

Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías



IL365 - Estructura de Datos - D01

Actividad de Aprendizaje #09

La Lista, Implementación Dinámica Simplemente Ligada

Alumna: Cervantes Araujo Maria Dolores

Código: 217782452

Fecha de Elaboración: 23 marzo de 2023



Autoevaluación			
Concepto	Si	No	Acumulación
Bajé el trabajo de internet o alguien me lo pasó (aunque sea de forma parcial)	-100 pts	0 pts	0
Incluí el código fuente en formato de texto (sólo si funciona cumpliendo todos los requerimientos)	+25pts	0 pts	25
Incluí las impresiones de pantalla (sólo si funciona cumpliendo todos los requerimientos)	+25pts	0 pts	25
Incluí una portada que identifica mi trabajo (nombre, código, materia, fecha, título)	+25 pts	0 pts	25
Incluí una descripción y conclusiones de mi trabajo	+25 pts	0 pts	25
Suma:			100

Introducción:

Para esta actividad, decidí hacer uso del concepto de clases anidadas, por lo que comencé creando la respectiva clase para los nodos de la lista y la clase para excepciones; posteriormente creé una lista simplemente ligada lineal sin encabezado para poder comprender el funcionamiento con respecto a mi programa, una vez realizada la actividad y comprendido la codificación, modifiqué la implementación a una circular, este tipo de listas me permitió realizar una comparación entre funcionamiento con cada tipo de lista enlazada, ya que, si bien el resultado es el mismo, internamente se conlleva otro proceso. Con respecto al programa, dado que era sobre la base de una problemática de actividades anteriores, simplemente se hicieron cambios en el menú y en la forma de definir e implementar la clase de tipo lista, debido a que en esta ocasión decidí utilizar template.

El uso de listas simplemente ligadas ya sea lineal o circular, con o sin encabezado, es debido a que nos permite agregar y eliminar nodos en cualquier punto de la lista en tiempo constante, siempre y cuando estén ya definidos o se puedan localizar; para este punto ya estamos haciendo uso de las listas dinámicas, es decir, manejo de memoria.



Código Fuente:

Lista.hpp

```
#ifndef LISTCIRCULAR_HPP_INCLUDED
#define LISTCIRCULAR_HPP_INCLUDED
#include <string>
#include <iostream>
#include <exception>

template <class T>
class Lista {
private:
    ///Clase anidada de NODO
    class Node {
private:
        T data;
        Node* next;
public:
        Node();
        Node(const T&);

        T getData() const;
        Node* getNext() const;

        void setData(const T&);
        void setNext(Node*);
    };

    Node *ancla; //ancla con apuntador a nodo
    bool validPosition(Node*) const;
    void copyAll(const Lista<T>&);

public:
    typedef Node* Position;

    ///CLASE DE EXCEPTION
    class Exception: public std::exception {
private:
        std::string msg;
public:
        explicit Exception(const char* message): msg(message) {}
        explicit Exception(const std::string& message): msg(message) {}
        virtual ~Exception() throw() {}
        virtual const char* what() const throw() {
            return msg.c_str();
        }
    };

    ///CLASE LISTA
    Lista();
    Lista(const Lista<T>&);
    ~Lista();
```



```
bool listVacia() const;

void insertData(Node*, const T&);
void deleteData(Node*);

//Posiciones de la lista: Primera, Última, Anterior y Siguiente
Node* getFirstPosition() const;
Node* getLastPosition() const;
Node* getBeforePosition(Node*) const;
Node* getNextPosition(Node*) const;
T retrieve(Node*) const;

static int compareByNameMusic(const T&, const T&);
static int compareByInterprete(const T&, const T&);

//Metodos de busqueda lineal
Node* findDatLin(const T&,int (const T&,const T&)) const;
void deleteAll(); //Anula
std::string toString() const;

Lista<T>& operator = (const Lista<T>&);
};

//+++++

using namespace std;
//Implementación NODO

template <class T>
Lista<T>::Node::Node() : next(nullptr) { }

template <class T>
Lista<T>::Node::Node(const T& m) : data(m), next(nullptr) { }

template <class T>
T Lista<T>::Node::getData() const {
    return data;
}

template <class T>
//Lista:: pertenecia de tipo de dato
typename Lista<T>::Node* Lista<T>::Node::getNext() const {
    return next;
}

template <class T>
void Lista<T>::Node::setData(const T& m) {
    data = m;
}

template <class T>
void Lista<T>::Node::setNext(Node* pos) {
    next = pos;
}
```



```
///Implementaci3n LISTA
```

```
template <class T>
```

```
Lista<T>::Lista() : ancla(nullptr) { }
```

```
template <class T>
```

```
Lista<T>::Lista(const Lista& l) : ancla(nullptr) {  
    copyAll(l);  
}
```

```
template <class T>
```

```
Lista<T>::~~Lista() {  
    deleteAll();  
}
```

```
template <class T>
```

```
void Lista<T>::copyAll(const Lista& l) {  
    if(l.listVacia()) {  
        return;  
    }
```

```
    Node* aux(l.ancla);  
    Node* last(nullptr);  
    Node* newNode;
```

```
    do {  
        newNode = new Node(aux->getData());  
        if(newNode == nullptr) {  
            throw Exception("Memoria no disponible");  
        }
```

```
        if(last == nullptr) {  
            ancla = newNode;  
        }
```

```
        else {  
            last->setNext(newNode);  
        }
```

```
        last = newNode;  
        aux = aux->getNext();  
    }
```

```
    while(aux != l.ancla);  
    last->setNext(ancla);  
}
```

```
///LISTA VACIA
```

```
template <class T>
```

```
bool Lista<T>::listVacia() const {  
    return ancla == nullptr;  
}
```

```
template <class T>
```

```
bool Lista<T>::validPosition(Node* pos) const {  
    if(!listVacia()) {  
        Node* aux(ancla);
```



```
do {
    if(aux==pos) {
        return true;
    }
    aux = aux->getNext();
}
while(aux != ancla);
return false;
}

//METODO INSERTAR
template <class T>
void Lista<T>::insertData(Node* pos, const T& music) {
    if(pos != nullptr && !validPosition(pos)) {
        throw Exception("Posicion invalida, insertMusic");
    }

    Node* aux(new Node(music));

    if(aux == nullptr) {
        throw Exception("Memoria no disponible, insertMusic");
    }

    ///Insertar al principio
    if(pos == nullptr) {
        if(listVacia()) { ///Primer nodo en ser insertado
            aux->setNext(aux);
        }
        else { ///Ya hay nodos
            aux->setNext(ancla);
            getLastPosition()->setNext(aux);
        }
        ancla = aux;
    }
    ///Insertar en cualquier otra posicion
    else {
        aux->setNext(pos->getNext());
        pos->setNext(aux);
    }
}

//METODO ELIMINAR
template <class T>
void Lista<T>::deleteData(Node* pos) {
    if(listVacia()) {
        throw Exception("Insuficiencia de datos, deleteMusic");
    }

    if(!validPosition(pos)) {
        throw Exception("Posicion invalida, deleteMusic");
    }
    ///Eliminar el primero
```



```
if(pos == ancla) {
    if(pos->getNext() == pos) { ///Solo queda un nodo en la lista
        ancla = nullptr;
    }
    else { ///Hay mas de un nodo
        getLastPosition()->setNext(ancla->getNext());

        ancla = ancla->getNext();
    }
}

else { ///Eliminar cualquier otro
    getBeforePosition(pos)->setNext(pos->getNext());
}

delete pos;
}

///RECUPERAR
template <class T>
T Lista<T>::retrieve(Node* pos) const {
    if(listVacia()) {
        throw Exception("Insuficiencia de datos, retrieve");
    }
    if(!validPosition(pos)) {
        throw Exception("Cancion invalida, retrieve");
    }
    return pos->getData();
}

///PRIMERA POSICIÓN
template <class T>
typename Lista<T>::Node* Lista<T>::getFirstPosition() const {
    return ancla;
}

///ÚLTIMA POSICIÓN
template <class T>
typename Lista<T>::Node* Lista<T>::getLastPosition() const {
    if(listVacia()) {
        return nullptr;
    }

    Node* aux(ancla);

    while(aux->getNext() != ancla) {
        aux = aux->getNext();
    }
    return aux;
}

///ANTES DE CIERTA POSICIÓN
template <class T>
typename Lista<T>::Node* Lista<T>::getBeforePosition(Node* pos) const {
    if(listVacia() || pos == ancla) {
        return nullptr;
    }
}
```



```
Node* aux(ancla);

do {
    if(aux->getNext() == pos) {
        return aux;
    }

    aux = aux->getNext();
}

while(aux != ancla);

return nullptr;
}

//DESPUÉS DE CIERTA POSICIÓN
template <class T>
typename Lista<T>::Node* Lista<T>::getNextPosition(Node* pos) const {
    if(!validPosition(pos) || pos->getNext() == ancla) {
        return nullptr;
    }
    return pos->getNext();
}

template <class T>
Lista<T>& Lista<T>::operator=(const Lista& l) {
    deleteAll();
    copyAll(l);
    return *this;
}

template <class T>
int Lista<T>::compareByNameMusic(const T& orig, const T& copyc) {
    return orig.getNameMusic().compare(copyc.getNameMusic());
}

template <class T>
int Lista<T>::compareByInterprete(const T& orig, const T& copyc) {
    return orig.getInterprete().compare(copyc.getInterprete());
}

//Busqueda-Localiza
template <class T>
typename Lista<T>::Node* Lista<T>::findDatLin(const T& song, int com(const T&, const
T&)) const {
    if(!listVacia()) {
        Node* aux(ancla);

        do {
            if(com(aux->getData(), song) == 0) {
                return aux;
            }

            aux = aux->getNext();
        }
    }
}
```




```
    }  
    while (aux != ancla);  
}  
return nullptr;  
}
```

```
template<class T>  
string Lista<T>::toString() const {  
    string listComplete;  
    if (!listVacia()) {  
        Node* aux(ancla);  
  
        listComplete += "CANCION";  
        listComplete.resize(30, ' ');  
        listComplete += " || ";  
  
        listComplete += "AUTOR";  
        listComplete.resize(60, ' ');  
        listComplete += " || ";  
  
        listComplete += "INTERPRETE";  
        listComplete.resize(100, ' ');  
        listComplete += " || ";  
  
        listComplete += "Ranking";  
        listComplete.resize(113, ' ');  
        listComplete += " || ";  
  
        listComplete += "MP3";  
        listComplete += "\n\n\n";  
  
        do {  
            listComplete += aux->getData().toString() + "\n";  
  
            aux = aux->getNext();  
        }  
        while (aux != ancla);  
    }  
    return listComplete;  
}
```

///ANULA LA LISTA

```
template <class T>  
void Lista<T>::deleteAll() {  
    if (listVacia()) {  
        return;  
    }  
  
    Node* mark(ancla);  
    Node* aux;  
  
    do {  
        aux=ancla;  
        ancla = ancla->getNext();  
        delete aux;  
    }
```



```
    }  
    while (ancla != mark);  
  
    ancla = nullptr;  
}  
  
#endif // LISTCIRCULAR_HPP_INCLUDED
```

Canción.hpp

```
#ifndef CANCION_HPP_INCLUDED  
#define CANCION_HPP_INCLUDED  
  
#include <string>  
#include <iomanip>  
  
class Cancion {  
private:  
    std::string nameMusic;  
    std::string autor;  
    std::string interprete;  
    std::string mp3;  
    int ranking;  
    void copyAll(const Cancion&);  
  
public:  
    Cancion();  
    bool operator == (const Cancion&) const;  
    bool operator != (const Cancion&) const;  
    bool operator >= (const Cancion&) const;  
    bool operator <= (const Cancion&) const;  
    bool operator > (const Cancion&) const;  
    bool operator < (const Cancion&) const;  
  
    static int compareByNameMusic(const Cancion&, const Cancion&);  
    static int compareByInterprete(const Cancion&, const Cancion&);  
  
    std::string getNameMusic() const;  
    std::string getAutor() const;  
    std::string getInterprete() const;  
    std::string getMp3() const;  
    int getRanking() const;  
  
    void setNameMusic(const std::string&);  
    void setAutor(const std::string&);  
    void setInterprete(const std::string&);  
    void setMp3(const std::string&);  
    void setRanking(const int&);  
  
    std::string toString() const;  
    friend std::ostream& operator >> (std::ostream&, Cancion&);  
    friend std::ostream& operator << (std::ostream&, Cancion&);  
    Cancion& operator = (const Cancion&);
```



```
};
```

```
#endif // CANCION_HPP_INCLUDED
```

Canción.cpp

```
#include "Cancion.hpp"
```

```
#include <iostream>
```

```
using namespace std;
```

```
Cancion::Cancion() { }
```

```
void Cancion::copyAll(const Cancion& song) {  
    nameMusic = song.nameMusic;  
    autor = song.autor;  
    interprete = song.interprete;  
    ranking = song.ranking;  
    mp3 = song.mp3;  
}
```

```
Cancion& Cancion::operator=(const Cancion& song) {  
    nameMusic = song.nameMusic;  
    autor = song.autor;  
    interprete = song.interprete;  
    ranking = song.ranking;  
    mp3 = song.mp3;  
  
    return *this;  
}
```

```
bool Cancion::operator==(const Cancion& c) const {  
    return nameMusic == c.nameMusic || interprete == c.interprete;  
}
```

```
bool Cancion::operator!=(const Cancion& c) const {  
    return nameMusic != c.nameMusic || interprete != c.interprete; //!( *this == c );  
}
```

```
bool Cancion::operator>=(const Cancion& c) const {  
    return nameMusic >= c.nameMusic || interprete >= c.interprete; //!( *this < c );  
}
```

```
bool Cancion::operator<=(const Cancion& c) const {  
    return nameMusic <= c.nameMusic || interprete <= c.interprete; /*this < c ||  
*this == c;  
}
```

```
bool Cancion::operator>(const Cancion& c) const {  
    return nameMusic > c.nameMusic || interprete > c.interprete; //!( *this <= c );  
}
```



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos



```
bool Cancion::operator<(const Cancion& c) const {
    return nameMusic < c.nameMusic || interprete < c.interprete;
}

int Cancion::compareByNameMusic(const Cancion& a, const Cancion& b) {
    return a.nameMusic.compare(b.nameMusic);
}

int Cancion::compareByInterprete(const Cancion& a, const Cancion& b) {
    return a.interprete.compare(b.interprete);
}

string Cancion::getNameMusic() const {
    return nameMusic;
}

string Cancion::getAutor() const {
    return autor;
}

string Cancion::getInterprete() const {
    return interprete;
}

int Cancion::getRanking() const {
    return ranking;
}

string Cancion::getMp3() const {
    return mp3;
}

void Cancion::setNameMusic(const string& songs) {
    nameMusic=songs;
}

void Cancion::setAutor(const string& creador) {
    autor=creador;
}

void Cancion::setInterprete(const string& inter) {
    interprete=inter;
}

void Cancion::setRanking(const int& rang) {
    ranking = rang;
}

void Cancion::setMp3(const string& _mp3) {
    mp3=_mp3;
}

ostream& operator >> (ostream& is, Cancion& obj) {
    string myStr;
```



```
getline(is, obj.nameMusic);
getline(is, obj.autor);
getline(is, obj.interprete);
getline(is, myStr);
obj.ranking = atoi(myStr.c_str());
getline(is, obj.mp3);
return is;
}

ostream& operator << (std::ostream& os, Cancion& obj) {
    os << obj.nameMusic << endl;
    os << obj.autor << endl;
    os << obj.interprete << endl;
    os << obj.ranking << endl;
    os << obj.mp3;
    return os;
}

string Cancion::toString() const {
    string temp;

    temp += nameMusic;
    temp.resize(30, ' ');
    temp += " || ";

    temp += autor;
    temp.resize(60, ' ');
    temp += " || ";

    temp += interprete;
    temp.resize(100, ' ');
    temp += " || ";

    temp += to_string(ranking);
    temp.resize(113, ' ');
    temp += " || ";

    temp += mp3;
    temp += " ";

    return temp;
}
```



Menu.hpp

```
#ifndef MENU_HPP_INCLUDED
#define MENU_HPP_INCLUDED
#include <iostream>
#include "ListCircular.hpp"
#include "Cancion.hpp"
#include <windows.h>

class Menu {
private:
    void visualizacion(Lista<Cancion>&);
    void submenu(Lista<Cancion>&);
    Cancion capture();
public:
    Menu(Lista<Cancion>&);
};

#endif // MENU_HPP_INCLUDED
```

Menu.cpp

```
#include "Menu.hpp"

using namespace std;

Menu::Menu(Lista<Cancion> &viewList) {
    system("Color E5");
    visualizacion(viewList);
}

void Menu::visualizacion(Lista<Cancion> &viewList) {
    int x=0, opc;
    Cancion music;
    string findSong, findinter, song;

    Lista<Cancion>::Position pos;

    while(x==0) {
        system("cls");

        cout<<viewList.toString();

        cout<<"\n\n ---- Compu Radio ----"<<endl<<endl;
        cout<<"[1] Insertar Canciones "<<endl;
        cout<<"[2] Eliminar Canciones "<<endl;
        cout<<"[3] Recuperar Canciones "<<endl;
        cout<<"[4] Busqueda Lineal"<<endl;
        cout<<"[5] Limpiar Playlist "<<endl;
        cout<<"[6] Salir"<<endl;
        cin>>opc;

        switch(opc) {
```



```
case 1:
    system("cls");
    submenu(viewList);
    break;

case 2:
    system("cls");
    cin.ignore();
    cout<<"Cancion que desea eliminar: ";
    getline(cin, song);

    music.setNameMusic(song);
    pos = viewList.findDatLin(music, viewList.compareByNameMusic);
    cout<<endl<<"Eliminando..."<<endl<<endl;
    Sleep(2000);

    if(pos == nullptr) {
        viewList.deleteData(pos);
        cin.get();
    }
    else {
        cout<<"CANCION ELIMINADA"<<endl;
        viewList.deleteData(pos);
        Sleep(2000);
    }
    break;

case 3:
    system("cls");
    cin.ignore();
    cout<<"Cancion que desea recuperar: ";
    getline(cin, song);

    music.setNameMusic(song);
    pos = viewList.findDatLin(music, viewList.compareByNameMusic);
    cout<<endl<<"Buscando..."<<endl<<endl;
    Sleep(2000);

    if(pos == nullptr) {
        viewList.retrieve(pos);
        cin.get();
    }
    else {
        cout<<"\nCANCION ENCONTRADA"<<endl<<endl;
        cout<<"CANCION: " <<endl<<
viewList.retrieve(pos).toString()<<endl<<endl;
    }
    system("PAUSE");
    break;

///Busqueda Lineal
case 4:
    system("cls");
    cin.get();
    cout<<"Busqueda Lineal"<<endl;
```



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos



```
cout<<"[1] Busqueda por Cancion"<<endl;
cout<<"[2] Busqueda por Interprete"<<endl;
cin>>opc;

if(opc==1) {
    cin.ignore();
    cout<<"\nCancion que desea buscar: "<<endl;
    getline(cin, findSong);
    music.setNameMusic(findSong);
    pos = viewList.findDatLin(music, viewList.compareByNameMusic);

}
else if(opc==2) {
    cin.ignore();
    cout<<"Interprete que desea buscar: "<<endl;
    getline(cin, findinter);

    music.setInterprete(findinter);
    pos = viewList.findDatLin(music, viewList.compareByInterprete);

}
else {
    cout<<"Valor invalido"<<endl;
    cin.get();
}

if(pos == nullptr) {
    cout<<"\nNo pudimos encontrar la cancion en la lista :("<<endl;
    Sleep(2000);
}
else {
    Sleep(2000);
    cout<<"\nREGISTRO ENCONTRADO \nEn la posicion "<< pos <<" del
playlist"<<endl<<endl;
    cout<<"Mp3: "<<viewList.retrieve(pos).getMp3()<<endl;
    cin.get();
}
break;

case 5:
    viewList.deleteAll();
    break;

case 6:
    cout<<"HASTA LA PROXIMA!! :D"<<endl;
    x+=1;
    break;

default:
    cout<<"Ingresa solo valores que se te solicitan"<<endl;
    break;
}

if(opc==6) {
    break;
}
}
```




```
}
```

```
void Menu::submenu(Lista<Cancion> &viewList) {
    int opc;
    char x;
    string song;
    Lista<Cancion>::Position pos;
    Cancion music;

    do {
        system("cls");
        cout<<"Elige en que lugar quieres insertar tus canciones"<<endl<<endl;
        cout<<"[1] Primer elemento de la lista "<<endl;
        cout<<"[2] Ultimo elemento de la lista "<<endl;
        cout<<"[3] Antes de cierta cancion de la lista "<<endl;
        cout<<"[4] Despues de cierta cancion de la lista "<<endl;
        cout<<"[5] Regresar al Menu Principal "<<endl;
        cin>>opc;

        switch(opc) {
            case 1:
                music = capture();
                viewList.insertData(viewList.getFirstPosition(), music);
                cout <<"Cancion agregada con exito :D"<<endl<<endl;
                break;

            case 2:
                music = capture();
                viewList.insertData(viewList.getLastPosition(), music);
                cout <<"Cancion agregada con exito :D"<<endl<<endl;
                break;

            case 3:
                cin.ignore();
                cout<<"Antes de cual cancion deseas ingresar otra cancion: ";
                getline(cin, song);

                music.setNameMusic(song);
                pos = viewList.findDatLin(music, viewList.compareByNameMusic);

                if(pos == nullptr) {
                    cout<<"NO EXISTE ESA CANCION\n"<<endl;
                    Sleep(2000);
                }
                else {
                    cout<<"CANCION ENCONTRADA \n\n Inserte cancion:"<<endl;
                    music = capture();
                    viewList.insertData(viewList.getBeforePosition(pos), music);
                    cout <<"Cancion agregada con exito :D"<<endl<<endl;
                }
                break;

            case 4:
                cin.ignore();
```



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos



```
cout<<"Despues de cual cancion deseas ingresar otra cancion: ";
getline(cin, song);

music.setNameMusic(song);
pos = viewList.findDataLin(music, viewList.compareByNameMusic);

if(pos == nullptr) {
    cout<<"NO EXISTE ESA CANCION\n"<<endl;
    Sleep(2000);
}
else {
    cout<<"CANCION ENCONTRADA \n\n Inserte cancion:"<<endl;
    music = capture();
    viewList.insertData(viewList.getNextPosition(pos), music);
    cout <<"Cancion agregada con exito :D"<<endl<<endl;
}
break;

case 5:
    getchar();
    system("cls");
    visualizacion(viewList);
    x+=1;
    break;
default:
    std::cout<<"Ingresa solo valores que se te solicitan"<<std::endl;
    break;
}

cout << "Desea insertar otra cancion? [y/n]: ";
cin >> x;
cin.ignore();
}
while(x== 'y');
}

Cancion Menu::capture() {
    string myStr;
    Cancion music;

    cin.ignore();
    cout<<"\nCancion: ";
    getline(cin, myStr);
    music.setNameMusic(myStr);

    cout<<"Autor: ";
    getline(cin, myStr);
    music.setAutor(myStr);

    cout<<"Interprete: ";
    getline(cin, myStr);
    music.setInterprete(myStr);

    cout<<"Ranking: ";
    getline(cin, myStr);
```



```
music.setRanking(atof(myStr.c_str()));  
  
cout<<"MP3: ";  
getline(cin, myStr);  
music.setMp3(myStr);  
  
return music;  
  
}
```

Main.cpp

```
#include "Menu.hpp"  
int main() {  
    Lista<Cancion> m;  
    Menu start(m);  
}
```

Impresiones de Pantalla:

Inicialización de lista (vacía):

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

```
---- Compu Radio ----  
[1] Insertar Canciones  
[2] Eliminar Canciones  
[3] Recuperar Canciones  
[4] Búsqueda Lineal  
[5] Limpiar PlayList  
[6] Salir  
1_
```

Insertar canciones a la lista:

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

Elige en que lugar quieres insertar tus canciones

```
[1] Primer elemento de la lista  
[2] Ultimo elemento de la lista  
[3] Antes de cierta cancion de la lista  
[4] Despues de cierta cancion de la lista  
[5] Regresar al Menu Principal  
2
```

```
Cancion: Nuestros horarios  
Autor: Sabino  
Interprete: Sabino  
Ranking: 28  
MP3: romanceSab.mp3  
Cancion agregada con exito :D
```

Desea insertar otra cancion? [y/n]: n_



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos



"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

CANCION	AUTOR	INTERPRETE	Ranking	MP3
Colegne	Mario Ortega	Junior H Ovi	27	Junior_Colegne.mp3
Nuestros horarios	Sabino	Sabino	28	romanceSab.mp3

---- Compu Radio ----

- [1] Insertar Canciones
- [2] Eliminar Canciones
- [3] Recuperar Canciones
- [4] Búsqueda Lineal
- [5] Limpiar PlayList
- [6] Salir

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

Elige en que lugar quieres insertar tus canciones

- [1] Primer elemento de la lista
- [2] Ultimo elemento de la lista
- [3] Antes de cierta cancion de la lista
- [4] Despues de cierta cancion de la lista
- [5] Regresar al Menu Principal

3

Antes de cual cancion deseas ingresar otra cancion: Nuestros horarios

CANCION ENCONTRADA

Inserte cancion:

Cancion: Dibujame

Autor: Samantha Barron

Interprete: Samantha Barron & Nanpa

Ranking: 31

MP3: DibSaman.mp3

Cancion agregada con exito :D

Desea insertar otra cancion? [y/n]: y

****Si se desea ingresar canción en posición invalida le arroja el siguiente mensaje:**

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

Elige en que lugar quieres insertar tus canciones

- [1] Primer elemento de la lista
- [2] Ultimo elemento de la lista
- [3] Antes de cierta cancion de la lista
- [4] Despues de cierta cancion de la lista
- [5] Regresar al Menu Principal

4

Despues de cual cancion deseas ingresar otra cancion: La dosis perfecta
NO EXISTE ESA CANCION

Desea insertar otra cancion? [y/n]:



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos



Utilizamos una inyección de datos, quedando de la siguiente manera la lista:

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

CANCION	AUTOR	INTERPRETE	Ranking	MP3
Red velvet	Future	ISA	23	RookieSubs.mp3
Perfect	Ed Sheran	Ed Sheran	20	pop.mp3
Fu	Rihana Mendoza	Miley Cyrus	26	electroSong.mp3
Colegne	Mario Ortega	Junior H Ovi	27	Junior_Colegne.mp3
Un mes a la vez	Aldo Carriola	Aldo Carriola	28	mylove.mp3
When we're high	Loudes Pereida	LP	18	gringasChidad[SubEspañol].mp3
XT4S1S	Danna Paola	Danna Paola	17	nocheloca.mp3
Vienna	Fleetwood Mac	Billy Joel	15	BillyVienna[Español].mp3
Quiero bailar	Ivy Queen	Ivy Queen	14	ParaHacerTareaChido.mp3
Footloose	Kenny Loggins	Kenny Loggins	10	Artistadesconocido.mp3
Noches sin dormir	Aldo Carriola	Aldo Cana	24	Sanchomusic.mp3
En el hood	Micro TDH	Micro TDH	7	partyparty.mp3
Aprovechame	Yoky Barrios	Nanpa Basico & Yoky Barrios	13	NanProvechame.mp3
Otra Cerveza	Carlos Santana	Crasa	6	Atomar.mp3
La chona	Cecilia Velazco	Los Tucanes de Tijuana	4	parabailar.mp3
Like the moon	Gaho	Lee Min Ho	1	moon.mp3
Morena	Lola Perez	Jorge Chavez	5	prietitabonita.mp3
Party	Bennito Ayala	Bad Bunny	16	perreito.mp3
Dibujame	Samantha Barron	Samantha Barron & Nanpa	31	DibSaman.mp3
Nuestros horarios	Sabino	Sabino	28	romanceSab.mp3
La noche mas linda	Albert Carlin	Adalberto Santiago	27	canciondesconocida.mp3
Flowers	Miley Cyrus	Miley Cyrus	2	moodBichota.mp3
Mentiras	Miguel Orlendas	Banda Cuesillos	9	MentirasdeCuesillos.mp3
La gata bajo la lluvia	Rafael Perez Botija	Rocio Durcal	8	melancolicas.mp3
This is what falling in love f	Santino Barrera	JVKE	12	ThisLove.mp3
Ella y yo	Don Omar	Aventura - Don Omar & Romero Santos	11	LaInfidelidadEllayYo.mp3
Mucho para Mi	Franco Escamilla	Santa RM & Franco Escamilla	24	MuchoparaFranco.mp3
Tirando flow sesh #6	Dan Garcia, Bizarrap y mas	Dan Garcia, Ibarra, Rito y mas	19	LosAlucines.mp3
Love shot	Gaho	EXO	21	chinitosuwu.mp3
Lo dire	Christian Burgos	Christian Burgos	22	Lodire[SubEspañol].mp3
Sick of u	BoyWithUke ft. Oliver Tree	Oliver Tree	25	goresong.mp3
Como la flor	Selena Quintanilla	Selena Quintanilla	3	texmex.mp3

---- Compu Radio ----

[1] Insertar Canciones
[2] Eliminar Canciones
[3] Recuperar Canciones
[4] Busqueda Lineal
[5] Limpiar Playlist
[6] Salir

Eliminar canciones de la lista:

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

Cancion que desea eliminar: Colegne

Eliminando...

CANCION ELIMINADA

*Ya no se muestra en la lista:

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

CANCION	AUTOR	INTERPRETE	Ranking	MP3
Red velvet	Future	ISA	23	RookieSubs.mp3
Aprovechame	Yoky Barrios	Nanpa Basico & Yoki Barrios	13	NanProvechame.mp3
Morena	Lola Perez	Jorge Chavez	5	prietitabonita.mp3
Perfect	Ed Sheran	Ed Sheran	20	pop.mp3
Fu	Rihana Mendoza	Miley Cyrus	26	electroSong.mp3
Un mes a la vez	Aldo Carriola	Aldo Carriola	28	mylove.mp3
When we're high	Loudes Pereida	LP	18	gringasChidad[SubEspañol].mp3
XT4S1S	Danna Paola	Danna Paola	17	nocheloca.mp3
Vienna	Fleetwood Mac	Billy Joel	15	BillyVienna[Español].mp3
Quiero bailar	Ivy Queen	Ivy Queen	14	ParaHacerTareaChido.mp3
Footloose	Kenny Loggins	Kenny Loggins	10	Artistadesconocido.mp3
Noches sin dormir	Aldo Carriola	Aldo Cana	24	Sanchomusic.mp3
En el hood	Micro TDH	Micro TDH	7	partyparty.mp3
Aprovechame	Yoky Barrios	Nanpa Basico & Yoky Barrios	13	NanProvechame.mp3
Otra Cerveza	Carlos Santana	Crasa	6	Atomar.mp3
La chona	Cecilia Velazco	Los Tucanes de Tijuana	4	parabailar.mp3
Like the moon	Gaho	Lee Min Ho	1	moon.mp3
Morena	Lola Perez	Jorge Chavez	5	prietitabonita.mp3
La noche mas linda	Albert Carlin	Adalberto Santiago	27	canciondesconocida.mp3
Flowers	Miley Cyrus	Miley Cyrus	2	moodBichota.mp3
Mentiras	Miguel Orlendas	Banda Cuesillos	9	MentirasdeCuesillos.mp3
La gata bajo la lluvia	Rafael Perez Botija	Rocio Durcal	8	melancolicas.mp3
This is what falling in love f	Santino Barrera	JVKE	12	ThisLove.mp3
Ella y yo	Don Omar	Aventura - Don Omar & Romero Santos	11	LaInfidelidadElIayYo.mp3
Mucho para Mi	Franco Escamilla	Santa RM & Franco Escamilla	24	MuchoparaFranco.mp3
Tirando flow sesh #6	Dan Garcia, Bizarrap y mas	Dan Garcia, Ibarra, Rito y mas	19	LosAlucines.mp3
Love shot	Gaho	EXO	21	chinitosuwu.mp3
Lo dire	Christian Burgos	Christian Burgos	22	Lodire[SubEspañol].mp3
Sick of u	BoyWithUke ft. Oliver Tree	Oliver Tree	25	goresong.mp3
Nuestros horarios	Sabino	Sabino	29	romanceSab.mp3
Como la flor	Selena Quintanilla	Selena Quintanilla	3	texmex.mp3

En caso de querer eliminar una canción invalida:

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

Cancion que desea eliminar: Rivers

Eliminando...

```
terminate called after throwing an instance of 'Lista<Cancion>::Exception'
what(): Posicion invalida, deleteMusic
```

En caso de borrar una canción cuando la lista está vacía:

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

Cancion que desea eliminar: Eso y mas

Eliminando...

```
terminate called after throwing an instance of 'Lista<Cancion>::Exception'
what(): Insuficiencia de datos, deleteMusic
```

Recuperar/Consultar canciones de la lista:

 "C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

Cancion que desea recuperar: Like the moon

Buscando...


CANCION ENCONTRADA

CANCION:

Like the moon	Gaho	Lee Min Ho	1	moon.mp3
---------------	------	------------	---	----------

Presione una tecla para continuar . . .

En caso de querer recuperar una canción invalida:


 "C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

Cancion que desea recuperar: Fue en un cafe

Buscando...

```
terminate called after throwing an instance of 'Lista<Cancion>::Exception'
  what(): Cancion invalida, retrieve
```

En caso de querer recuperar una canción cuando la lista está vacía:

 "C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

Cancion que desea recuperar: En el hood

Buscando...

```
terminate called after throwing an instance of 'Lista<Cancion>::Exception'
  what(): Insuficiencia de datos, retrieve
```

Búsqueda Lineal Por Canción:

Haremos la búsqueda de una canción que no exista, de FU y Noches Sin Dormir:

1

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Del

Busqueda Lineal

[1] Busqueda por Cancion

[2] Busqueda por Interpretete

1

Cancion que desea buscar:

TOM

No pudimos encontrar la cancion en la lista :(



2

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"  
Busqueda Lineal  
[1] Busqueda por Cancion  
[2] Busqueda por Interprete  
1  
  
Cancion que desea buscar:  
Fu  
  
REGISTRO ENCONTRADO  
En la posicion 0x26329e0 del playlist  
  
Mp3: electroSong.mp3
```

3

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"  
Busqueda Lineal  
[1] Busqueda por Cancion  
[2] Busqueda por Interprete  
1  
  
Cancion que desea buscar:  
Noches sin dormir  
  
REGISTRO ENCONTRADO  
En la posicion 0x26335c0 del playlist  
  
Mp3: Sanchomusic.mp3
```

Búsqueda Lineal Por Interprete:

Haremos la búsqueda del intérprete de Sabino, de Miley Cyrus y de un intérprete que no esté en la lista:

1

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Act  
Busqueda Lineal  
[1] Busqueda por Cancion  
[2] Busqueda por Interprete  
2  
Interprete que desea buscar:  
Sabino  
  
REGISTRO ENCONTRADO  
En la posicion 0xe22e40 del playlist  
  
Mp3: romanceSab.mp3
```

2

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"  
Busqueda Lineal  
[1] Busqueda por Cancion  
[2] Busqueda por Interprete  
2  
Interprete que desea buscar:  
Miley Cyrus  
  
REGISTRO ENCONTRADO  
En la posicion 0xe22f80 del playlist  
  
Mp3: electroSong.mp3
```

3

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"  
Busqueda Lineal  
[1] Busqueda por Cancion  
[2] Busqueda por Interprete  
2  
Interprete que desea buscar:  
Los terricolas  
  
No pudimos encontrar la cancion en la lista :(
```




Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos



Anular-Eliminar Lista:

Antes:

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

CANCION	AUTOR	INTERPRETE	Ranking	MP3
Red velvet	Future	ISA	23	RookieSubs.mp3
Aprovechame	Yoky Barrios	Nanpa Basico & Yoki Barrios	13	NanProvechame.mp3
Morena	Lola Perez	Jorge Chavez	5	prietitabonita.mp3
Perfect	Ed Sheran	Ed Sheran	20	pop.mp3
Fu	Rihana Mendoza	Miley Cyrus	26	electroSong.mp3
Un mes a la vez	Aldo Carriola	Aldo Carriola	28	mylove.mp3
When we're high	Loudes Pereida	LP	18	gringasChidad[SubEspañol].mp3
XT4S1S	Danna Paola	Danna Paola	17	nocheloca.mp3
Vienna	Fleetwood Mac	Billy Joel	15	BillyVienna[Español].mp3
Quiero bailar	Ivy Queen	Ivy Queen	14	ParaHacerTareaChido.mp3
Footloose	Kenny Loggins	Kenny Loggins	10	Artistadesconocido.mp3
Noches sin dormir	Aldo Carriola	Aldo Cana	24	Sanchomusic.mp3
En el hood	Micro TDH	Micro TDH	7	partyparty.mp3
Aprovechame	Yoky Barrios	Nanpa Basico & Yoky Barrios	13	NanProvechame.mp3
Otra Cerveza	Carlos Santana	Crasa	6	Atomar.mp3
La chona	Cecilia Velazco	Los Tucanes de Tijuana	4	parabailar.mp3
Like the moon	Gaho	Lee Min Ho	1	moon.mp3
Morena	Lola Perez	Jorge Chavez	5	prietitabonita.mp3
La noche mas linda	Albert Carlin	Adalberto Santiago	27	canciondesconocida.mp3
Flowers	Miley Cyrus	Miley Cyrus	2	moodBichota.mp3
Mentiras	Miguel Orlendas	Banda Cuesillos	9	MentirasdeCuesillos.mp3
La gata bajo la lluvia	Rafael Perez Botija	Rocio Durcal	8	melancolicas.mp3
This is what falling in love f	Santino Barrera	JVKE	12	ThisLove.mp3
Ella y yo	Don Omar	Aventura - Don Omar & Romero Santos	11	LaInfidelidadEllayYo.mp3
Mucho para Mi	Franco Escamilla	Santa RM & Franco Escamilla	24	MuchoparaFranco.mp3
Tirando flow sesh #6	Dan Garcia, Bizarrap y mas	Dan Garcia, Ibarra, Rito y mas	19	LosAlucines.mp3
Love shot	Gaho	EXO	21	chinitosuwu.mp3
Lo dire	Christian Burgos	Christian Burgos	22	Lodire[SubEspañol].mp3
Sick of u	BoyWithUke ft. Oliver Tree	Oliver Tree	25	goresong.mp3
Nuestros horarios	Sabino	Sabino	29	romanceSab.mp3
Como la flor	Selena Quintanilla	Selena Quintanilla	3	texmex.mp3

---- Compu Radio ----

[1] Insertar Canciones
[2] Eliminar Canciones
[3] Recuperar Canciones
[4] Busqueda Lineal
[5] Limpiar Playlist
[6] Salir
5

Después:

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

---- Compu Radio ----

[1] Insertar Canciones
[2] Eliminar Canciones
[3] Recuperar Canciones
[4] Busqueda Lineal
[5] Limpiar Playlist
[6] Salir



Finalmente cerramos el programa

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad9\bin\Debug\Actividad9.exe"

```
---- Compu Radio ----  
[1] Insertar Canciones  
[2] Eliminar Canciones  
[3] Recuperar Canciones  
[4] Búsqueda Lineal  
[5] Limpiar Playlist  
[6] Salir  
6  
HASTA LA PROXIMA!! :D  
  
Process returned 0 (0x0)   execution time : 548.490 s  
Press any key to continue.
```

Resumen Personal

Al momento de cambiar a plantillas, me generaba errores para ingresar y mostrar los datos, ya que insertaba, pero no me los mostraba en la lista con el toString y con la lista simplemente ligada circular no me dejaba agregar más de tres valores a la lista, no fue hasta que modifiqué el método de captura a un objeto de tipo canción que pude solucionarlo.

Debido a que retomamos un trabajo previo, solo hice la optimización a plantillas y en la lista me propuse el reto de crear una lista simplemente ligada sin encabezado lineal y otra circular para practicar; esto me ayudo para lograr comprender mejor las diferencias entre estas listas al momento de manipular los datos internamente, debido a que el algoritmo varía un poco una de la otra.

Concluyo que es mejor el uso de plantillas cuando hablamos de trabajos cada vez más complejos, te facilita el uso de datos entre listas; y para el uso de listas simplemente ligadas considero que depende mucho el sentido que le quieras dar a tu lista, pero básicamente todas son óptimas; conforme avanzamos en el semestre siento que voy generando buenas prácticas de programación y desarrollando más el sentido lógico.