UNIVERSIDAD SAN CARLOS DE GUATEMALA
CENTRO UNIVERSITARIO DE OCCIDENTE
DIVISIÓN CIENCIAS DE LA INGENIERÍA
ESTRUCTURA DE DATOS
ING. PEDRO LUIS DOMINGO
PRIMER SEMESTRE DE 2023

Proyecto I. Reproductor por Terminal

por

Luis Alejandro Méndez Rivera, 202030627

Martes 11 de Abril, 2023

Reproductor Spotify Chapín (por Terminal)

Este manual contiene la documentación general sobre temas y aspectos relacionados al desarrollo de la Aplicación de Escritorio "Reproductor" solicitada para los siguientes criterios:

- Manejo de atributos/datos (Nodos/Objetos/Listas/Arreglos) en diversos contextos.
- Listas enlazadas, doblemente enlazadas y circulares.
- Pilas y Colas para la resolución de problemas (FIFO Y FILO).
- Punteros (referencias) y memoria.

Listas enlazadas

Una lista enlazada es una estructura de datos lineal en la que los elementos se almacenan en nodos, cada uno de los cuales contiene un valor y un puntero al siguiente nodo de la lista. El último nodo de la lista apunta a null. Esta estructura de datos es muy útil cuando se necesitan insertar o eliminar elementos con frecuencia, ya que no es necesario mover los elementos contiguos para hacer espacio o eliminar un elemento.

Las operaciones más comunes en una lista enlazada son:

- ☐ Insertar un elemento al principio o al final de la lista.
- ☐ Eliminar un elemento de la lista.
- ☐ Buscar un elemento en la lista.
- ☐ Listas doblemente enlazadas

Listas doblemente enlazadas

Las listas doblemente enlazadas son similares a las listas enlazadas, pero cada nodo también tiene un puntero al nodo anterior en la lista. Esto permite que la lista sea recorrida en ambas direcciones. Las operaciones de inserción y eliminación en una lista doblemente enlazada son más eficientes que en una lista enlazada simple.

Las operaciones más comunes en una lista doblemente enlazada son:

- ☐ Insertar un elemento al principio o al final de la lista.
- ☐ Eliminar un elemento de la lista.
- ☐ Buscar un elemento en la lista.
- ☐ Recorrer la lista hacia adelante o hacia atrás.
- ☐ Listas circulares

Listas circulares

Las listas circulares son listas enlazadas en las que el último nodo de la lista apunta al primer nodo en lugar de a null. Esto crea un bucle en la lista, permitiendo que la lista sea recorrida continuamente. Las listas circulares se utilizan a menudo en aplicaciones que necesitan un ciclo continuo, como en la animación.

Las operaciones más comunes en una lista circular son:

- ☐ Insertar un elemento al principio o al final de la lista.
- ☐ Eliminar un elemento de la lista.
- ☐ Buscar un elemento en la lista.
- ☐ Recorrer la lista en cualquier dirección.

En resumen, las listas enlazadas, doblemente enlazadas y circulares son estructuras de datos útiles en diferentes situaciones. Las listas enlazadas son útiles cuando se necesitan insertar o eliminar elementos con frecuencia, mientras que las listas doblemente enlazadas son más eficientes para estas operaciones. Las listas circulares se utilizan a menudo en aplicaciones que necesitan un ciclo continuo.

Pilas

Una pila es una estructura de datos que se utiliza para almacenar una colección de elementos. Los elementos se agregan y eliminan de la pila según el principio FILO, lo que significa que el último elemento agregado es el primero en eliminarse. Las pilas se pueden implementar utilizando un array o una lista enlazada.

Las operaciones básicas que se pueden realizar en una pila son:

- ☐ Push: Agrega un elemento a la pila.
- ☐ Pop: Elimina el elemento superior de la pila.
- ☐ Peek: Obtiene el valor del elemento superior sin eliminarlo.

Las pilas se utilizan en una amplia variedad de aplicaciones, como en la evaluación de expresiones aritméticas, en la implementación de llamadas de funciones en la memoria de una computadora, en la reversión de cadenas de caracteres, entre otras.

Colas

Una cola es una estructura de datos que se utiliza para almacenar una colección de elementos. Los elementos se agregan al final de la cola y se eliminan del principio de la cola, según el principio FIFO, lo que significa que el primer elemento agregado es el primero en eliminarse. Las colas se pueden implementar utilizando un array o una lista enlazada.

Las operaciones básicas que se pueden realizar en una cola son:

- ☐ Enqueue: Agrega un elemento al final de la cola.
- ☐ Dequeue: Elimina el elemento del principio de la cola.
- ☐ Peek: Obtiene el valor del elemento del principio sin eliminarlo.

Las colas se utilizan en una amplia variedad de aplicaciones, como en la gestión de tareas en un sistema operativo, en la implementación de procesamiento por lotes, en la impresión de documentos, entre otras.

Diferencias entre Pilas y Colas

Las pilas y las colas son similares en su estructura básica, ya que ambas son estructuras de datos lineales. Sin embargo, se diferencian en la forma en que se agregan y eliminan los elementos. Las pilas utilizan el principio FILO, mientras que las colas utilizan el principio FIFO. Esto significa que las operaciones de agregado y eliminación se realizan en diferentes extremos de la estructura de datos.

En resumen, las pilas y las colas son dos estructuras de datos fundamentales utilizadas para la resolución de problemas de FILO y FIFO, respectivamente. Las **pilas** se utilizan para almacenar una **colección** de elementos donde el **último elemento agregado** es el **primero en eliminarse**, mientras que las **colas** se utilizan para almacenar una colección de elementos donde el **primer elemento agregado** es el **primero en eliminarse**.

Funcionalidad

La aplicación Reproductor fue dearrollada en el lenguaje de programación C++, por lo cual, previo a poder ejecutar el programa, es necesario descargar e instalar el compilador de C++:

MinGW: https://sourceforge.net/projects/mingw/

Luego de contar con el requisito indispensable que permitirá el análisis de código en C++, podremos compilar el proyecto mediante el siguiente comando:

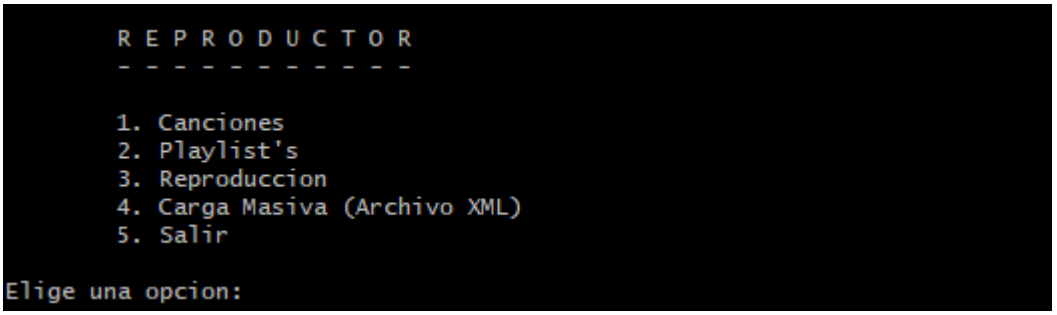
g++ Main.cpp -o Reproductor

Posterior a la compilación se espera un anuncio de compilación sin errores para poder ejecutar el programa. Para lo cual, podremos ejecutar el Reproductor dando doble click en el archivo "Reproductor.exe" (Windows), o mediante el siguiente comando (Linux) desde un terminal situado en la carpeta raíz del proyecto (carpeta que se genera al descomprimir el archivo .ZIP del repositorio que alberga el código fuente):

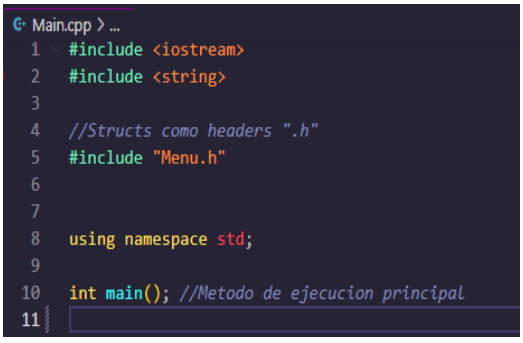
./Reproductor

Programa

El programa al ejecutarse, despliega un Menú principal para navegar entre las diferentes áreas:



Main.cpp



Método de ejecución principal que da inicio al Programa mediante el despliegue del Menú por el método "main()" de la librería/cabecera "Menu.h".

Menu.h

```
C Menu.h > ...
42  // Metodo de Menu principal
43  int main()
44  {
45      // int reproducir();
46      ListaCancionesStore *listaCanciones = new ListaCancionesStore();
47      ListaPlaylists *listaPlaylists = new ListaPlaylists();
48
49      bool bandera = false;
50      char tecla;
51
52      do
53      {
54          system("cls");
55          cout << "\tR E P R O D U C T O R" << endl;
56          cout << "\t- - - - - - - - - - -" << endl
57               << endl;
58          cout << "\t1. Canciones" << endl;
59          cout << "\t2. Playlist's" << endl;
60          cout << "\t3. Reproduccion" << endl;
61          cout << "\t4. Carga Masiva (Archivo XML)" << endl;
62          cout << "\t5. Salir" << endl
63               << endl;
64          cout << "Elige una opcion: ";
65          cin >> tecla;
66
67          switch (tecla)
68          {
69          case '1':
70              system("cls");
71              menuCanciones(listaCanciones);
72
73              pausa();
74              break;
75
76          case '2':
77              system("cls");
78              menuPlaylists(listaPlaylists, listaCanciones);
79
80              pausa();
81              break;
82
83          case '3':
84              system("cls");
85              cout << "FASE NO TERMINADA (EN DESARROLLO)...\n";
86              pausa();
87              break;
88
89          case '4':
90              system("cls");
91              cout << "FASE MENOS TERMINADA (X).\n";
92              pausa();
93              break;
94
95          case '5':
96              system("cls");
97              bandera = true;
98              break;
99
100         default:
101             system("cls");
102             cout << "La Opcion no es valida...\a" << endl;
103             pausa();
104             break;
105         }
106     } while (bandera != true);
107
108     return 0;
109 }
110
```

main(), es la función que contiene el Menú además de la mayoría de implementaciones (de métodos) de diversas librerías/cabeceras para el funcionamiento estructurado por nodos para navegar mediante punteros/referencias dentro de las listas;

Canciones:
Al seleccionar 1, se ejecuta el método menuCanciones() que despliega un nuevo menú que contiene las diversas opciones para manipulación de canciones.

Playlists:
Al seleccionar 2, se ejecuta el método menuPlaylists() que despliega un nuevo menu con diversas operaciones para manipulación de playlists.

Reproducción:
Al seleccionar 3, se ejecuta el método menuReproducción() que despliega un nuevo menu con diversas operaciones para tipos de reproducción.

Carga Masiva:
No se desarrolló.

Salir:
Al seleccionar 5, la ejecución del programa finaliza.

Canciones.h

Cancion.h > Cancion > Cancion(string, string)

You, hace 2 horas | 1 author (You)

```cpp
#include <iostream>
#include <string>

using namespace std;

You, hace 2 horas | 1 author (You)
class Cancion
{
private:
    string nombre;

private:
    string path;

    // Constructor
public:
    Cancion(string nombre, string path)
    {
        this->nombre = nombre;
        this->path = path;
    }

    //Getters y Setters
public:
    string getNombre()
    {
        return this->nombre;
    }

public:
    string getPath()
    {
        return this->path;
    }
};
```

Objeto Canción:
Se conforman de un nombre, y un path en texto.

Nodocancion.h

NodoCancion.h > NodoCancion > NodoCancion(Cancion *, int)

You, hace 2 horas | 1 author (You)

```cpp
#include <iostream>
#include <string>

using namespace std;

You, hace 2 horas | 1 author (You)
struct NodoCancion
{
    NodoCancion *siguiente;
    Cancion *cancion;
    int id;

    // Constructor
    NodoCancion(Cancion *cancion, int id)
    {
        this->siguiente = NULL;
        this->cancion = cancion;
        this->id = id;
    }
};

```

nodoCancion(Cancion *cancion, int id):
Alberga un objeto canción, una referencia al siguiente nodoCancion y su id/index que lo asocia a su posición dentro de una lista (que implementaremos a continuación).

menuCanciones();

Menu.h > menuCanciones(ListaCancionesStore *)

```
116  void menuCanciones(ListaCancionesStore *listaCanciones)
117  {
118      char opc;
119      system("cls");
120      cout << "\t CANCIONES " << endl;
121      cout << "\t- - - - - -" << endl
122           << endl;
123      cout << "\t1. Insertar Cancion";
124      cout << "\t2. Eliminar Cancion";
125      cout << "\t3. Buscar Cancion";
126      cout << "\t4. Listar Canciones";
127      cout << "\t5. Regrsar al Menu" << endl
128           << endl;
129      cout << "\tElige una opcion: ";
130      cin >> opc;
131
132      switch (opc)
133      {
134          system("cls");
135      case '1':
136          insertarCancionAStore(listaCanciones);
137          // pausa();
138          break;
139      case '2':
140          eliminarCancionDeStore(listaCanciones);
141          break;
142      case '3':
143          buscarCancionPorNombre(listaCanciones);
144          break;
145      case '4':
146          listaCanciones->listarCanciones();
147          break;
148
149      default:
150          break;
151      }
152  }
```