
# UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA

## FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

**CARRERA**

TECNOLOGÍAS DE LAS INFORMACIÓN.

**TEMA**

“Programa de gestión de cine”

**AUTOR(S)**

Delgado Ladines Emilio David

Perero Rocafuerte Anthony Daniel

Salinas Vera Yanina Elizabeth

**MATERIA**

Programación avanzada I

**DOCENTE**

Ing. Jaime Orozco

LA LIBERTAD – ECUADOR

2023

# Gestor de cine

## Tema.-

Programa de gestión de cine que permita al usuario encontrar la información completa sobre una película, su transmisión y clasificación, registrar datos de cine, salas y sus sesiones y buscar información de manera específica, diseñado con el uso del lenguaje C++ dentro del editor de código DevC++.

## Objetivo.-

Desarrollar un programa en el lenguaje C++ capaz de gestionar y relacionar listas simples que guardan información sobre las películas en cartelera de uno o varios cines, junto a detalles de las mismas.

## Detalles del desarrollo.-

El programa será desarrollado en el lenguaje de programación “C++”, utilizando el Editor de Código “Dev-C++”. Dado que se manejará información, se necesitará utilizar Listas Simples doblemente enlazadas con la finalidad de almacenar y recuperar de manera eficiente la información correspondiente.

# — Código —

Cada clasificación será ordenada según su orden dentro del código fuente. La documentación de cada función está dividida en Nombre, Autor, Descripción y Funcionamiento.

## - Main -

Función Main

```
int main() {
  Cine * pcabCine = NULL;
  Cine * pfinCine = NULL;
  Pelicula * pcabPelicula = NULL;
  Pelicula * pfinPelicula = NULL;
  void busqueda_pelicula(Cine * pcabCine);

  setlocale(LC_CTYPE,"Spanish");
  int op;
  do{
    system("cls");
    op = menu_principal();
    operaciones_pr(op,pcabCine, pfinCine, pcabPelicula, pfinPelicula);
    fflush(stdin);
    gotoxy(55, 1);cout<<"Desea realizar otra operacion si(1): ";gotoxy(93,1);cin>>op;
  }while(op==1);




  getch();
  return 0;
}
```

# - Librerías -

## Programador(a) Autor(a):

Delgado Ladines Emilio David

## Descripción:

Conjunto de Librerías de C++ que se están utilizando a lo largo del Proyecto.

## Funcionamiento Breve:

Accede a librerías de C++ para facilitar el trabajo

```
#include<iostream>
#include<string.h>
#include <string>
#include<conio.h>
#include<locale.h>
#include<stdio.h>
#include<windows.h>
#include<stdlib.h>
using namespace std;
```

---

💡 Inicio de funciones Extras

# - Funciones Extra -

## Implementador(a) Autor(a):

Anthony Daniel Perero Rocafuerte

## Descripción:

El siguiente código utiliza find_first_not_of y find_last_not_of para obviar los espacios en blanco del inicio y final.

```
string trim(string str) {
    size_t first = str.find_first_not_of(' ');
    if (string::npos == first) {
        return str;
    }
    size_t last = str.find_last_not_of(' ');
    return str.substr(first, (last - first + 1));
}
```

---

### Implementador(a) Autor(a):

Emilio David Delgado Ladines

### Descripción:

Función “gotoxy” para modificar la posición de los ”cout” en la consola.

```
void gotoxy(int x,int y){
  HANDLE hcon;
  hcon=GetStdHandle(STD_OUTPUT_HANDLE);
  COORD dwPos;
  dwPos.X=x;
  dwPos.Y=y;
  SetConsoleCursorPosition(hcon,dwPos);
}
```

---

💡 Inicio de la documentación de Structs

## - Structs -

### Nombre:

Sesión

### Programador(a) Autor(a):

Delgado Ladines Emilio David

### Descripción:

Una estructura para generar nodos.

### Funcionamiento Breve:

Contiene los datos necesarios para la sesión, (en el futuro se deberá agregar una capacidad de ingresar a datos películas)

```
struct Sesion{
  int numero;
  string hora;


  Sesion * ant;
  Sesion * sig;
};
```

---

### Nombre:

Película

### Programador(a) Autor(a):

Emilio David Delgado Ladines

### Descripción:

Una estructura para generar nodos del tipo Película.

### Funcionamiento Breve:

Contiene los datos necesarios para registrar una Película, junto a sus punteros internos.

```
struct Pelicula{

  string nombre;

  string clasificacion;

  string protagonistas;

  Pelicula*ant;
  Pelicula * sig;
};
```

---

### Nombre:

Sala

### Programador(a) Autor(a):

Anthony Daniel Perero Rocafuerte

### Descripción:

Una estructura para generar nodos del tipo Sala.

### Funcionamiento Breve:

Contiene los datos necesarios para registrar una Sala, junto a sus punteros internos. Guarda los punteros “pcab” y “pfin” del tipo Sesion para generar la lista interna de Sesiones.

```
struct Sala {

    int numero;
    int Sesiones;
    bool Horario1;
    bool Horario2;
    bool Horario3;
    bool Horario4;

    Pelicula * PeliSala;

    Sesion * pcabSe;
    Sesion * pfinSe;


    struct Sala *ant;
```

```
    struct Sala *sig;
};
```

---

### Nombre:

Cine

### Programador(a) Autor(a):

Yanina Elizabeth Salinas Vera

### Descripción:

Una estructura para generar nodos.

### Funcionamiento Breve:

Contiene los datos necesarios para registrar un Cine, junto a sus punteros internos. Tiene una variable que guarda el número de salas que posee y será aumentada en 1 según las salas creadas. También cuenta con punteros cabecera y finales del tipo Sala, con la finalidad de guardar los datos de un listado de salas dentro del nodo.

```
struct Cine {
    string Nombre;
    int Salas;

    Sala * pcabS;
    Sala * pfinS;

    Cine* ant;
    Cine* sig;
};
```

---

💡 Inicio de la documentación de funciones

## - Funciones -

### Nombre:

menu_principal()

### Programador(a) Autor(a):

Delgado Ladines Emilio David

### Descripción:

Despliega un menú con las opciones principales.

### Funcionamiento Breve:

La función regresará un valor del tipo “int”. Despliega un menú en la pantalla para que el usuario elija una de las opciones, se trabaja con un “do-while” para verificar si la opción existe, si no existe vuelve a pedir la opción. La respuesta será guardada en la variable “resp” del tipo “int”, que es el valor devuelto por la función.

```
int menu_principal(){
  int resp;
  do{
    system("cls");
    cout<<"------------------MENU PRINCIPAL------------------"<<endl;
    cout<<"> 1. Registrar Cine"<<endl;
    cout<<"> 2. Registrar Salas"<<endl;
    cout<<"> 3. Registrar Sesiones"<<endl;
    cout<<"> 4. Registrar Pelicula"<<endl;
    cout<<"> 5. Visualizar información de Sala Concreta"<<endl;
    cout<<"> 6. Buscar pelicula"<<endl;
    cout<<"> 7. Modificar una película de una sala concreta"<<endl;
    cout<<"> 8. Listar peliculas por edades"<<endl;
    cout<<"> 9. Listar carteleras de todos los cines"<<endl;
    cout<<"> 10. Salir del programa"<<endl;
    cout<<"------------------||||||||||||||------------------"<<endl<<endl;
    cout<<"Elije una opcion [1--10]: ";
    cin>>resp;
  }while(resp<1 || resp>10 );

}
```

---

### Nombre:

operaciones_pr(op)

### Programador(a) Autor(a):

Delgado Ladines Emilio David

### Descripción:

Función de operaciones que ejecuta una funcionalidad dependiendo del valor de su parámetro.

### Funcionamiento Breve:

Compara el valor del parámetro “op” con los diferentes casos con ayuda del “switch”. En dado caso de coincidir el valor del parámetro con uno de los casos del switch, se ejecutarán las líneas y funciones correspondientes.

```
void operaciones_pr(int op , Cine *&pcabCine, Cine *& pfinCine,Pelicula *& pcabPelicula,Pelicula *& pfinPelicula){
  int numb=0;
  string cine;
  Cine * act;
  switch(op){
    case 1: //registrar cine

      RegistrarCine(pcabCine,pfinCine);
      //fun_listar(pcabCine);
      break;

    case 2: //Registrar Salas
      if(pcabCine!=NULL){
        crear_sala(pcabCine );
        break;
      }else{
        cout<<"No existen cines,Registra uno";
        break;
      }


    case 3: // registrar sesiones

      Registrar_Sesion(numb,pcabCine, pfinCine);
```

```
      break;

    case 4: //registra pelicula
      Registrar_Pelicula(pcabPelicula,pfinPelicula,pcabCine);
      break;
    case 5: //visualizar informacion de una sala concreta
      if (pcabCine!=NULL and pcabCine->pcabS!=NULL){
        info_Sala(pcabCine);
        break;
      }else{
        cout<<"No es posible ver la informacion de una sala, ";
        break;
      }


    case 6: //Buscar película
      if(pcabPelicula!=NULL){
        busqueda_pelicula(pcabCine, pcabPelicula);
        break;
      }else{
        cout<<"No existen peliculas";
        break;
      }


    case 7: //Listar las peliculas por edades
      pelis_edades(pcabPelicula, pcabCine);
      break;
    case 8: // listar carteleras de todos los cines
      Cartelera(pcabCine, pcabPelicula);
      break;
    case 9: //salir del programa
      char conf;
                cout << "¿Está seguro de salir (S/N)? ";cin >> conf;
                if (conf == 'S' || conf == 's') {

                    cout << "Saliendo del programa..." << endl;
                    exit(0);
                }else{
                  break;
        }


  }
}
```

### Nombre:

RegistrarCine(Cine*&pcab, Cine*&pfin)

### Programador(a) Autor(a):

Yanina Elizabeth Salinas Vera

### Descripción:

Crea un nodo con estructura del tipo cine y lo registra en la lista.

### Funcionamiento Breve:

Usando como parámetros punteros pcab y pfin enlazados a la estructura Cine, crea un bucle do-while en el que se pide al usuario el nombre del cine y el bucle repetirá la acción solo cuando el nombre del cine ya exista. Se eliminan los espacios iniciales y finales del nombre del cine con la función “trim()” y se realiza la verificación de nombre repetido con la función “verificar _Cine()”, esto devolverá un valor del tipo “bool” que se guardará en la variable Verf, la cual será evaluada para validad o rechazar el nombre.

Los valores internos del nodo son inicializados y definidos para evitar futuros problemas, dándole valor “NULL” a las variables que guarden dirección y de 1 a la variable “Salas” del nodo. A partir de ello, se ubica el nodo en la lista de Cines usando los punteros pcab y pfin como referencias.

```
void RegistrarCine(Cine*&pcab, Cine*&pfin) {
    Cine* NuevoNodo = new(Cine);
    //NuevoNodo->Nombre = nombre;
  bool Verf;
  system("cls");
  do{

      fflush(stdin);
      gotoxy(25,6);cout<<"Ingrese el nombre del cine: ";
      gotoxy(55,6);getline(cin, NuevoNodo->Nombre);
      NuevoNodo->Nombre = trim(NuevoNodo->Nombre);

      Verf = verificar_Cine(NuevoNodo->Nombre, pcab);
      if(Verf == true){
        gotoxy(55,6);cout<<"                    ";
        gotoxy(25,8);cout<<"** Cine Ya Existente **";
      }
  }while(Verf == true);
  gotoxy(25,8);cout<<"** Cine Registrado con Exito **";
  NuevoNodo->Salas= 1;
  NuevoNodo->pcabS = NULL;
    NuevoNodo->pfinS = NULL;
  NuevoNodo->sig = NULL;
  NuevoNodo->ant = NULL;



    if(pcab==NULL){ //significa que es el primero
    pcab=NuevoNodo;
  }else{ //ya existe nodo
    pfin->sig=NuevoNodo;
    NuevoNodo->ant=pfin;
  }
  pfin=NuevoNodo;
}
```

### Nombre:

buscar_cine(string datobuscar,Cine *pcab)

### Programador(a) Autor(a):

Yanina Elizabeth Salinas Vera

### Descripción:

Busca un nodo Cine que coincida con el dato ingresado.

### Funcionamiento Breve:

Devuelve una dirección de nodo del tipo Cine, usando de parametro “datobuscar” que contendrá un string y el puntero cabecera de la lista Cine buscará un nodo. Iniciando por guardar el nodo pcab en la variable enlazada a la estructura Cine llamada “nodoAct”, la cual se actualizará constantemente. Se crea un bucle while que verificará si la variable nodoAct contiene un valor diferente de NULL, lo que significaría guarda un nodo. Se ingresa al nodo guardado por nodoAct y compara el valor de su variable interna “Nombre” con el parámero “datobuscar”, si el dato coindice la dirección del nodo con dicho dato será retornada, sino entrará al siguiente nodo.  En caso de no encontrar el nodo con el dato buscado, retornará el valor NULL.

```
Cine *buscar_cine(string datobuscar,Cine *pcab){
  Cine *nodoAct=pcab;
  while(nodoAct!=NULL){
    if(nodoAct->Nombre==datobuscar){
      return nodoAct;
    }
      nodoAct=nodoAct->sig;
  }

   return NULL;
}
```

---

### Nombre:

verificar_Cine(string nombre, Cine *pcab)

### Programador(a) Autor(a):

Yanina Elizabeth Salinas Vera

### Descripción:

Verifica si un nodo Cine existe.

### Funcionamiento Breve:

Devuelve un valor del tipo “bool”, utiliza los parámetros “nombre” y “pcab”.

Crea una variable enlazada a la estructura Cine llamada “dirCine”, esta variable tomará el valor retornado de la función “buscar_cine()”. Si esta dirección es diferente a “NULL”, la función retornará el valor de verdad “true”, sino retornará “false”.

```
bool verificar_Cine(string nombre, Cine *pcab){
  Cine * dirCine;
  dirCine = buscar_cine(nombre, pcab);

  if(dirCine != NULL){
    return true;
  }
  return false;
}
```

---

### Nombre:

crear_sala(Cine*pcabcine)

### Programador(a) Autor(a):

Anthony Daniel Perero Rocafuerte

### Descripción:

Crea una nueva sala dentro de un nodo Cine.

### Funcionamiento Breve:

Crea un nuevo nodo con la estructura Sala usando como parámetro la variable “pcabcine” enlazada a la estructura Cine. Inicializa las variables necesarias, “cineAct” que guardará un nodo Cine, “cine” que guardará el nombre del cine, “Verf” que guardará un valor de verdad dado una verificación, y dentro del nuevo nodo “nuevaSala” su variable interna “Sesiones” tendrá el valor de 1.

Se realiza la petición al usuario del nombre del cine al que pertenecerá la sala, este valor se guarda en la variable cine y se usará para verificar la existencia de un nodo con el mismo valor dentro de la lista Cine. Esta funcionalidad está dentro de un do-while por lo que se repetirá siempre que la variable Verf sea falsa, en otras palabras, si no existe un nodo Cine con el nombre que se busca.

Una vez validado el nombre, cineAct tomará el valor del nodo Cine encontrado. El número de la sala será asignado en función a la variable interna “Salas” del nodo Cine correspondiente. Las variables internas del nodo Sala son inicializadas, siendo los punteros iguales a NULL, los datos del tipo “bool” iguales a cero y se aumenta el contador de Salas.

A continuación, se ubica el nodo Sala dentro de la lista interna del cine usando los punteros correspondientes pertenecientes al nodo Cine.

```
void crear_sala(Cine*pcabcine){
  Sala * nuevaSala = new(Sala);
  Cine * cineAct;
  string cine;
  bool Verf;
  nuevaSala->Sesiones = 1;
//  string nombre;
  system("cls");
  fflush(stdin);

  //pregunta a que cine pertenece y verifica si existe
  do{
    gotoxy(35, 5);cout<<("                              ");
    gotoxy(5, 5);cout<<("Ingrese a que Cine pertenece: ");
    gotoxy(35, 5);getline(cin,cine);

    Verf = verificar_Cine(cine, pcabcine);
  }while(Verf == false);
  cineAct = buscar_cine(cine,pcabcine);
  gotoxy(5, 7);cout<<("Numero de Sala: ");gotoxy(35, 7);cout<<cineAct->Salas;
```

```
  //------- Ingresar demas detos

  nuevaSala->numero = cineAct->Salas;
  nuevaSala->pcabSe = NULL;
  nuevaSala->PeliSala = NULL;
  nuevaSala->pfinSe = NULL;
  nuevaSala -> ant = NULL;
  nuevaSala -> sig = NULL;
  nuevaSala -> Horario1 = false;
  nuevaSala -> Horario2 = false;
  nuevaSala -> Horario3 = false;
  nuevaSala -> Horario4 = false;

  cineAct->Salas += 1;




  if (cineAct->pcabS == NULL){

    cineAct->pcabS = nuevaSala;
  }else{
    nuevaSala->ant = cineAct->pfinS;
    cineAct->pfinS->sig = nuevaSala;
  }
  cineAct->pfinS = nuevaSala;

}
```

---

### Nombre:

buscar_sala(Sala *inicio, int num)

### Programador(a) Autor(a):

Anthony Daniel Perero Rocafuerte

### Descripción:

Buscan un nodo del tipo Sala.

### Funcionamiento Breve:

Devuelve una dirección de nodo del tipo Sala, usando de parámetro “num” que contendrá un integer y el puntero cabecera de la lista Sala, buscará un nodo. Iniciando por guardar el nodo inicio en la variable enlazada a la estructura Cine llamada “actual”, la cual se actualizará constantemente. Se crea un bucle while que verificará si la variable “actual” contiene un valor diferente de NULL, lo que significaría guarda un nodo. Se ingresa al nodo guardado por “actual” y compara el valor de su variable interna “numero” con el parámero “num”, si el dato coindice la dirección del nodo con dicho dato será retornada, sino entrará al siguiente nodo.  En caso de no encontrar el nodo con el dato buscado, retornará el valor NULL.

```
Sala *buscar_sala(Sala *inicio, int num) {
  Sala *actual = inicio;
    while (actual != NULL) {// de no existir un sala existente se continua y registra
        if (actual->numero == num) {
            return actual;
        }
        actual = actual->sig;
    }
    return NULL;
}
```

---

### Nombre:

bool verificar_Sala(int num, Sala *pcab)

### Programador(a) Autor(a):

Anthony Daniel Perero Rocafuerte

### Descripción:

Verifica si un nodo del tipo “Sala” existe, en base a una variable.

### Funcionamiento Breve:

Devuelve un valor del tipo “bool”, utiliza los parámetros “num” y “pcab”.

Crea una variable enlazada a la estructura Sala llamada “dirSala”, esta variable tomará el valor retornado de la función “buscar_sala()”. Si esta dirección es diferente a “NULL”, la función retornará el valor de verdad “true”, sino retornará “false”.

```
bool verificar_Sala(int num, Sala *pcab){
  Sala * dirSala;
  dirSala = buscar_sala(pcab,num);

  if(dirSala != NULL){
    return true;
  }
  return false;
}
```

---

### Nombre:

Registrar_Sesion( int numero_sala,Cine *&pcabCine, Cine *& pfinCine)

### Programador(a) Autor(a):

Delgado Ladines Emilio David

### Descripción:

Registra una nueva Sesión.

### Funcionamiento Breve:

Crea un nuevo Nodo con la estructura Sesión, pide los datos que necesita el nodo y asigna sus punteros internos a “NULL”.  Pide el cine al que pertenecerá, haciendo las verificaciones necesarias, luego pide el número de sala a la que pertenecerá la sesión. Se accede a la sala mediante el puntudo cabecera Sala del cine padre. Verificará si dentro del nodo padre Sala, las variables de horario están vacias, si es así seguirá adelante, sino avisará al usuario que no podrá crear una nueva Sesion.

Se ubica el nodo dentro de la lista interna del nodo Sala asignado y asigna el horario a la sesión.

```
void Registrar_Sesion( int numero_sala,Cine *&pcabCine, Cine *& pfinCine){

  Sala * salaAct;
```

```
  Cine * cineAct;
  string cine;
  int sala;
  bool Verf;
//  string nombre;
  system("cls");
  fflush(stdin);

  //pregunta a que cine pertenece y verifica si existe
  do{
    gotoxy(35, 5);cout<<("                              ");
    gotoxy(5, 5);cout<<("Ingrese a que Cine pertenece: ");gotoxy(35, 5);getline(cin,cine);
    Verf = verificar_Cine(cine, pcabCine);
  }while(Verf == false);
  cineAct = buscar_cine(cine,pcabCine);

  do{
    gotoxy(35, 7);cout<<("                              ");
    gotoxy(5, 7);cout<<("Ingrese a que Sala pertenece: ");gotoxy(35, 7);cin>>sala;
    Verf = verificar_Sala(sala, cineAct->pcabS);
  }while(Verf == false);
  salaAct = buscar_sala(cineAct->pcabS, sala);

  if(salaAct->Horario1 == false || salaAct->Horario2 == false||salaAct->Horario3 == false||salaAct->Horario4 == false){
    Sesion * nuevaSesion = new(Sesion);

  nuevaSesion->numero = salaAct->Sesiones;
  gotoxy(5, 9);cout<<("Numero de Sesion: ");gotoxy(35, 9);cout<<salaAct->Sesiones;
  salaAct->Sesiones +=1;

  nuevaSesion->sig = NULL;
  nuevaSesion->ant = NULL;

  if (salaAct->pcabSe == NULL){

    salaAct->pcabSe = nuevaSesion;
  }else{
    nuevaSesion->ant = salaAct->pfinSe;
    salaAct->pfinSe->sig = nuevaSesion;
  }
  salaAct->pfinSe = nuevaSesion;



  Seleccion_Horario(salaAct, nuevaSesion);



}else{
if(salaAct->Horario1 == true && salaAct->Horario2 == true&&salaAct->Horario3 == true&&salaAct->Horario4 == true){
      gotoxy(5, 9);cout<<(" No hay Horarios Disponibles ");
    }
  }
}
```

## Nombre:

Seleccion_Horario(Sala * salaAct, Sesion * sesionAct)

## Programador(a) Autor(a):

Emilio David Delgado Ladines

## Descripción:

Le proporciona al usuario las opciones para elegir el horario de la sesión creada.

## Funcionamiento Breve:

Se encarga de permitir al usuario seleccionar un horario para una sesión en una sala de cine. La función muestra una lista de horarios disponibles en la pantalla y solicita al usuario que elija uno. El usuario ingresa un número correspondiente al horario deseado, y la función verifica si ese horario está disponible en la sala. Si el horario está disponible, se asigna a la sesión y se marca como ocupado en la sala. Si el horario ya está ocupado, se notifica al usuario y se le vuelve a solicitar que elija un horario disponible. Este proceso es realizado mediante la implementación de un “switch” que evalúa cada caso, adicionalmente, una vez elegido el horario se modificará la variable “hora” de la sesión para que tenga el valor elegido.

```
void Seleccion_Horario(Sala * salaAct, Sesion * sesionAct){
  int op;
  bool verf;
  do{
    verf = false;
    do{
        fflush(stdin);

        gotoxy(45, 5);cout<<("|");gotoxy(50, 5);cout<<("1.- Horario 1: 10:30am ");
        gotoxy(45, 7);cout<<("|");gotoxy(50, 7);cout<<("2.- Horario 2: 12:30pm ");
        gotoxy(45, 9);cout<<("|");gotoxy(50, 9);cout<<("3.- Horario 3: 15:00pm ");
        gotoxy(45, 11);cout<<("|");gotoxy(50, 11);cout<<("4.- Horario 4: 20:00pm ");
        gotoxy(50, 13);cout<<("Ingrese el horario de la sala: ");gotoxy(82, 13);cin>>op;
    }while(op<0 || op>4);

    switch(op){
      case 1:
        if(salaAct->Horario1 != true){
          sesionAct->hora = "10:30am";
          salaAct->Horario1 = true;

          break;
        }else{
          verf= true;
          break;
        }



      case 2:
        if(salaAct->Horario2 != true){
          sesionAct->hora = "12:30am";
          salaAct->Horario2 = true;
          break;
        }else{
          verf= true;
          break;
        }

      case 3:
        if(salaAct->Horario3 != true){
          sesionAct->hora = "15:00pm";
          salaAct->Horario3 = true;
          break;
        }else{
          verf= true;
          break;
        }

      case 4:
        if(salaAct->Horario4 != true){
          sesionAct->hora = "20:00pm";
          salaAct->Horario4 = true;
          break;
        }else{
          verf= true;
          break;
        }

      }
```

```
  }while(verf==true);

}
```

---

### Nombre:

Registrar_Pelicula(Pelicula *&pcabP,Pelicula *&pfinP, Cine * pcabCine)

### Programador(a) Autor(a):

Emilio David Delgado Ladines

### Descripción:

Da al usuario la opción de crear una película o asignar una película a una sala. Al momento de asignar una película a una sala y esta ya esté ocupada, dicha película será reemplazada por la de nueva asignación.

### Funcionamiento Breve:

La función Registrar_Pelicula es una función que permite al usuario realizar dos acciones relacionadas con películas en un sistema de gestión de cines. La función comienza mostrando un menú en pantalla con dos opciones: "Registrar Película" y "Asignar Película a Sala". El usuario elige una de estas opciones ingresando un número (1 o 2). La función valida que la elección del usuario esté dentro del rango esperado (1 o 2).

Si el usuario elige la opción de "Registrar Película" (op == 1), la función permite al usuario ingresar información sobre una nueva película, incluyendo su nombre, clasificación y los nombres de los protagonistas. Verifica si la película ya existe en la lista de películas utilizando la función Buscar_Pelicula. Luego, crea un nuevo nodo de película y asigna los valores ingresados por el usuario a ese nodo. Luego, agrega ese nodo a una lista enlazada de películas representada por pcabP y pfinP.

Si el usuario elige la opción de "Asignar Película a Sala" (op == 2), la función permite al usuario seleccionar una película ya registrada y asignarla a una sala de cine específica. El usuario debe ingresar el nombre de la película, el nombre del cine al que pertenece la sala y el número de la sala. La función verifica que la película, el cine y la sala existan en las respectivas listas utilizando funciones como verificar_Peli, verificar_Cine y verificar_Sala. Luego, asigna la película a la sala especificada en el campo PeliSala de la estructura de sala.

En resumen, esta función permite al usuario registrar nuevas películas en el sistema o asignar películas existentes a salas de cine específicas, gestionando así la base de datos de películas y su asignación a las salas.

```
void Registrar_Pelicula(Pelicula *&pcabP,Pelicula *&pfinP, Cine * pcabCine){
  int op;
  string nombre;
  string clasificacion;
  string Protas;
  system("cls");
  fflush(stdin);
  do{
    gotoxy(5,5); cout<<"1.- Registrar Película";
    gotoxy(5,7); cout<<"2.- Asignar Película a Sala";
    gotoxy(5,10); cout<<"Ingrese opcion a elegir |1 - 2|: ";gotoxy(40,10);cin>>op;
  }while(op<0 || op>2);

  if (op==1){
    Pelicula *NuevoNodo=new(Pelicula); //ab001
    do{
      fflush(stdin);
      gotoxy(5, 12);cout<<"Ingrese el Nombre: ";gotoxy(40, 12);getline(cin, NuevoNodo->nombre);
    }while(Buscar_Pelicula(NuevoNodo->nombre,pcabP) != NULL);
```

```
    //registrar clasificacion
    clasificacion = Registrar_Clasificcion();
    NuevoNodo->clasificacion = clasificacion;

    fflush(stdin);
    gotoxy (5,24);cout<<"Ingrese el nombre de los protagonistas: "; gotoxy(45, 24);getline(cin, Protas);
    NuevoNodo->protagonistas = Protas;




    NuevoNodo->ant=NULL;
    NuevoNodo->sig=NULL;

    if(pcabP==NULL){ //significa que es el primero
      pcabP=NuevoNodo;
    }else{ //ya existe nodo
      pfinP->sig=NuevoNodo;
      NuevoNodo->ant=pfinP;
    }
    pfinP=NuevoNodo;
  }else{
    Sala * salaAct;
    Cine * cineAct;
    Pelicula * peliAct;
    string cine;
    string peli;
    int sala;
    bool Verf;
  //  string nombre;
    fflush(stdin);

    do{
      gotoxy(35, 14);cout<<("                              ");
      gotoxy(5, 14);cout<<("Ingrese la Pelicula a asignar: ");gotoxy(35, 14);getline(cin,peli);
      Verf = verificar_Peli(peli, pcabP);
    }while(Verf == false);
    peliAct = Buscar_Pelicula(peli,pcabP);


    //pregunta a que cine pertenece y verifica si existe
    do{
      gotoxy(35, 16);cout<<("                              ");
      gotoxy(5, 16);cout<<("Ingrese a que Cine pertenece: ");gotoxy(35, 16);getline(cin,cine);
      Verf = verificar_Cine(cine, pcabCine);
    }while(Verf == false);
    cineAct = buscar_cine(cine,pcabCine);

    do{
      gotoxy(35, 18);cout<<("                              ");
      gotoxy(5, 18);cout<<("Ingrese a que Sala pertenece: ");gotoxy(35, 18);cin>>sala;
      Verf = verificar_Sala(sala, cineAct->pcabS);
    }while(Verf == false);
    salaAct = buscar_sala(cineAct->pcabS, sala);

    salaAct-> PeliSala = peliAct;
  }
}
```

## Nombre:

Buscar_Pelicula(string datobuscar,Pelicula *pcab)

## Programador(a) Autor(a):

Emilio David Delgado Ladines

### Descripción:

Busca un nodo Película en base a un dato y devuelve su dirección.

### Funcionamiento Breve:

La función Buscar_Pelicula tiene como objetivo encontrar una película en una lista enlazada de películas, buscando por su nombre. La función recibe el nombre de la película a buscar en el parámetro datobuscar y la lista de películas representada por pcab. La función inicia un bucle while que recorre la lista de películas desde el principio hasta el final. En cada iteración, compara el nombre de la película en el nodo actual (nodoAct) con el nombre proporcionado en datobuscar. Si encuentra una coincidencia, la función retorna un puntero al nodo que contiene la película buscada. Esto indica que la película ha sido encontrada en la lista.

Si el bucle termina de recorrer la lista y no encuentra una película con el nombre especificado, la función retorna NULL, lo que indica que la película no se encuentra en la lista. En resumen, esta función busca una película por su nombre en una lista de películas y devuelve un puntero al nodo que contiene la película si se encuentra, o NULL si la película no está en la lista.

```
Pelicula *Buscar_Pelicula(string datobuscar,Pelicula *pcab){
  Pelicula *nodoAct=pcab;
  while(nodoAct!=NULL){
    if(nodoAct->nombre ==datobuscar){
      return nodoAct;
    }
      nodoAct=nodoAct->sig;
  }

   return NULL;
}
```

---

### Nombre:

verificar_Peli(string nombre, Pelicula *pcab)

### Programador(a) Autor(a):

Emilio David Delgado Ladines

### Descripción:

Verifica si existe un nodo película en base a un dato dado.

### Funcionamiento Breve:

La función verificar_Peli se encarga de verificar si una película con un nombre específico ya existe en la lista de películas representada por pcab. Para hacer esto, la función recibe el nombre de la película a verificar en el parámetro nombre y llama a la función Buscar_Pelicula para buscar esa película en la lista. Si la función Buscar_Pelicula devuelve un puntero válido (diferente de NULL), lo que indica que la película existe en la lista, la función verificar_Peli retorna true, lo que significa que la película está registrada en la base de datos. En caso contrario, si la película no se encuentra en la lista, la función retorna false, indicando que la película no existe en la base de datos.

```
bool verificar_Peli(string nombre, Pelicula *pcab){
  Pelicula * dirPeli;
  dirPeli = Buscar_Pelicula(nombre, pcab);
```

```
  if(dirPeli != NULL){
    return true;
  }
  return false;
}
```

---

### Nombre:

Registrar_Clasificcion()

### Programador(a) Autor(a):

Emilio David Delgado Ladines

### Descripción:

Registra una clasificación en una película.

### Funcionamiento Breve:

La función Registrar_Clasificcion permite al usuario seleccionar una clasificación para una película. La función muestra una lista de opciones en pantalla, representando las clasificaciones de películas comunes, como "Todo Público (TP)", "Mayores de 15 (M15)", "Mayores de 18 (M18)" e "Infantil (IFN)". El usuario elige una clasificación ingresando un número correspondiente a las opciones presentadas. La función verifica que el número seleccionado esté dentro del rango válido (entre 1 y 4) y, según la selección del usuario, retorna la clasificación correspondiente en forma de una cadena de texto, como "TP", "M15", "M18" o "IFN".

La estructura switch se utiliza en esta función para determinar cuál clasificación de película ha seleccionado el usuario en función del valor de op. El switch es una estructura de control que permite seleccionar uno de varios casos basados en el valor de una expresión. En este caso, la expresión es el valor de op, que representa la elección del usuario.

```
string Registrar_Clasificcion(){
  int op;



  do{
    fflush(stdin);
    gotoxy(5, 14);cout<<("1.- Todo Publico > TP ");
    gotoxy(5, 16);cout<<("2.- Mayores de 15 > M15 ");
    gotoxy(5, 18);cout<<("3.- Mayores de 18 > N18 ");
    gotoxy(5, 20);cout<<("4.- Infantil > IFN");
    gotoxy(5, 22);cout<<("Ingrese la clasificacion de la Pelicula |1 - 4|: ");gotoxy(55, 22);cin>>op;
  }while(op<0 || op>4);

    switch(op){
      case 1:
        return "TP";
        break;

      case 2:
        return "M15";
        break;


      case 3:
        return "M18";
        break;
```

```
      case 4:
        return "IFN";
        break;
    }
  }
```

### Nombre:

info_Sala(Cine*pcabCine)

### Programador(a) Autor(a):

Emilio David Delgado Ladines

### Descripción:

Muestra la información de una sala concreta.

### Funcionamiento Breve:

La función info_Sala tiene como objetivo mostrar información detallada sobre una sala de cine en particular. En primer lugar, se solicita al usuario que ingrese el nombre del cine al que pertenece la sala y se verifica que el cine exista en la lista de cines. Luego, se le pide al usuario que ingrese el número de sala y se verifica que la sala exista en la lista de salas del cine seleccionado. Una vez obtenidos el cine y la sala, se muestra en pantalla información relevante, como el nombre del cine, el número de sala, y si está asignada una película a esa sala, se muestra el nombre de la película, sus protagonistas y su clasificación. En caso de que no haya película asignada a la sala, se informa al usuario.

Además, la función llama a listar_sesiones para mostrar los horarios de las sesiones en esa sala. En resumen, esta función proporciona información detallada sobre una sala de cine específica, incluyendo los detalles de la película asignada y los horarios de las sesiones disponibles.

```
void info_Sala(Cine*pcabCine){

  //identificacion de la sala --------------------------
  Sala * salaAct;
  Cine * cineAct;
  string cine;
  int sala;
  bool Verf;
//  string nombre;
  system("cls");
  fflush(stdin);

  //pregunta a que cine pertenece y verifica si existe
  do{
    gotoxy(35, 5);cout<<("                              ");
    gotoxy(5, 5);cout<<("Ingrese a que Cine pertenece: ");gotoxy(35, 5);getline(cin,cine);
    Verf = verificar_Cine(cine, pcabCine);
  }while(Verf == false);
  cineAct = buscar_cine(cine,pcabCine);

  do{
    gotoxy(35, 7);cout<<("                              ");
    gotoxy(5, 7);cout<<("Ingrese la Sala: ");gotoxy(35, 7);cin>>sala;
    Verf = verificar_Sala(sala, cineAct->pcabS);
  }while(Verf == false);
  salaAct = buscar_sala(cineAct->pcabS, sala);

  // ----------------------------------------

  //Mostrar Info-------
```

```
  gotoxy(5, 10);cout<<"Cine: ";gotoxy(20, 10);cout<<cineAct->Nombre;
  gotoxy(5, 12);cout<<"Sala: ";gotoxy(20, 12);cout<<salaAct->numero;
  if(salaAct->PeliSala !=NULL){
    gotoxy(5, 16);cout<<"Pelicula: ";gotoxy(20, 16);cout<<salaAct->PeliSala->nombre;
    gotoxy(5, 18);cout<<"Protagonistas: ";gotoxy(20, 18);cout<<salaAct->PeliSala->protagonistas;
    gotoxy(5, 20);cout<<"Clasificacion: ";gotoxy(20, 20);cout<<salaAct->PeliSala->clasificacion;

  }else{
    gotoxy(5, 16);cout<<"Pelicula: ";gotoxy(20, 16);cout<<"Sin Pelicula Asignada";
    gotoxy(5, 18);cout<<"Protagonistas: ";gotoxy(20, 18);cout<<"Sin Pelicula Asignada";
    gotoxy(5, 20);cout<<"Clasificacion: ";gotoxy(20, 20);cout<<"Sin Pelicula Asignada";
  }


  int e = 0;

  listar_sesiones(salaAct->pcabSe,23,e);
  fflush(stdin);
}
```

---

### Nombre:

listar_sesiones(Sesion * pcab, int Y, int &ref)

### Programador(a) Autor(a):

Emilio David Delgado Ladines

### Descripción:

Muestra las sesiones dentro de una sala.

### Funcionamiento Breve:

Usando el puntero “pcab”, se recorre la lista de sesiones dentro de una sala y ve muestran sus datos en pantalla. La función cuenta por un parámetro “Y” que será usado para modificar las funciones gotoxy, también cuenta con un valor referencia que aumentará para que la función principal reconozca dónde termina el listado de las sesiones.

```
void listar_sesiones(Sesion * pcab, int Y, int &ref){
  Sesion *nodoAct=pcab;
  int y = Y;//23;

  while(nodoAct!=NULL){
    //Presentar los datos
    gotoxy(5, y);cout<<"|>-----------------------------------------------<|";
    gotoxy(5, y+2);cout<<"|>Numero de Sesion: ";gotoxy(30,y+2);cout<<nodoAct->numero;
    gotoxy(5, y+4);cout<<"|>Horario de la sesion: ";gotoxy(30,y+4);cout<<nodoAct->hora;
    gotoxy(5,y+6);cout<<"|>-----------------------------------------------<|";
    //alterar la condicion o el bucle
    y+=6;
    ref += 8;
    nodoAct=nodoAct->sig;
    //gotoxy(5,30);cout<<"SIguienteeee";

  }
}
```

---

### Nombre:

busqueda_pelicula(Cine * pcabCine, Pelicula * pcabP)

### Programador(a) Autor(a):

Emilio David Delgado Ladines

### Descripción:

Busca una película en base a su nombre y nombra dónde será proyectada.

### Funcionamiento Breve:

La función busqueda_pelicula tiene como objetivo buscar una película en una lista de cines y salas, y mostrar información relacionada con esa película. Primero, se borra la pantalla para limpiarla. Luego, se le pide al usuario que ingrese el nombre de una película. La función verifica si la película existe en la lista de películas representada por pcabP usando la función verificar_Peli, y si se encuentra, se asigna la película correspondiente a peliAct. A continuación, se muestran en pantalla detalles de la película, como su nombre, clasificación y protagonistas.

Luego, la función recorre la lista de cines representada por la estructura Cine a través del puntero pcabCine. Para cada cine, recorre la lista de salas de ese cine a través del puntero salaAct. Si la película seleccionada se proyecta en alguna de las salas de ese cine (verificado por salaAct->PeliSala == peliAct), muestra detalles de la sala, como su número y los horarios de las sesiones disponibles usando la función listar_sesiones. La información se presenta en la pantalla de manera organizada.

```
void busqueda_pelicula(Cine * pcabCine, Pelicula * pcabP){
  system("cls");
  Pelicula * peliAct;
  string cine;
  string peli;
  int sala;
  bool Verf;
  //  string nombre;
  fflush(stdin);

  do{
    gotoxy(35, 10);cout<<("                              ");
    gotoxy(5, 10);cout<<("Ingrese la Pelicula a asignar: ");gotoxy(35, 10);getline(cin,peli);
    Verf = verificar_Peli(peli, pcabP);
  }while(Verf == false);
  peliAct = Buscar_Pelicula(peli,pcabP);
  gotoxy(5,11);cout<<"Nombre: ";gotoxy(35,11);cout<<peliAct->nombre;
  gotoxy(5,12);cout<<"Clasificacion: ";gotoxy(35,12);cout<<peliAct->clasificacion;
  gotoxy(5,13);cout<<"Protagonistas: ";gotoxy(35,13);cout<<peliAct->protagonistas;


  Cine *cineAct=pcabCine;
  Sala * salaAct;
  int y= 16;
  //int y2=18;

  while(cineAct!=NULL){
    //Presentar los datos
    gotoxy(5, y);cout<<"Cine: ";gotoxy(15, y);cout<<cineAct->Nombre;
    salaAct = cineAct->pcabS;

    if(salaAct->PeliSala == peliAct){
      while(salaAct!=NULL){



      gotoxy(5, y+2);cout<<"Sala:";gotoxy(15, y+2 );cout<<salaAct->numero;
```

```
      listar_sesiones(salaAct->pcabSe, y+3, y);

      salaAct = salaAct->sig;
      y+=4;
      }
    }


    y +=4 ;
    cineAct=cineAct->sig;
  }
}
```

---

### Nombre:

Cartelera(Cine * pcabCine, Pelicula * pcabP)

### Programador(a) Autor(a):

Emilio David Delgado Ladines

### Descripción:

Muestra todas las sesiones y películas de todos los cines.

### Funcionamiento Breve:

La función Cartelera se encarga de mostrar en pantalla una lista de cines, sus salas y las películas que se proyectan en cada sala. Comienza limpiando la pantalla y, a través de un bucle, recorre la lista de cines representada por la estructura Cine a través del puntero pcabCine. Para cada cine, muestra el nombre del cine y, a continuación, recorre la lista de salas de ese cine a través del puntero salaAct. Para cada sala, muestra el número de sala y, si hay una película proyectada en esa sala (representada por salaAct->PeliSala), muestra detalles de la película, como su nombre, clasificación y protagonistas. Además, llama a la función “listar_sesiones” para mostrar los horarios de las sesiones de esa sala.

```
void Cartelera(Cine * pcabCine, Pelicula * pcabP){
  system("cls");
  Pelicula * peliAct;
  string cine;
  string peli;
  int sala;
  bool Verf;
  //  string nombre;
  fflush(stdin);
  Cine *cineAct=pcabCine;
  Sala * salaAct;
  int y= 8;
  //int y2=18;

  while(cineAct!=NULL){
    //Presentar los datos
    gotoxy(5, y);cout<<"Cine >>> ";gotoxy(25, y);cout<<cineAct->Nombre;
    salaAct = cineAct->pcabS;

    while(salaAct!=NULL){
      gotoxy(5, y+2);cout<<"Sala >>>";gotoxy(25, y+2 );cout<<salaAct->numero;
      if(salaAct->PeliSala){
        gotoxy(5,y+4);cout<<"Nombre >>> ";gotoxy(25,y+4);cout<<salaAct->PeliSala->nombre;
        gotoxy(5,y+6);cout<<"Clasificacion >>> ";gotoxy(25,y+6);cout<<salaAct->PeliSala->clasificacion;
        gotoxy(5,y+8);cout<<"Protagonistas >>> ";gotoxy(25,y+8);cout<<salaAct->PeliSala->protagonistas;
      }else{
```

```
        gotoxy(5,y+4);cout<<"Nombre >>> ";gotoxy(25,y+4);cout<<"    ";
        gotoxy(5,y+6);cout<<"Clasificacion >>> ";gotoxy(25,y+6);cout<<"    ";
        gotoxy(5,y+8);cout<<"Protagonistas >>> ";gotoxy(25,y+8);cout<<"    ";
      }




      listar_sesiones(salaAct->pcabSe, y+10, y);
      //gotoxy(5, y2+y2);cout<<"sexoooooooooooooooooooooooooooo";
      //y2+ =
      salaAct = salaAct->sig;
      y+=8;
    }


    y +=5 ;
    cineAct=cineAct->sig;
  }
}
```

---

### Nombre:

pelis_edades(Pelicula * pcab, Cine * pcabCine)

### Programador(a) Autor(a):

Emilio David Delgado Ladines

### Descripción:

Muestra al usuario todas las películas con una clasificación en común.

### Funcionamiento Breve:

Tiene como objetivo buscar y mostrar información sobre películas en función de su clasificación. Primero, se solicita al usuario que ingrese una clasificación (por ejemplo, "TP", "M15", "M18" o "IFN"). Luego, la función recorre la lista de películas representada por la estructura “Pelicula” a través del puntero “pcab” y busca las películas que coincidan con la clasificación ingresada. Si encuentra una película con la clasificación especificada, muestra detalles como el nombre, la clasificación y los protagonistas de esa película. Luego, recorre la lista de cines representada por la estructura “Cine” a través del puntero “pcabCine”, y para cada cine, muestra las salas y sesiones en las que se proyecta esa película, siempre y cuando esté presente en ese cine. La información se muestra en la pantalla con la función “gotoxy()”.

```
void pelis_edades(Pelicula * pcab, Cine * pcabCine){
  Pelicula * peliAct = pcab;
  system("cls");
  string clasi;
  Cine *cineAct=pcabCine;
  Sala * salaAct;
  menu_pelis();
  gotoxy(5, 5);cout<<"Ingrese la clasificacion a buscar: ";gotoxy(40, 5);cin>>clasi;
  int y = 8;
  clasi= trim(clasi);

  if(clasi!="TP" and clasi!="M15" and clasi!="M18" and clasi!="IFN"){
    gotoxy(5, 17);cout<<"No existe esta clasificacion";
  }else{
    system("cls");
    Pelicula *nodoAct=pcab;
    while(nodoAct!=NULL){
```

```
      if(nodoAct->clasificacion == clasi){
        gotoxy(5,y+2);cout<<"Nombre >>> ";gotoxy(35,y+2);cout<<peliAct->nombre;
        gotoxy(5,y+4);cout<<"Clasificacion >>> ";gotoxy(35,y+4);cout<<peliAct->clasificacion;
        gotoxy(5,y+6);cout<<"Protagonistas >>> ";gotoxy(35,y+6);cout<<peliAct->protagonistas;
        //return nodoAct;
        while(cineAct!=NULL){
        //Presentar los datos
        gotoxy(5, y);cout<<"Cine >>> ";gotoxy(15, y);cout<<cineAct->Nombre;
        salaAct = cineAct->pcabS;

          while(salaAct!=NULL){



            gotoxy(5, y+8);cout<<"Sala >>>";gotoxy(15, y+8 );cout<<salaAct->numero;


            listar_sesiones(salaAct->pcabSe, y+10, y);

            //y2+ =
            salaAct = salaAct->sig;
            y+=8;
          }


        y +=5 ;
        cineAct=cineAct->sig;
      }



      }
        nodoAct=nodoAct->sig;
    }

     //return NULL;
  }
  }
```

---

## Nombre:

menu_pelis()

## Programador(a) Autor(a):

Emilio David Delgado Ladines

## Descripción:

Muestra el menú de clasificaciones para películas disponibles para buscar.

## Funcionamiento Breve:

Usando gotoxy(), muestra el menú de clasificaciones para películas disponibles para buscar.

```
void menu_pelis(){
  gotoxy(50,5);cout<<" -------- Elija una Clasificacion: ---------";
  gotoxy(50,7);cout<<">> Si busca para Todo Publico Escriba   | TP  | ";
  gotoxy(50,9);cout<<">> Si busca para Mayores de 15 Escriba  | M15 | ";
  gotoxy(50,11);cout<<">> Si busca para Mayores de 18 Escriba  | M18 |";
  gotoxy(50,13);cout<<">> Si busca Infantiles Escriba          | IFN |";
}
```

---