

Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías



IL365 - Estructura de Datos - D01

Actividad de Aprendizaje #10

La Lista, Implementación Dinámica Doblemente Ligada

Alumna: Cervantes Araujo Maria Dolores

Código: 217782452

Fecha de Elaboración: 27 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:

Esta semana tomamos de referencia el trabajo anterior y realizamos cambios en la implementación de lista, así como agregar el guardado y lectura a disco. Utilice una lista doblemente ligada circular con encabezado “Dummy”; primero realicé una lista doblemente ligada lineal, y de ahí como ya tenía la circular, pase a una circular con encabezado para poder comprender su funcionamiento en relación con el programa.

La lista doblemente ligada circular con encabezado “Dummy” es un poco más cómoda para el manejo dinámico de datos, es como una lista lineal en la que cada nodo tiene dos enlaces; en este tipo de listas nos olvidamos del “ancla” que era lo que veníamos manejando anteriormente, en su lugar creamos un encabezado, lo que es clave para realizar todas las modificaciones a lo largo de la lista.



Código Fuente:

Lista.hpp

```
#ifndef LISTDUMMY_HPP_INCLUDED
#define LISTDUMMY_HPP_INCLUDED
#include <string>
#include <iostream>
#include <exception>
#include <fstream>

template <class T>
class Lista {
private:
    ///Clase anidada de NODO
    class Node {
private:
        T* dataPtr;
        Node* prev;
        Node* next;

public:
        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();
            }
        };

        Node();
        Node(const T&);
        ~Node();

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

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

    Node* header;
    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 isEmpty() const;

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

    Node* getFirstPosition() const;
    Node* getLastPosition() const;
    Node* getBeforePosition(Node*) const;
    Node* getNextPosition(Node*) const;
    T retrieve(Node*) const;
    Node* localiza(const T&) const;

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

    Node* findDatLin(const T&,int (const T&,const T&)) const;
    void deleteAll();
    std::string toString() const;

    ///Disk
    void writeFromDisk(const std::string&);
    void readFromDisk(const std::string&);

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

///+++++
```



```
using namespace std;
///-----Implementación NODO

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

template <class T>
Lista<T>::Node::Node(const T& m) : dataPtr(new T(m)), prev(nullptr), next(nullptr)
{
    if(dataPtr == nullptr) {
        throw Exception("Memoria insuficiente, se creara un nodo");
    }
}

template <class T>
Lista<T>::Node::~~Node() {
    delete dataPtr;
}

template <class T>
T* Lista<T>::Node::getDataPtr() const {
    return dataPtr;
}

template <class T>
T Lista<T>::Node::getData() const {
    if(dataPtr == nullptr) {
        throw Exception("Dato inexistente, getData");
    }
    return *dataPtr; //desferenciacion para obtener el dato
}

template <class T>
typename Lista<T>::Node* Lista<T>::Node::getPrev() const {
    return prev;
}

template <class T>
typename Lista<T>::Node* Lista<T>::Node::getNext() const {
    return next;
}

template <class T>
void Lista<T>::Node::setDataPtr(const T* pos) {
    dataPtr = pos;
}

template <class T>
void Lista<T>::Node::setData(const T& e) {
    if(dataPtr == nullptr) {
        if((dataPtr = new T(e)) == nullptr) {
            throw Exception("Memoria no disponible, setData");
        }
    }
}
```



```
    }  
    else {  
        *dataPtr = e;  
    }  
}  
  
template <class T>  
void Lista<T>::Node::setPrev(Node* pos) {  
    prev = pos;  
}  
  
template <class T>  
void Lista<T>::Node::setNext(Node* pos) {  
    next = pos;  
}  
  
///-----Implementaci³n LISTA  
//Constructor  
template <class T>  
Lista<T>::Lista() : header(new Node) {  
    if(header == nullptr) {  
        throw Exception("Memoria no disponible, inicializando lista...");  
    }  
  
    header->setPrev(header);  
    header->setNext(header);  
}  
  
template <class T>  
Lista<T>::Lista(const Lista& l) : Lista() {  
    copyAll(l);  
}  
  
template <class T>  
Lista<T>::~~Lista() {  
    deleteAll();  
    delete header;  
}  
  
template <class T>  
void Lista<T>::copyAll(const Lista& l) {  
    Node* aux(l.header->getNext());  
    Node* newNode;  
  
    while(aux != l.header) {  
        try {  
            if((newNode = new Node(aux->getData())) == nullptr) {  
                throw Exception("Memoria no disponible, copyAll");  
            }  
        }  
        catch(typename Node::Exception ex) {  
            throw Exception(ex.what());  
        }  
    }  
}
```



```
newNode->setPrev(header->getPrev());
newNode->setNext(header);

header->getPrev()->setNext(newNode);
header->setPrev(newNode);

aux = aux->getNext();
}
}

///LISTA VACIA
template <class T>
bool Lista<T>::isEmpty() const {
    return header->getNext() == header;
}

template <class T>
bool Lista<T>::validPosition(Node* pos) const {
    Node* aux(header->getNext());

    while(aux != header) {
        if(aux == pos) {
            return true;
        }
        aux = aux->getNext();
    }
    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, insertData");
    }

    Node* aux;
    try {
        aux = new Node(music);
    }
    catch (typename Node::Exception ex) {
        throw Exception(ex.what());
    }

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

    //Insertar en el encabezado
    if(pos == nullptr) {
        pos = header;
    }
}
```



```
aux->setPrev(pos);  
aux->setNext(pos->getNext());  
  
pos->getNext()->setPrev(aux);  
pos->setNext(aux);  
}
```

///METODO ELIMINAR

```
template <class T>  
void Lista<T>::deleteData(Node* pos) {  
    if(!validPosition(pos)) {  
        throw Exception("Posicion invalida, deleteMusic");  
    }  
    pos->getPrev()->setNext(pos->getNext());  
    pos->getNext()->setPrev(pos->getPrev());  
    delete pos;  
}
```

///RECUPERAR

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

///Localiza

```
template <class T>  
typename Lista<T>::Node* Lista<T>::localiza(const T& music) const {  
    Node* aux(header->getNext());  
  
    while(aux != header) {  
        if(aux->getData() == music) {  
            return aux;  
        }  
        aux = aux->getNext();  
    }  
    return nullptr;  
}
```

///PRIMERA POSICIÓN

```
template <class T>  
typename Lista<T>::Node* Lista<T>::getFirstPosition() const {  
    if(isEmpty()) {  
        return nullptr;  
    }  
    return header->getNext();  
}
```

///ÚLTIMA POSICIÓN

```
template <class T>
```




```
typename Lista<T>::Node* Lista<T>::getLastPosition() const {
    if (isEmpty()) {
        return nullptr;
    }
    return header->getPrev();
}

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

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

template <class T>
typename Lista<T>::Node* Lista<T>::findDatLin(const T& song, int com(const T&,
const T&)) const {
    Node* aux(header->getNext());
```



```
while(aux != header) {
    if(com(aux->getData(), song) == 0) {
        return aux;
    }
    aux = aux->getNext();
}
return nullptr;
}

///Escritura a disco
template <class T>
void Lista<T>::writeFromDisk(const string& fileName) {
    ofstream myFile;

    myFile.open(fileName, ios_base::trunc);

    if(!myFile.is_open()) {
        throw Exception("No se puede abrir el archivo para escritura,
writeFromDisk");
    }

    Node* aux = header->getNext();
    T data;

    while(aux != header) {
        data = aux->getData();
        myFile << data << endl;
        aux = aux->getNext();
    }
    myFile.close();
}

///Lectura a disco
template <class T>
void Lista<T>::readFromDisk(const string& fileName) {
    ifstream myFile;

    myFile.open(fileName);
    if(!myFile.is_open()) {
        throw Exception("No se pudo abrir el archivo para lectura, readFromDisk");
    }

    deleteAll();
    T myData;
    try {
        while(myFile >> myData) { //!myFile.eof()
            insertData(getLastPosition(), myData);
        }
    }
    catch(Exception ex) {
        myFile.close();
        throw Exception(ex.what());
    }
}
```



```
myFile.close();
}

template<class T>
string Lista<T>::toString() const {
    Node* aux(header->getNext());
    string listComplete;

    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";

    while(aux != header) {
        listComplete += aux->getData().toString() + "\n";
        aux = aux->getNext();
    }
    return listComplete;
}

template <class T>
void Lista<T>::deleteAll() {
    Node* aux;

    while(header->getNext() != header) {
        aux = header->getNext();
        header->setNext(aux->getNext());
        delete aux;
    }
    header->setPrev(header);
}

#endif // LISTDUMMY_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::istream& operator >> (std::istream&, 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) {
    copyAll(song);
    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 );
}

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;
}

istream& operator >> (istream& 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 "ListDummy.hpp"  
#include "Cancion.hpp"  
#include <windows.h>  
  
class Menu {  
    private:  
        void visualizacion(Lista<Cancion>&);  
        void submenu(Lista<Cancion>&);  
        void writeFromDisk(Lista<Cancion>&);  
        void readFromDisk(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();  
        //readFromDisk(viewList);
```

```
        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] Leer Canciones del Disco"<<endl;  
        cout<<"[6] Guardar Canciones del Disco"<<endl;  
        cout<<"[7] Limpiar PlayList "<<endl;  
        cout<<"[8] 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;
    cout<<"[1] Busqueda por Cancion"<<endl;
    cout<<"[2] Busqueda por Interpretador"<<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:  
    readFromDisk(viewList);  
    break;  
  
case 6:  
    writeFromDisk(viewList);  
    break;  
  
case 7:  
    viewList.deleteAll();  
    break;  
  
case 8:  
    cout<<"HASTA LA PROXIMA!! :D"<<endl;  
    x+=1;  
    break;  
  
default:  
    cout<<"Ingresa solo valores que se te solicitan"<<endl;  
    break;  
}  
if(opc==8) {  
    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();  
cout<<"Despues 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.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;
}

void Menu::writeFromDisk(Lista<Cancion>& viewList) {
    cout<<"Escribiendo al disco..."<<endl;
    viewList.writeFromDisk("Songs.txt");
    cout<<"\nAccion Completada"<<endl;
    system("PAUSE");
    system("cls");
}

void Menu::readFromDisk(Lista<Cancion>& viewList) {
    cout<<"Leyendo del disco..."<<endl;
    viewList.readFromDisk("Songs.txt");
    Sleep(2000);
    system("cls");
}
```

Main.cpp

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



Impresiones de Pantalla:

Inicialización de lista:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad10\bin\Debug\Actividad10.exe"
CANCION || AUTOR || INTERPRETE || Ranking || MP3

---- Compu Radio ----

[1] Insertar Canciones
[2] Eliminar Canciones
[3] Recuperar Canciones
[4] Busqueda Lineal
[5] Leer Canciones del Disco
[6] Guardar Canciones del Disco
[7] Limpiar Playlist
[8] Salir
```

Insertar canciones a la lista:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad10\bin\Debug\Actividad10.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
```

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad10\bin\Debug\Actividad10.exe"
CANCION || AUTOR || INTERPRETE || Ranking || MP3
Nuestros horarios || Sabino || Sabino || 28 || romanceSab.mp3

---- Compu Radio ----

[1] Insertar Canciones
[2] Eliminar Canciones
[3] Recuperar Canciones
[4] Busqueda Lineal
[5] Leer Canciones del Disco
[6] Guardar Canciones del Disco
[7] Limpiar Playlist
[8] Salir
```



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\Actividad10\bin\Debug\Actividad10.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\Actividad10\bin\Debug\Actividad10.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]:

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

Seleccionar "C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad10\bin\Debug\Actividad10.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	Wiley Cyrus	26	electroSong.mp3
Cologne	Mario Ortega	Junior H Ovi	30	Junior_Cologne.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	Wiley Cyrus	Wiley 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	chinitosuuu.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] Leer Canciones del Disco
- [6] Guardar Canciones del Disco
- [7] Limpiar Playlist
- [8] Salir



Guardado a Disco:

```
---- Compu Radio ----  
  
[1] Insertar Canciones  
[2] Eliminar Canciones  
[3] Recuperar Canciones  
[4] Busqueda Lineal  
[5] Leer Canciones del Disco  
[6] Guardar Canciones del Disco  
[7] Limpiar PlayList  
[8] Salir  
6  
Escribiendo al disco...  
  
Accion Completada  
Presione una tecla para continuar . . .
```

Songs

27/03/2023 01:13 p. m.

Documento de texto

3 KB

Lectura a disco:

```
---- Compu Radio ----  
  
[1] Insertar Canciones  
[2] Eliminar Canciones  
[3] Recuperar Canciones  
[4] Busqueda Lineal  
[5] Leer Canciones del Disco  
[6] Guardar Canciones del Disco  
[7] Limpiar PlayList  
[8] Salir  
5  
Leyendo del disco...
```

Quedando de la siguiente manera:

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad10\bin\Debug\Actividad10.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	
Cologne	Mario Ortega	Junior H Ovi	30	Junior_Cologne.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	Artistasdesconocido.mp3	
Noches sin dormir	Aldo Carriola	Aldo Cana	24	Sancomusic.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	mood81chota.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	chinitosuuu.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	



Eliminar canciones:

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad10\bin\Debug\Actividad10.exe"

Cancion que desea eliminar: Colegne

Eliminando...

CANCION ELIMINADA

Lista actualizada:

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad10\bin\Debug\Actividad10.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	Rihanna 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	ParaHacerTareasChido.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	chinitosuuu.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\Actividad10\bin\Debug\Actividad10.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\Actividad10\bin\Debug\Actividad10.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\Actividad10\bin\Debug\Actividad10.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\Actividad10\bin\Debug\Actividad10.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\Actividad10\bin\Debug\Actividad10.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:



```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad10\bin\Del  
Busqueda Lineal  
[1] Busqueda por Cancion  
[2] Busqueda por Interprete  
1  
  
Cancion que desea buscar:  
TQM  
  
No pudimos encontrar la cancion en la lista :(
```



2

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad10\bin\Debug\Actividad10.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\Actividad10\bin\Debug\Actividad10.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\Actividad10\bin\Debug\Acti

```
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\Actividad10\bin\Debug\Actividad10.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\Actividad10\bin\Debug\Actividad10.exe"

```
Busqueda Lineal
[1] Busqueda por Cancion
[2] Busqueda por Interprete
2

Interprete que desea buscar:
Los terrícolas

No pudimos encontrar la cancion en la lista :(
```



Anular-Eliminar Lista:

Antes:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad10\bin\Debug\Actividad10.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   | Miley Cyrus     | 20      | pop.mp3
Fu           | Rihanna 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      | Artistasdesconocido.mp3
Noches sin dormir | Aldo Carriola | Aldo Carriola   | 24      | Sanchomusic.mp3
En el hood   | Micro TDM   | Micro TDM       | 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      | LaInfidelidadEllyayYo.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      | Lodine[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] Leer Canciones del Disco
[6] Guardar Canciones del Disco
[7] Limpiar Playlist
[8] Salir
7
```

Después:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Actividad10\bin\Debug\Actividad10.exe"

CANCION      | | AUTOR      | | INTERPRETE      | | Ranking | | MP3

---- Compu Radio ----

[1] Insertar Canciones
[2] Eliminar Canciones
[3] Recuperar Canciones
[4] Busqueda Lineal
[5] Leer Canciones del Disco
[6] Guardar Canciones del Disco
[7] Limpiar Playlist
[8] Salir
```

Finalmente cerramos el programa

```
8
HASTA LA PROXIMA!! :D

Process returned 0 (0x0)   execution time : 103.559 s
Press any key to continue.
```



Resumen Personal-Conclusión

Fue una práctica rápida dado que en la anterior comprendí en su totalidad el funcionamiento de las listas simplemente ligadas, para esta actividad solo fue cuestión de realizar modificaciones para las listas doblemente ligadas guiándome del pseudocódigo proporcionado en la plataforma y complementando con el guardado y lectura a disco. En lo personal concluyo que el modelo de la lista “Dummy” considero que es el más práctico de todos los que estuvimos viendo a lo largo de estas dos semanas, ya que te ahorras excepciones creando una o dos que te cubren para casi todo el algoritmo, menos líneas de código con mayor eficiencia.

Lo único que me genero conflicto fue para la lectura de disco, que si bien no es un requerimiento que pida la actividad decidí implementarlo para ver su uso mediante las listas y así poderlo implementar más rápido a la hora de crear el proyecto final; me llevo de retroalimentación el recordar no solo sobrecargar los operadores, sino también recordar retornar el valor, de ahí en más no tuve otras complejidades para el algoritmo del programa.