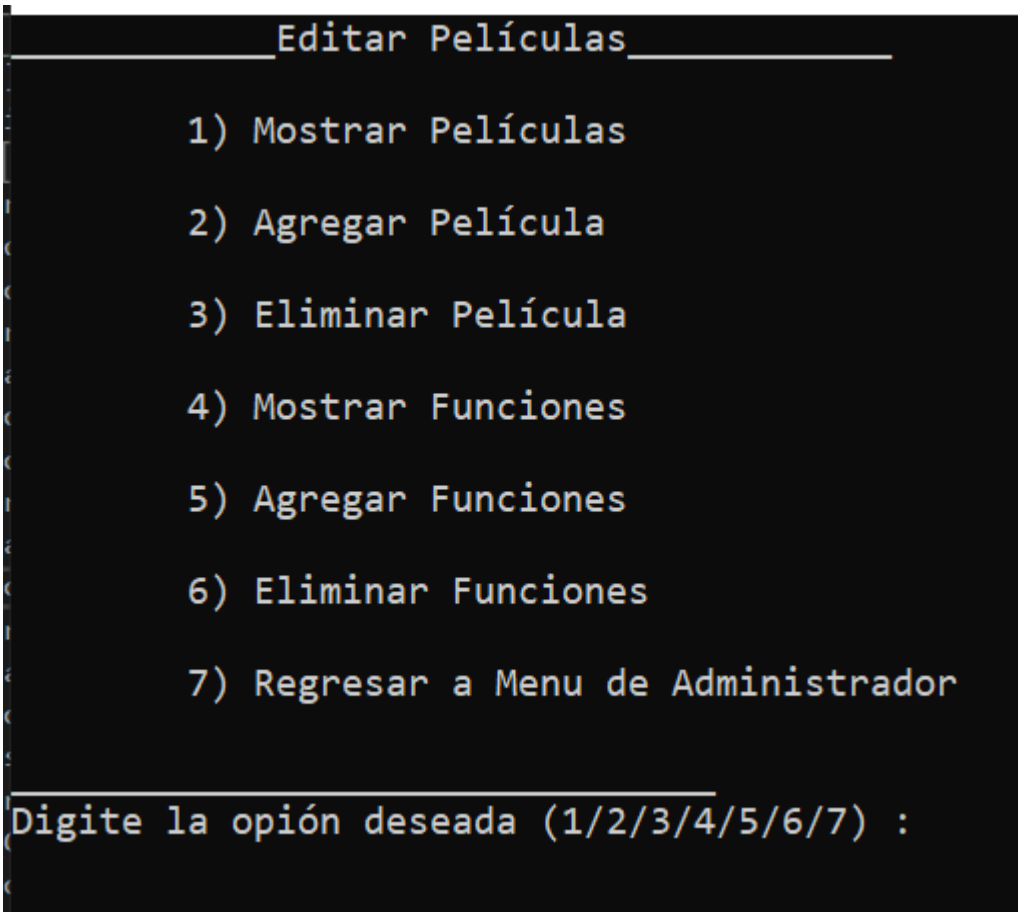
```
	return Encont;
}
bool ListUsuario::Cambiar_Cash(string usu, int rest)
{
	int cash =0;
	bool encontrado = false;
	if (!cabecera) //cabecera == NULL
	{
		encontrado = false;
	}
	nodo = cabecera;
	while (nodo)//nodo != NULL
	{

		if (nodo->nom == usu || nodo->correo == usu)
		{
			cash = nodo->cash;
			if (cash >= rest)
			{
				nodo->cash = cash - rest;
				encontrado = true;
			}
			else
			{
				encontrado = false;
			}
			break;
		}
		nodo = nodo->sig;
	}
	return encontrado;
}
bool ListUsuario::Sumar_Cash(string usu, int sum)
{
	int cash = 0;
	bool encontrado = false;
	if (!cabecera) //cabecera == NULL
	{
		encontrado = false;
	}
	nodo = cabecera;
	while (nodo)//nodo != NULL
	{

		if (nodo->nom == usu || nodo->correo == usu)
		{
			cash = nodo->cash;

			nodo->cash = cash + sum;
			encontrado = true;
			}
			else
			{
				encontrado = false;
			}
			break;

		nodo = nodo->sig;
```

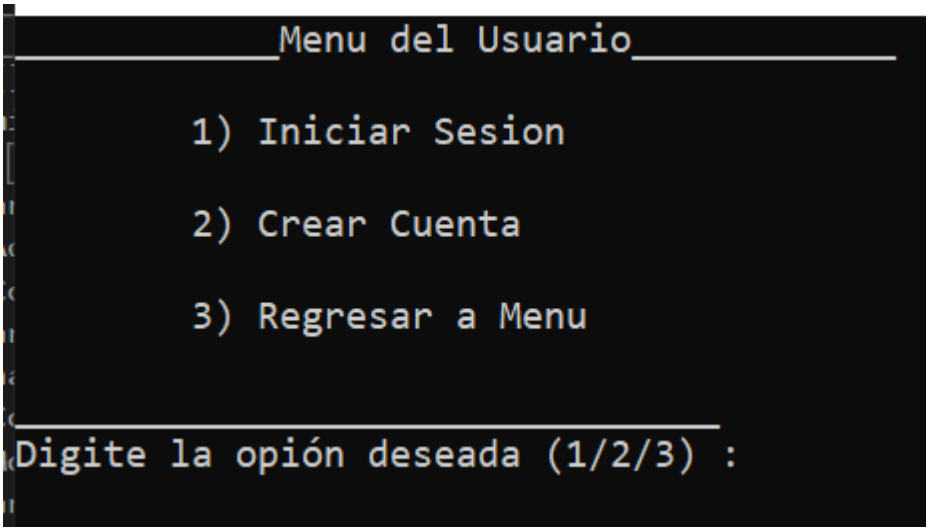
```
        }
        return encontrado;
}


#include "Cinema.h"

int main()
{
        Cinema miCine;
        miCine.LeeryCopiarAdmin();
        miCine.LeeryCopiarMovie();
        miCine.LeeryCopiarSalaHora();
        miCine.LeeryCopiarUsu();
        miCine.LeeryCopiarArch();
        miCine.Menu();
        return 0;
}
```

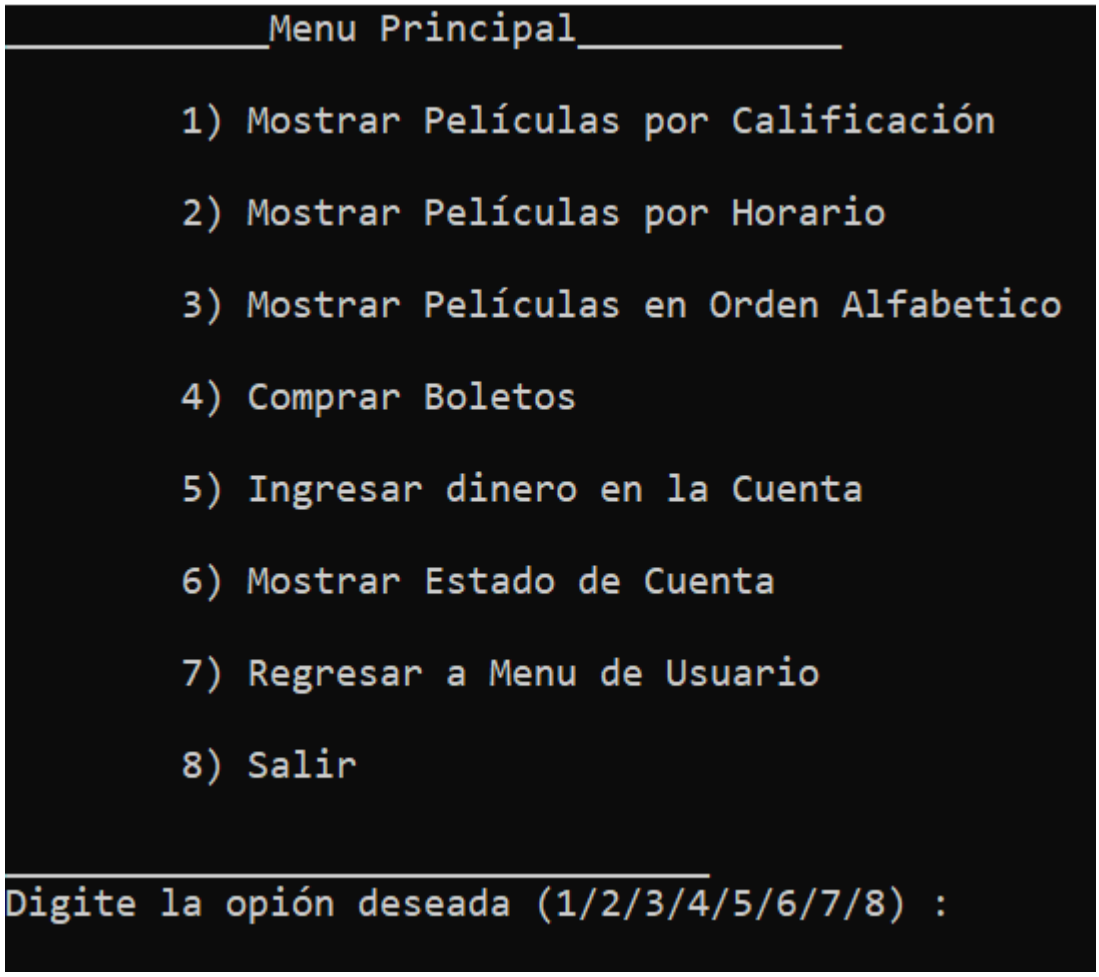
Capturas de pantalla:






Editar Películas
1) Mostrar Películas
2) Agregar Película
3) Eliminar Película
4) Mostrar Funciones
5) Agregar Funciones
6) Eliminar Funciones
7) Regregar a Menu de Administrador
Digite la opión deseada (1/2/3/4/5/6/7) :


Menu del Usuario
1) Iniciar Sesion
2) Crear Cuenta
3) Regresar a Menu
Digite la opión deseada (1/2/3) :


Menu Principal
1) Mostrar Películas por Calificación
2) Mostrar Películas por Horario
3) Mostrar Películas en Orden Alfabetico
4) Comprar Boletos
5) Ingresar dinero en la Cuenta
6) Mostrar Estado de Cuenta
7) Regresar a Menu de Usuario
8) Salir
Digite la opión deseada (1/2/3/4/5/6/7/8) :