PEL

Actividad Colaborativa 3

Marcos Eladio Somoza Corral

Francisco Javier Ruiz Joya

Daniel Gutiérrez Torres

12/12/2021

ÍNDICE

Documentación del ejercicio 1

Enunciado

Una discográfica, recientemente ha decidido actualizar sus sistemas informáticos, pasando a almacenar todos los álbumes con la información correspondiente al título del álbum, grupo al que pertenece y género musical que los representa. Dicho programa, debe tener un sistema de búsquedas, que muestre por consola los resultados de buscar, ya sea el grupo, el título o el género. En caso de no encontrarlo se mostrará un mensaje por pantalla que indique que no se ha podido encontrar el álbum. Para realizar dicho proyecto, los álbumes se almacenarán en un vector, y debe manejarse las búsquedas mediante estos.

Resolución

Para diseñar la arquitectura, se ha creado una clase álbum que contiene los atributos privados de su **nombre**, su **género** y su **grupo**. Además tiene un constructor **album(string _name, string _genre, string _group)** para rellenar dichos atributos.

```
class album
{
private:
    string name;
    string genre;
    string group;
public:
    string getGenre() { return this->genre; }
    album(string _name, string _genre, string _group)
    {   name = _name;
        genre = _genre;
        group = _group; };
    string ToString() { return "Name: " + name + "\n" + "Genre: " + genre
+ "\n" + "Group: " + group; };
    bool IsName(const string &_name) { return name == _name; }
    bool IsGenre(const string &_genre) { return genre == _genre; }
    bool IsGroup(const string &_group) { return group == _group; }
};
```

Cuenta también con un método **ToString()** encargado de devolver un string con toda la información del álbum.Finalmente cuenta con tres métodos que comparan cada uno de sus atributos con el string pasado por referencia. **IsName(const string &_name)** que compara los nombres, **IsGenre(const string &_genre)** que compara los géneros e **IsGroup(const string &_group)** que compara los grupos.

Se ha creado un enum **Search** que contiene **name**, **genre** y **group**; se usará más adelante para diferenciar qué tipo de búsqueda pide el usuario.

```
enum Search{    name,    genre,    group };
```

Además, se ha creado la clase discográfica, encargada de almacenar los álbumes con el atributo privado **std::vector<album> albumes** y gestiona los añadidos y las búsquedas al mismo.

```
class discografica {
private:
    vector<album> albums;
public:
    void AddAlbum(album &&album) { albums.push_back(album); }
    void FindAlbumsBy(Search search) {
        vector<album> found_albums = vector<album>();
        string input;
        switch (search) {
        case name:
            cout << "Name: " << endl; cin >> input;
            for (auto &it : albums)
                if (it.IsName(input))
                    found_albums.push_back(it); break;
        case genre:
            cout << "Genre: " << endl; cin >> input;
            for (auto &it : albums)
                if (it.IsGenre(input))
                    found_albums.push_back(it); break;
        case group:
            cout << "Group: " << endl; cin >> input;
            for (auto &it : albums)
                if (it.IsGroup(input))
                    found_albums.push_back(it); break; }
        if (found_albums.size() > 0) {
            cout << "-------------------------- " << endl
                 << "SE ENCONTRARON LOS ALBUMES " << endl
                 << "-------------------------- " << endl;
            for (auto &it : found_albums)
                cout << it.ToString() << endl << endl; }
        else
            cout << "NO SE ENCONTRO NINGUN ALBUM" << endl << endl;
    };
};
```

Cuenta entonces con el método **AddAlbum(album &&album)** que recibe un objeto **album** como referencia de valor derecho y lo añade al vector **albumes** con **push_back(album)**.
También, para realizar una búsqueda de un álbum se ha creado el método **FindAlbumBy(Search search)** que recibe un **enum search** como parámetro. En función del valor de dicho **enum** entrará en un **case** o otro del **switch**. Cada case funciona similar; pide al usuario el **string** que quiere buscar, recorre el vector **albums** con un iterador (**for auto &it : albums**) y comparará cada elemento it, en base al case en el que esté, con **IsName**, **IsGenre** o **IsGroup**. Si dicha comparación devolviera **true**, añade el elemento a un nuevo vector inicializado al principio **found_albums** del método que almacenará todos los resultados compatibles con la búsqueda. Finalmente, mostrará los elementos encontrados si el tamaño del vector es mayor de cero, o dirá que no se encontró ningún álbum en caso de que no.

Por último, en el **main(int, char**)** se creará una **discográfica discografica discografica_disco_stu** y se le añadirán álbumes variados. También se encuentra el **do-while** encargado del display del menú así como de llamar al método **FindAlbumsBy(Search search)** del objeto discográfica desde un **switch**, en base a la selección del usuario.

```
int main(int, char **)
{
    discografica discografica_disco_stu;
    discografica_disco_stu.AddAlbum(album("el_circulo", "rap", "kase_o"));
    discografica_disco_stu.AddAlbum(album("rojo_y_negro", "rap",
"ayax_y_prok"));
    discografica_disco_stu.AddAlbum(album("rap_sin_corte", "rap",
"foyone"));
    discografica_disco_stu.AddAlbum(album("el_madrileño", "pop",
"c.tangana"));
    discografica_disco_stu.AddAlbum(album("estopa", "rock", "estopa"));
    discografica_disco_stu.AddAlbum(album("el_que_mas", "rock", "obus"));
    discografica_disco_stu.AddAlbum(album("en_un_lugar_de_la_mancha",
"rock", "baron_rojo"));
    discografica_disco_stu.AddAlbum(album("finisterra", "rock",
"mago_de_oz"));

    int selection;
    do
{
        cout << "--------------" << endl
             << "ALBUM SEARCHER" << endl
             << "--------------" << endl;
        cout << "1. Search by name" << endl
             << "2. Search by genre" << endl
             << "3. Search by group" << endl
             << "4. Exit" << endl;
        cin >> selection;
```

```
        switch (selection)
        {
        case 1: discografica_disco_stu.FindAlbumsBy(Search::name); break;
        case 2: discografica_disco_stu.FindAlbumsBy(Search::genre); break;
        case 3: discografica_disco_stu.FindAlbumsBy(Search::group); break;
        default:
            cout << "Seleccion no valida" << endl;
            break;
        }
    }
    while(selection != 4);
}
```

Documentación del ejercicio 2

Enunciado

Un videoclub solicita a vuestra empresa realizar un proyecto por el cual, se puedan cargar nuevos discos en el sistema de alquileres. Para ello, es necesario almacenar la información correspondiente al título de la película, el precio del alquiler y si es un DVD, o un blu-ray. El sistema además tendrá que indicar si el disco está o no alquilado. Este sistema debe permitir buscar los discos mediante su título. Si un título está alquilado, no estará disponible su información, pero si no lo está se mostrará al usuario en pantalla. También debe haber una opción de alquilar, que cambie el estado de un no alquilado a alquilado, y un sistema de devolución, que cambie el estado de alquilado a no alquilado.

Resolución

Para este ejercicio se nos pedía hacer un sistema de gestión de alquiler de películas, con las funcionalidades de alquilar, devolver y ver las películas disponibles para ello.

Se ha empezado definiendo el objeto Película, al que se le han añadido los atributos: título, tipo (para indicar si son Blu-Ray o DVD), precio y un booleano que indica si la película está alquilada o no. Además, se han creado funciones para recoger el título de una película (Se requiere para buscar por título), saber el estado de ésta y para imprimir los datos de la película en cuestión, aparte de los métodos de devolver y alquilar películas.

Ya en la clase principal se declara y crea un vector de objetos película para ir añadiendo las películas que se crean, para facilitar las pruebas se añaden unas películas de ejemplo al empezar. Hay métodos para crear el menú principal con 5 opciones.

Para crear una película y añadirla al catálogo se utiliza la función *pintarMenuCargar ()* que recoge los datos de la película y crea una para añadirla.

Para alquilar una película se llama a la función *alquilerPeliculas (),* en ella se le pide al usuario introducir el nombre de la película y se van buscando en el vector a través de un for, al encontrar la película, le cambia el bool que determina si la película está alquilada o no, lo mismo para devolver películas en la función *devolucionPeliculas ()*.

Documentación del ejercicio optativo

Enunciado

El mes pasado un estudio desarrollado por investigadores británicos revelaba que la protección ofrecida por las dos dosis de las vacunas contra la COVID-19 de Pfizer/BioNTech y AstraZeneca comienza a disminuir transcurridos seis meses desde que se completó la pauta de inmunización.

Como consecuencia de este y otros muchos estudios la universidad Europea de Madrid ha decidido implementar un nuevo sistema informático el cual almacena el número de expediente del alumno, el tipo de vacuna administrada y la fecha de vacunación. Este permitirá acceder a los campos de información mencionados anteriormente de cada estudiante para poder tomar medidas para reducir el riesgo de contagios en el campus universitario. Por ejemplo, si los expertos concluyen que a partir del año de la vacunación con pauta completa, la efectividad de esta disminuye exponencialmente, el sistema deberá ser capaz de buscar de entre todos los alumnos registrados en el sistema aquellos con fecha de inoculación superior al año con objetivo de tomar medidas como la administración de una tercera dosis.

Resolución

Para resolver se ha creado la clase estudiante la cual contendrá atributos de los datos del expediente académico, la vacuna con la vacuna inoculada y su fecha en la que se produjo.

```
class estudiante
{
private:
    string exp;
    string vac;
    string mes;

public:
    string getVac() { return this->vac; }
    inline estudiante(string expediente)
    {
        exp = expediente;
        vac = mes = "";
    }
    estudiante(string expediente, string vacuna, string mesV)
    {
        exp = expediente;
        vac = vacuna;
        mes = mesV;
    };
    string ToString() { return "Número de expediente: " + exp + "\n" + "Vacuna inoculada: " + vac + "\n" + "Mes de vacunación: "
    bool IsExp(const string &expediente) { return exp == expediente; }
    bool IsVac(const string &vacuna) { return vac == vacuna; }
    bool IsMes(const string &mesV) { return mes == mesV; }
};
```

De esta misma manera se ha creado un vector de estudiantes en el cual con los métodos addEstudiante(), se pueden añadir al vector alumnos o findEstudianteBy(), para realizar las búsquedas personalizadas y poder ir iterando y buscando los datos seleccionados en función de la opción solicitada por el usuario y posteriormente introducida en un switch.

```
class sistema
{
private:
    vector<estudiante> vectorEstudiantes;

public:
    void AddEstudiante(estudiante &&estudiante) {vectorEstudiantes.push_back
    void RemoveLastEstudiante() {vectorEstudiantes.pop_back(); }
    void FindEstudianteBy(Search search)
    {
        vector<estudiante> found_estudiante = vector<estudiante>();
        string input;
        switch (search)
        {
        case exp:
            cout << "Numero de expediente: " << endl;
            cin >> input;
            for (auto &it : vectorEstudiantes)
                if (it.IsExp(input))
                    found_estudiante.push_back(it);
            break;
        case vac:
            cout << "Vacuna inoculada: " << endl;
            cin >> input;
            for (auto &it : vectorEstudiantes)
                if (it.IsVac(input))
                    found_estudiante.push_back(it);
            break;
        case mes:
            cout << "Mes de vacunación: " << endl;
            cin >> input;
            for (auto &it : VectorEstudiantes)
                if (it.IsMes(input))
                    found_estudiante.push_back(it);
            break;
        }

        if (found_estudiante.size() > 0)
        {
            cout << "------------------------- " << endl
                 << "SE ENCONTRARON  " << endl
                 << "------------------------- " << endl;
            for (auto &it : found_estudiante)
                cout << it.ToString() << endl
                     << endl;
        }
        else
            cout << "NO SE ENCONTRO NINGUN ESTUDIANTE" << endl
                 << endl;
    };
```

Incorpora otros métodos como toString para devolver la información de los estudiantes o IsVac para realizar las comparaciones y localizar la información seleccionada.

```
do
{
    cout << "--------------" << endl
         << "Selecione el tipo de busqueda que desea realizar:" << endl
         << "--------------" << endl;
    cout << "1. Buscar por numero de expediente" << endl
         << "2. Buscar por vacuna inoculada" << endl
         << "3. Buscar por mes de vacunación" << endl
         << "4. Salir" << endl;
    cin >> selection;
    switch (selection)
    {
    case 1: s.FindEstudiantesBy(Search::exp); break;
    case 2: s.FindEstudiantesBy(Search::vac); break;
    case 3: s.FindEstudiantesBy(Search::mes); break;
    default:
        cout << "Opción introducida no valida" << endl;
        break;
    }
}
while(selection != 4);
```