CUCEI

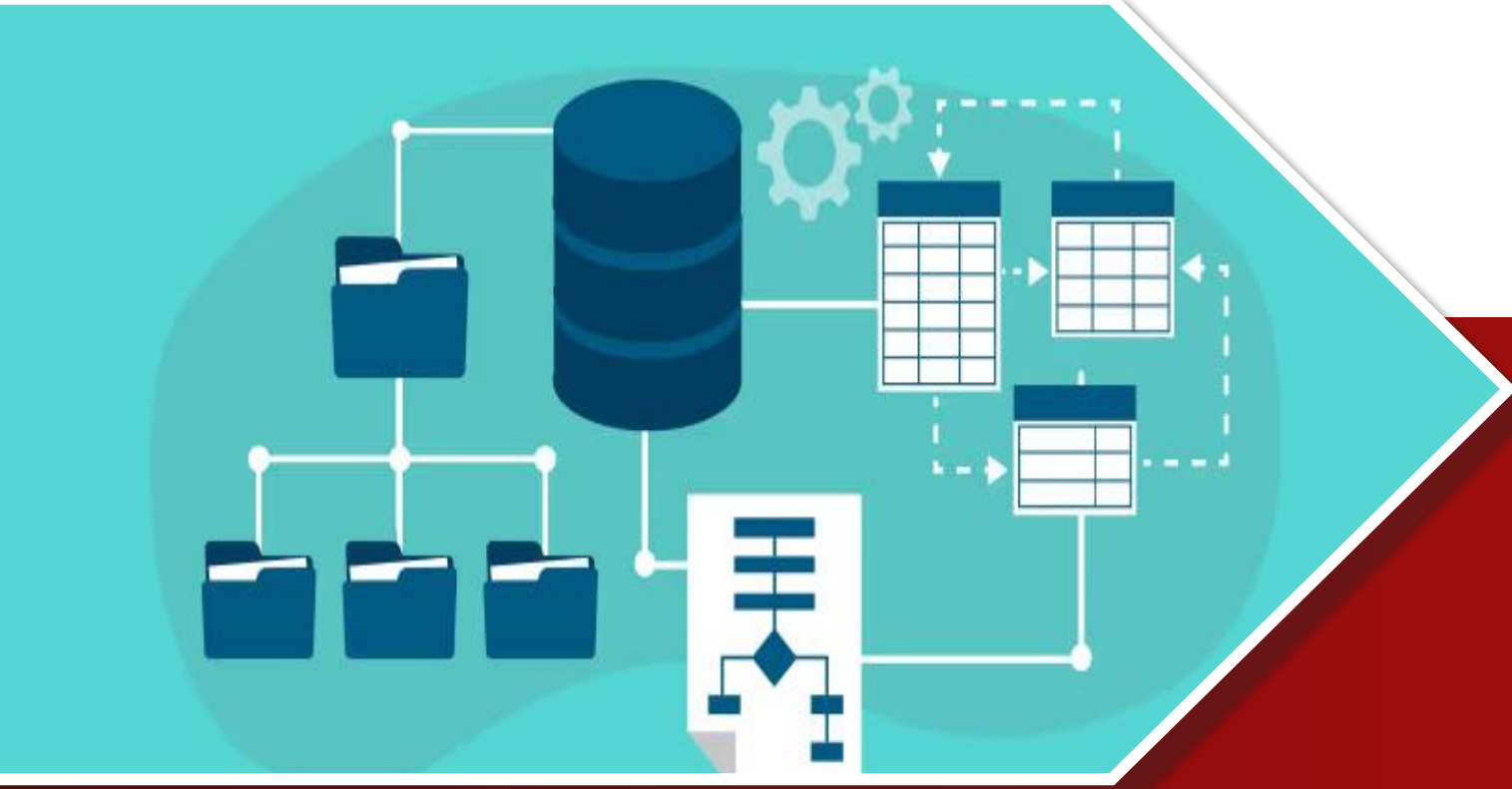
Actividad de Aprendizaje 09

Apuntadores

Centro Universitario de Ciencias Exactas e Ingenierías

Materia: Estructuras de Datos

Clave: V0731 ***Sección***: D02



Alumno: Mariscal Rodríguez Omar Jesús

Código: 220858478

Profesor: Gutiérrez Hernández Alfredo

Fecha: 12 de Octubre de 2025

Contenido

Test de Autoevaluación

Autoevaluación			
Concepto	***Sí***	***No***	***Acumulado***
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)***	***+25 pts***	***0 pts***	***25***
Incluí las ***impresiones de pantalla (sólo si funciona cumpliendo todos los requerimientos)***	***+25 pts***	***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 y Abordaje del Problema

El propósito de esta actividad fue practicar con los apuntadores en el lenguaje C++, como contenedores de una dirección de memoria que se interpreta y comporta como un tipo de dato que nosotros se lo indiquemos. Para ello, cambiaremos la implementación de la lista estática que almacena canciones por una lista que contiene un arreglo que almacena direcciones de memoria, con toda la responsabilidad e implicaciones que ello conlleva.

Planteamiento del Problema

Para esta actividad, es muy importante revisar a fondo la implementación de la lista que tenemos, ya que el manejo de punteros es más delicado que el de los otros tipos de datos primitivos; esto puede ocasionar comportamientos inesperados y cierres repentinos del programa (e ahí, la importancia de inicializar los punteros en nullptr cuando no estén en uso), también considerando la posibilidad de tener fugas de memoria en el programa, esto cuando no se libera un puntero correctamente y perdemos la dirección de memoria de esa información (ya que a lo que "apunta" un apuntador no es una variable en el sentido estricto de la palabra); por ello, esto es una revisión exhaustiva del código planeando cuando los punteros almacenarán la referencia, cuando eliminarlos y demás.

Lo primero que se verá afectado es el arreglo que almacena la lista, ya no será solamente T, sino que será un arreglo de T* (apuntadores a T), pero como lo crearemos dinámicamente, data[] se convertirá en T** data, es decir, un apuntador a un apuntador del tipo T, esto porque data se comportará como un arreglo. Podemos manejarlo así ya que un arreglo no deja de ser espacios en memoria contiguos, y con la aritmética de apuntadores que hace internamente C++, podemos recorrer el data como si fuera un arreglo convencional. Las principales funciones que se verán afectadas por estos cambios serán las de insertar y eliminar elementos, ya no almacenan un elemento T (canción en este programa), sino que se creará mediante new el elemento T y lo que guardaremos en data será la dirección de memoria sobre la cual vamos a poder acceder a todos sus métodos. Esto tiene muchas ventajas, siendo una de las principales en las asignaciones, para mover un elemento de una posición del arreglo a otra ya no tenemos que trasladar todo un objeto con toda su información en él, sino que ahora lo que moveremos serán las direcciones de memoria (de 64 bits en el caso estándar), siendo

mucho más sencillo en costo computacional y considerando posteriormente objetos más pesados.

Como buena práctica, vamos a hacer que el constructor inicialice todos los elementos del arreglo en Nullptr, y el destructor lo que hará será liberar toda la memoria que se haya estado utilizando.

En funciones de ordenamientos, optaremos por desreferenciar los punteros para poder acceder a la información de las canciones, el operador de asignación y copyAll() también se adatarán para trabajar con apuntadores; así como copyAll() nos fue de mucha utilidad para reciclar código, ahora crearemos una función llamada freeAll(), que lo que hará es liberar todos los apuntadores existentes e inicializarlos en Nullptr, esta será la función que usaremos antes del copyAll() para no dejar direcciones y memoria colgada. Una de las cosas que vamos a hacer es procurar el modelo de la lista, es decir, no haré que reciba punteros o algo así, en primer lugar, ya tenemos un programa montado detrás, esta actualización debe seguir siendo compatible con ello sin requerir una refactorización, y segundo, la lista en un tipo de dato abstracto, por lo que nos basamos en el modelo, modificamos la implementación, pero seguimos las bases del modelo de la lista.

Programación

Una vez con estas propuestas, nos preparamos para revisar la lista, y teniendo en cuenta todo esto, la programación fue relativamente eficaz, sabiendo que íbamos a modificar, sin embargo, eso no lo dejó exento de errores que tuvimos que ir solucionando, sobre todo, punteros que por fallos lógicos en el orden de ejecución del programa intentábamos acceder a ellos o liberar memoria que ya estaba liberada; en estos casos, la herramienta de debbug de Visual Code Studio es muy útil para rastrear el flujo de ejecución del programa.

Desreferenciamos en comparaciones y en el swapData solo cruzamos los punteros, no el objeto entero; al crear el arreglo de manera dinámica (haciendo un data = new T[ARRAYSIZE]) manejamos la creación y destrucción del mismo cuidando que los espacios de punteros que no almacenaban información tenerlos en Nullptr así mismo cuando se elimina un elemento de la lista.

Fue un proceso más de corrección de errores y seguimiento para encontrar las fugas en la lógica que harían que perdiéramos las referencias, pero con el modelo de la lista ya hecho de antemano, su adaptación eficiente mucho el costo computacional de swaps o corrimientos.

Código Fuente

Carpeta Include

list.hpp

```
#ifndef __LIST_H__
#define __LIST_H__

#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>

#include "ownexceptions.hpp"

template <class T, int ARRAYSIZE = 1024>
class List {
 private:
  T** data;
  int last;

  void copyAll(const List<T, ARRAYSIZE>&);
  void freeAll();

  void swapData(T**, T**);

 public:
  List<T, ARRAYSIZE>();
  List<T, ARRAYSIZE>(const List<T, ARRAYSIZE>&);
  ~List<T, ARRAYSIZE>();

  bool isEmpty() const;
  bool isFull() const;
  bool isSorted() const;

  void insertElement(const T&, const int&);
  void deleteData(const int&);
  T* retrieve(const int&);

  // Getter's
  int getFirstPosition() const;
  int getLastPosition() const;

  int getPrevPosition(const int&) const;
  int getNextPosition(const int&) const;

  std::string toString() const;
  std::string toString(const T&, int(const T&, const T&)) const;
```

```
  void deleteAll();

  bool isValidPosition(const int&) const;

  int findDataL(const T&);
  int findDataB(const T&);

  void insertSortedData(const T&);

  void sortDataBubble();
  void sortDataInsert();
  void sortDataSelect();
  void sortDataShell();

  List<T, ARRAYSIZE> operator=(const List<T, ARRAYSIZE>&);

  template <class X>
  friend std::ostream& operator<<(std::ostream&, const List<X>&);
  template <class X>
  friend std::istream& operator>>(std::istream&, List<X>&);

  // Métodos Extras al Modelo
  bool isSorted(int(const T&, const T&)) const;

  int findDataL(const T&, int(const T&, const T&));
  int findDataB(const T&, int(const T&, const T&));

  void sortDataBubble(int(const T&, const T&));
  void sortDataInsert(int(const T&, const T&));
  void sortDataSelect(int(const T&, const T&));
  void sortDataShell(int(const T&, const T&));

  void insertSortedData(const T&, int(const T&, const T&));
};

template <class T, int ARRAYSIZE>
List<T, ARRAYSIZE>::List() : last(-1) {
  this->data = new T*[ARRAYSIZE];
  for (int i = 0; i < ARRAYSIZE; i++)
    this->data[i] = nullptr;
}

template <class T, int ARRAYSIZE>
List<T, ARRAYSIZE>::~List() {
  freeAll();
  delete[] data;
  data = nullptr;
```

```
}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::copyAll(const List<T, ARRAYSIZE>& other) {
  this->last = other.last;

  this->freeAll();
  for (int i = 0; i <= other.last; i++) {
    this->data[i] = new T(*other.data[i]);
  }

  for (int i = other.last + 1; i < ARRAYSIZE; i++) {
    if (this->data[i])
      delete this->data[i];
    this->data[i] = nullptr;
  }
}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::freeAll() {
  for (int i = 0; i <= this->last; i++) {
    if (this->data[i] != nullptr)
      delete this->data[i];
    this->data[i] = nullptr;
  }
}

template <class T, int ARRAYSIZE>
bool List<T, ARRAYSIZE>::isValidPosition(const int& position) const {
  return !(position > last || position < 0);
}

template <class T, int ARRAYSIZE>
List<T, ARRAYSIZE>::List(const List<T, ARRAYSIZE>& other) {
  this->data = new T*[ARRAYSIZE];

  for (int i = 0; i < ARRAYSIZE; i++)
    this->data[i] = nullptr;

  copyAll(other);
}

template <class T, int ARRAYSIZE>
bool List<T, ARRAYSIZE>::isEmpty() const {
  return this->last == -1;
}

template <class T, int ARRAYSIZE>
```

```
bool List<T, ARRAYSIZE>::isFull() const {
  return this->last == (ARRAYSIZE - 1);
}

template <class T, int ARRAYSIZE>
bool List<T, ARRAYSIZE>::isSorted() const {
  for (int i = 0; i < this->last; i++)
    if (*this->data[i] > *this->data[i + 1])
      return false;

  return true;
}

template <class T, int ARRAYSIZE>
bool List<T, ARRAYSIZE>::isSorted(int cmp(const T&, const T&)) const {
  for (int i = 0; i < this->last; i++)
    if (cmp(*this->data[i], *this->data[i + 1]) > 0)
      return false;

  return true;
}

// Inserción en el Punto de Interés
template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::insertElement(const T& newData, const int&
position) {
  if (isFull())
    throw DataContainersExceptions::MemoryDeficiency(
        "Lista Llena, InsertElement(List)");

  if (!isValidPosition(position) && position != last + 1)
    throw DataContainersExceptions::InvalidPosition(
        "Posicion Invalida, InsertElement(List)");

  for (int i = last; i >= position; i--)
    this->data[i + 1] = this->data[i];

  this->data[position] = new T(newData);
  last++;
}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::deleteData(const int& position) {
  if (!isValidPosition(position))
    throw DataContainersExceptions::InvalidPosition(
        "Poscion Invalida, delteData(List)");

  // Liberar la Memoria de la posicion indicada:
```

```
    delete this->data[position];
  this->data[position] = nullptr;

  // Recorrer la Lista
  for (int i = position; i < last; i++)
    this->data[i] = this->data[i + 1];
  last--;
}

template <class T, int ARRAYSIZE>
T* List<T, ARRAYSIZE>::retrieve(const int& position) {
  if (!isValidPosition(position))
    throw DataContainersExceptions::InvalidPosition(
        "Posicion Invalida, retrieve(List)");
  return data[position];
}

template <class T, int ARRAYSIZE>
int List<T, ARRAYSIZE>::getFirstPosition() const {
  return isEmpty() ? -1 : 0;
}

template <class T, int ARRAYSIZE>
int List<T, ARRAYSIZE>::getLastPosition() const {
  return this->last;
}

template <class T, int ARRAYSIZE>
int List<T, ARRAYSIZE>::getPrevPosition(const int& position) const {
  return (!isValidPosition(position) || position == 0) ? -1 : (position -
1);
}

template <class T, int ARRAYSIZE>
int List<T, ARRAYSIZE>::getNextPosition(const int& position) const {
  return (!isValidPosition(position) || position == last) ? -1 :
(position + 1);
}

template <class T, int ARRAYSIZE>
std::string List<T, ARRAYSIZE>::toString() const {
  std::ostringstream oss;
  for (int i = 0; i <= this->last; i++) {
    oss << "| " << std::to_string(i) << std::setw(11 -
std::to_string(i).size())
        << "" << this->data[i]->toString() << "\n";
  }
```

```
    return oss.str();
}

template <class T, int ARRAYSIZE>
std::string List<T, ARRAYSIZE>::toString(const T& search,
                                         int cmp(const T&, const T&))
const {
  std::ostringstream oss;
  for (int i = 0; i <= this->last; i++) {
    if (cmp(search, *this->data[i]) == 0)
      oss << "| " << std::to_string(i)
          << std::setw(11 - std::to_string(i).size()) << ""
          << this->data[i]->toString() << "\n";
  }

  return oss.str();
}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::deleteAll() {
  this->freeAll();
  this->last = -1;
}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::insertSortedData(const T& newData) {
  int i(0);

  while ((i <= this->last) && (newData > *this->data[i]))
    i++;

  this->insertElement(newData, i);
}

template <class T, int ARRAYSIZE>
int List<T, ARRAYSIZE>::findDataL(const T& searchedData) {
  for (int i = 0; i <= this->last; i++)
    if (*this->data[i] == searchedData)
      return i;
  return -1;
}

template <class T, int ARRAYSIZE>
int List<T, ARRAYSIZE>::findDataB(const T& searchedData) {
  int i(0), j(this->last), middle;

  while (i <= j) {
    middle = (i + j) / 2;
```

```
      if (*this->data[middle] == searchedData)
        return middle;
      if (searchedData < *this->data[middle])
        j = middle - 1;
      else
        i = middle + 1;
    }
    return -1;
}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::swapData(T** a, T** b) {
    T* aux = *a;
    *a = *b;
    *b = aux;
}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::sortDataBubble() {
    int i(this->last), j;
    bool flag;

    do {
      flag = false;
      j = 0;

      while (j < i) {
        if (*this->data[j] > *this->data[j + 1]) {
          swapData(&this->data[j], &this->data[j + 1]);
          flag = true;
        }
        j++;
      }

      i--;
    } while (flag);
}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::sortDataInsert() {
    int i(1), j;
    T* aux;

    while (i <= this->last) {
      aux = this->data[i];

      j = i;
      while (j > 0 && *aux < *this->data[j - 1]) {
```

```
        this->data[j] = this->data[j - 1];

        j--;
      }

      if (i != j) {
        this->data[j] = aux;
      }

      i++;
    }
  }

  template <class T, int ARRAYSIZE>
  void List<T, ARRAYSIZE>::sortDataSelect() {
    int i(0), j, menor;

    while (i <= this->last) {
      menor = i;

      j = i + 1;

      while (j <= this->last) {
        if (*this->data[j] < *this->data[menor]) {
          menor = j;
        }
        j++;
      }

      if (i != menor) {
        this->swapData(&this->data[i], &this->data[menor]);
      }
      i++;
    }
  }

  template <class T, int ARRAYSIZE>
  void List<T, ARRAYSIZE>::sortDataShell() {
    int series[] = {4181, 2584, 1597, 987, 610, 377, 233, 144, 89, 55,
                    34,   21,   13,   8,   5,   3,   2,   1,   0};
    int pos(0), dif(series[pos]), i, j;

    while (dif > 0) {
      i = dif;
      while (i <= this->last) {
        j = i;

        while (j >= dif && *this->data[j - dif] > *this->data[j]) {
```

```
          this->swapData(&this->data[j - dif], &this->data[j]);
          j -= dif;
        }

        i++;
      }

      dif = series[++pos];
    }
}

template <class T, int ARRAYSIZE>
List<T, ARRAYSIZE> List<T, ARRAYSIZE>::operator=(
    const List<T, ARRAYSIZE>& other) {
  copyAll(other);
  return *this;
}

template <class X>
std::ostream& operator<<(std::ostream& os, const List<X>& list) {
  int i = 0;
  while (i <= list.last)
    os << *list.data[i++] << "," << std::endl;

  return os;
}

template <class X>
std::istream& operator>>(std::istream& is, List<X>& list) {
  X obj;

  try {
    while (is >> obj) {
      if (!list.isFull())
        list.data[++list.last] = new X(obj);
    }
  } catch (const std::invalid_argument& ex) {
  }
  return is;
}

// Extras al Modelo de la Lista:
template <class T, int ARRAYSIZE>
int List<T, ARRAYSIZE>::findDataL(const T& searchedData,
                                  int cmp(const T&, const T&)) {
  for (int i = 0; i <= this->last; i++)
    if (cmp(searchedData, *this->data[i]) == 0)
      return i;
```

```
  return -1;
}

template <class T, int ARRAYSIZE>
int List<T, ARRAYSIZE>::findDataB(const T& searchedData,
                                  int cmp(const T&, const T&)) {
  int i(0), j(this->last), middle;

  while (i <= j) {
    middle = (i + j) / 2;

    if (cmp(searchedData, *this->data[middle]) == 0)
      return middle;
    if (cmp(searchedData, *this->data[middle]) < 0)
      j = middle - 1;
    else
      i = middle + 1;
  }

  return -1;
}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::insertSortedData(const T& newData,
                                          int cmp(const T&, const T&)) {
  int i(0);

  while ((i <= this->last) && (cmp(newData, *this->data[i]) > 0))
    i++;

  insertElement(newData, i);
}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::sortDataBubble(int cmp(const T&, const T&)) {
  int i(this->last), j;
  bool flag;

  do {
    flag = false;
    j = 0;

    while (j < i) {
      if (cmp(*this->data[j], *this->data[j + 1]) > 0) {
        swapData(&this->data[j], &this->data[j + 1]);
        flag = true;
      }
```

```
      j++;
    }

    i--;
  } while (flag);
}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::sortDataInsert(int cmp(const T&, const T&)) {
  int i(1), j;
  T* aux;

  while (i <= this->last) {
    aux = this->data[i];

    j = i;
    while (j > 0 && cmp(*aux, *this->data[j - 1]) < 0) {
      this->data[j] = this->data[j - 1];

      j--;
    }

    if (i != j) {
      this->data[j] = aux;
    }

    i++;
  }
}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::sortDataSelect(int cmp(const T&, const T&)) {
  int i(0), j, menor;

  while (i <= this->last) {
    menor = i;

    j = i + 1;

    while (j <= this->last) {
      if (cmp(*this->data[j], *this->data[menor]) < 0) {
        menor = j;
      }
      j++;
    }

    if (i != menor) {
      this->swapData(&this->data[i], &this->data[menor]);
```

```
    }
    i++;
  }
}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::sortDataShell(int cmp(const T&, const T&)) {
  int series[] = {4181, 2584, 1597, 987, 610, 377, 233, 144, 89, 55,
                  34,   21,   13,   8,   5,   3,   2,   1,   0};
  int pos(0), dif(series[pos]), i, j;

  while (dif > 0) {
    i = dif;
    while (i <= this->last) {
      j = i;

      while (j >= dif && cmp(*this->data[j - dif], *this->data[j]) > 0) {
        this->swapData(&this->data[j - dif], &this->data[j]);
        j -= dif;
      }

      i++;
    }

    dif = series[++pos];
  }
}

#endif  // __LIST_H__
```

menu.hpp

```
#ifndef __MENU_H__
#define __MENU_H__

#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <string>

#include "list.hpp"
#include "name.hpp"
#include "song.hpp"

class Menu {
 private:
  List<Song>& songList;

  void enterToContinue();
  int readInteger(std::string, const int&, const int&);
  Name readName(std::string);
  std::string readLinePrompt(const std::string&, bool = false);
  char readChar(const std::string&, const char*);

  bool handleOption(const std::string&);
  std::string windowHeader(const int&, const std::string&) const;
  std::string songTable(const int& = 10,
                        const int& = 35,
                        const int& = 30,
                        const int& = 25) const;

  void noDataMessage();

  void mainMenu();
  void insertSong();
  void deleteSong(const int&);
  void deleteAllSongs();
  void editSong(const int&);
  void exitProgram();

  void searchMenu();
  void searchBySongName();
  void searchByIntepreter();

  void sortMenu();
  void sortBySongName();
  void sortByInterpreter();
  void sortByRanking();
```

```
    void saveToDisk();
    void readFromDisk();

  public:
    Menu();
    Menu(const Menu&);
    Menu(List<Song>&);
};

#endif  // __MENU_H__
```

name.hpp

```
#ifndef __NAME_H__
#define __NAME_H__

#include <fstream>
#include <iostream>
#include <string>

#include "ownexceptions.hpp"

class Name {
 private:
  std::string first;
  std::string last;

 public:
  Name();
  Name(const Name&);
  Name(const std::string&, const std::string&);

  // Interfaz
  // Setter's
  void setFirst(const std::string&);
  void setLast(const std::string&);

  // Getter's
  std::string getFirst() const;
  std::string getLast() const;

  std::string toString() const;

  Name& operator=(const Name&);

  bool operator==(const Name&) const;
  bool operator!=(const Name&) const;
  bool operator<(const Name&) const;
  bool operator>(const Name&) const;
  bool operator<=(const Name&) const;
  bool operator>=(const Name&) const;

  int compareTo(const Name&) const;
  int static compare(const Name&, const Name&);

  friend std::ostream& operator<<(std::ostream&, const Name&);
  friend std::istream& operator>>(std::istream&, Name&);
};
#endif  // __NAME_H__
```

ownexceptions.hpp

```
#ifndef __OWNEXCEPTIONS_H__
#define __OWNEXCEPTIONS_H__

#include <stdexcept>
#include <string>

namespace DataContainersExceptions {
class MemoryDeficiency : public std::runtime_error {
 public:
  explicit MemoryDeficiency(const std::string& msg = "Insuficiencia de
Memoria")
      : std::runtime_error(msg) {}
};

class MemoryOverflow : public std::runtime_error {
 public:
  explicit MemoryOverflow(const std::string& msg = "Desbordamiento de
Memoria")
      : std::runtime_error(msg) {}
};

class InvalidPosition : public std::runtime_error {
 public:
  explicit InvalidPosition(
      const std::string& msg = "La posicion Ingresada es Invalida")
      : std::runtime_error(msg) {}
};
}  // namespace DataContainersExceptions

namespace InputExceptions {
class InvalidOption : public std::runtime_error {
 public:
  explicit InvalidOption(
      const std::string& msg = "La opcion ingresada esta fuera de rango")
      : runtime_error(msg) {}
};

class EmptyString : public std::runtime_error {
 public:
  explicit EmptyString(
      const std::string& msg = "El string no puede estar vacio")
      : runtime_error(msg) {};
};

class OperationCanceledException : public std::runtime_error {
 public:
  explicit OperationCanceledException(
```

```
        const std::string msg = "Operacion Cancelada")
      : runtime_error(msg) {}
};
}  // namespace InputExceptions

#endif  // __OWNEXCEPTIONS_H__
```

song.hpp

```
#ifndef __SONG_H__
#define __SONG_H__

#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <string>

#include "name.hpp"

class Song {
 private:
  int ranking;
  std::string songName;
  Name author;
  Name interpreter;
  std::string mp3Name;

 public:
  Song();
  Song(const Song&);

  /// @brief
  /// @param  Ranking
  /// @param  NombreCancion
  /// @param  NombreAutor
  /// @param  NombreInterprete
  /// @param  NombreMP3
  Song(const int&,
       const std::string&,
       const Name&,
       const Name&,
       const std::string&);

  // Interfaz:
  // Setter's
  void setRanking(const int&);
  void setSongName(const std::string&);
  void setAuthor(const Name&);
  void setInterpreter(const Name&);
  void setMp3Name(const std::string&);

  // Getter's
  int getRanking() const;
  std::string getSongName() const;
  Name getAuthor() const;
```

```
    Name getInterpreter() const;
    std::string getMp3Name() const;

    /// @brief Función toString para la lsit.hpp
    /// @param  widthRanking
    /// @param  widthSongName
    /// @param  widthName
    /// @param  widthMP3
    std::string toString(const int& = 10,
                         const int& = 35,
                         const int& = 30,
                         const int& = 25) const;  // Para impresiones en
list.hpp

    /// @brief Función de 1 sola canción
    /// @param widthBorder
    /// @return
    std::string toStringOnly(
        const int& = 60) const;  // Para impresiones de solo 1 canción

    Song& operator=(const Song&);

    // Operadores Relacionales que utilizan el ranking como compardor
    bool operator==(const Song&) const;
    bool operator!=(const Song&) const;
    bool operator<(const Song&) const;
    bool operator>(const Song&) const;
    bool operator<=(const Song&) const;
    bool operator>=(const Song&) const;

    int compareTo(const Song&) const;
    static int compare(const Song&, const Song&);

    static int compareBySongName(const Song&, const Song&);
    static int compareByAutor(const Song&, const Song&);
    static int compareByInterpreter(const Song&, const Song&);
    static int compareByMP3Name(const Song&, const Song&);

    friend std::ostream& operator<<(std::ostream&, const Song&);
    friend std::istream& operator>>(std::istream&, Song&);
};
#endif  // __SONG_H__
```

Carpeta src

main.cpp

```
#include "menu.hpp"

int main() {
  new Menu(*new List<Song>);

  return 0;
}
```

menu.cpp

```cpp
#include "menu.hpp"

using namespace std;

Menu::Menu() : songList(*new List<Song>) {
  mainMenu();
}

Menu::Menu(const Menu& other) : songList(other.songList) {
  mainMenu();
}

Menu::Menu(List<Song>& s) : songList(s) {
  mainMenu();
}

void Menu::enterToContinue() {
  cout << "[Enter] para continuar..." << endl;
  getchar();
}

int Menu::readInteger(string oss,
                      const int& lowerLimit,
                      const int& upperLimit) {
  string aux("");
  int result;
  while (true) {
    try {
      system("CLS");
      cout << oss;
      getline(cin, aux);
      result = stoi(aux);

      if (result > upperLimit || result < lowerLimit)
        throw InputExceptions::InvalidOption("Numero Fuera de Rango");
      break;
    } catch (const std::invalid_argument& ex) {
      system("CLS");
      cout << "Entrada invalida" << endl;
      cout << "Intente nuevamente" << endl;
      enterToContinue();
    } catch (const InputExceptions::InvalidOption& msg) {
      system("CLS");
      cout << msg.what() << endl;
      enterToContinue();
    }
  }
```

```
  return result;
}

Name Menu::readName(string prompt) {
  Name result;
  result.setFirst(readLinePrompt(prompt));
  prompt += result.getFirst() + "\n";
  result.setLast(readLinePrompt(prompt + "Ingrese el Apellido: "));

  return result;
}

string Menu::readLinePrompt(const string& prompt, bool allowEmpty) {
  string result;
  while (true) {
    system("CLS");
    cout << prompt;
    getline(cin, result);
    if (!allowEmpty && result.empty()) {
      system("CLS");
      cout << "No puede estar vacio.\nIntentelo nuevamente." << endl;
      enterToContinue();
      continue;
    }
    return result;
  }
}

char Menu::readChar(const std::string& prompt, const char* posibilities)
{
  char result, comparation;

  while (true) {
    int i = 0;
    system("CLS");
    cout << prompt;
    cin >> result;

    result = toupper(result);
    do {
      comparation = *(posibilities + i);
      if (result == comparation)
        return result;
      i++;
    } while (comparation != '\0');

    system("CLS");
```

```
      cout << "Opcion Invalida" << endl;
      cout << "Intentelo Nuevamente" << endl;
      system("PAUSE");
    }
}

string Menu::windowHeader(const int& widthBorder, const string& prompt)
const {
  ostringstream oss;

  oss << left << setfill('=') << setw(widthBorder) << "" << endl;
  oss << setfill(' ');

  // Título de Ventana
  oss << setw(widthBorder / 2 - (prompt.size() / 2)) << "| " << prompt
      << setw((widthBorder / 2) - (prompt.size() / 2) - 2) << "" << "|"
<< endl;
  oss << setfill('-') << setw(widthBorder) << "" << endl;
  oss << setfill(' ');

  return oss.str();
}

bool Menu::handleOption(const std::string& promt) {
  string response;

  system("CLS");
  cout << promt;
  getline(cin, response);

  if (response.empty())
    return true;

  // Hacer las letras mayúsculas
  char option =
      static_cast<char>(std::toupper(static_cast<unsigned
char>(response[0])));

  // buscar primer dígito después de la letra (saltando espacios)
  std::size_t pos = 1;
  while (pos < response.size() &&
         std::isspace(static_cast<unsigned char>(response[pos])))
    ++pos;

  bool hasNumber = false;
  int index = -1;
  if (pos < response.size() &&
      std::isdigit(static_cast<unsigned char>(response[pos]))) {
```

```
    std::size_t start = pos;
    std::size_t end = start;
    while (end < response.size() &&
           std::isdigit(static_cast<unsigned char>(response[end])))
      ++end;
    std::string numstr = response.substr(start, end - start);
    try {
      index = std::stoi(numstr);
      hasNumber = true;
    } catch (...) {
      hasNumber = false;
    }
  }

  switch (option) {
    case 'A':
      this->insertSong();
      break;

    case 'B':
      if (!hasNumber) {
        system("CLS");
        std::cout << "Falta numero de posicion. Ej: B2\n";
        this->enterToContinue();
        break;
      }
      if (!this->songList.isValidPosition(index)) {
        system("CLS");
        std::cout << "Posicion de lista invalida\n";
        this->enterToContinue();
        break;
      }
      this->editSong(index);
      break;

    case 'C':
      if (!hasNumber) {
        system("CLS");
        std::cout << "Falta numero de posicion. Ej: C12\n";
        this->enterToContinue();
        break;
      }
      if (!this->songList.isValidPosition(index)) {
        system("CLS");
        std::cout << "Posicion de lista invalida\n";
        this->enterToContinue();
        break;
      }
```

```
        this->deleteSong(index);
      break;

    case 'D':
      if (!hasNumber) {
        system("CLS");
        std::cout << "Falta numero de posiciin. Ej: D12\n";
        this->enterToContinue();
        break;
      }
      if (!this->songList.isValidPosition(index)) {
        system("CLS");
        std::cout << "Posicion de lista invalida\n";
        this->enterToContinue();
        break;
      }
      system("CLS");
      {
        Song* s = this->songList.retrieve(index);
        if (s)
          std::cout << s->toStringOnly();
        else
          std::cout << "Cancion no encontrada\n";
      }
      this->enterToContinue();
      break;
    case 'E':
      this->deleteAllSongs();
      break;
    case 'F':
      this->saveToDisk();
      break;
    case 'G':
      this->readFromDisk();
      break;
    case 'H':
      this->searchMenu();
      break;
    case 'I':
      this->sortMenu();
      break;
    case 'J':
      this->exitProgram();
      return false;

    default:
      system("CLS");
      std::cout << "Comando invalido\nIntentelo nuevamente.\n";
```

```
        enterToContinue();
        break;
    }  // switch

    return true;
}

std::string Menu::songTable(const int& widthRanking,
                            const int& widthSongName,
                            const int& widthName,
                            const int& widthMP3) const {
    ostringstream oss;

    int widthBorder =
        widthRanking + widthSongName + (widthName * 2) + widthMP3 + 16;

    oss << windowHeader(widthBorder, "LISTA DE EXITOS");
    oss << "| " << "# En Lista" << setw(3) << "| " << "Ranking"
        << setw(widthRanking - 7) << "| " << "Nombre de la Cancion"
        << setw(widthSongName - 20) << "| " << "Nombre del Artista"
        << setw(widthName - 18) << "| " << "Nombre del Interprete"
        << setw(widthName - 21) << "| " << "Nombre del MP3" <<
setw(widthMP3 - 14)
        << "|" << endl;

    oss << setfill('-');
    oss << setw(widthBorder) << " ";
    oss << setfill(' ') << endl;

    oss << this->songList.toString();

    oss << setfill('-');
    oss << setw(widthBorder) << " ";
    oss << setfill(' ');
    oss << endl;

    return oss.str();
}

void Menu::noDataMessage() {
    cout << "+------------------------------------------------------+" <<
endl;
    cout << "+            No hay Canciones Registradas Aun          +" <<
endl;
    cout << "+              Regresando al Menu...                   +" <<
endl;
    cout << "+------------------------------------------------------+" <<
endl;
```

```
    this->enterToContinue();
}

void Menu::mainMenu() {
  ostringstream oss;
  bool running = true;

  while (running) {
    system("CLS");

    // Limpiar el ostringstream
    oss.str("");
    oss.clear();

    oss << this->songTable();
    oss << "Opciones: \n";
    oss << "[A] Agregar una Cancion.  [B<n>] Editar una Cancion [C<n>] "
           "Eliminar "
           "una Cancion. [D<n>]  Mostrar Detalles de
Cancion.  [E]    Eliminar "
           "Todas las Canciones. \n\n"
           "[F] Guardar la Database   [G] Leer del Disco   "
           "[H] Buscar una Cancion    [I] Ordenar Lista  [J] Salir.\n\n";
    oss << "Seleccione un Comando: ";

    running = handleOption(oss.str());
  }
}

void Menu::insertSong() {
  int widthBorder = 100;
  Song newSong;
  string myString("");
  int myInt(0);
  Name myName;
  ostringstream oss;

  do {
    system("CLS");
    // Linea Exterior
    oss << windowHeader(widthBorder, "INSERTAR EXITO");

    oss << "Ingrese el Nombre de la Cancion: ";
    myString = this->readLinePrompt(oss.str(), false);
    newSong.setSongName(myString);
    oss << newSong.getSongName() << endl;

    oss << "Ingrese el Ranking de la Cancion: ";
```

```
while (true) {
  try {
    system("CLS");
    myInt = readInteger(oss.str(), 0, 3000);
    newSong.setRanking(myInt);
    if (songList.findDataL(newSong) != -1)
      throw std::invalid_argument("Ranking ya utilizado");
    break;
  } catch (const InputExceptions::InvalidOption& msg) {
    system("CLS");
    cout << msg.what() << endl;
    enterToContinue();
  } catch (const std::invalid_argument& msg) {
    system("CLS");
    cout << msg.what() << endl;
    enterToContinue();
  }
}

oss << newSong.getRanking() << endl;
oss << "Ingrese el Nombre del Autor: ";
myName = readName(oss.str());
oss << myName.getFirst() << endl;
oss << "Ingrese el Apellido: " << myName.getLast() << endl;

newSong.setAuthor(myName);

oss << "Ingrese el Nombre del Interprete: ";
myName = readName(oss.str());
oss << myName.getFirst() << endl;
oss << "Ingrese el Apellido: " << myName.getLast() << endl;

newSong.setInterpreter(myName);

oss << "Ingrese el nombre del Archivo MP3: ";
myString = this->readLinePrompt(oss.str(), false);
newSong.setMp3Name(myString);
oss << newSong.getMp3Name() << endl;

if (songList.isEmpty())
  songList.insertElement(newSong, 0);
else {
  oss << "Ingrese la posicion en la lista que tendra la cancion: ";
  while (true) {
    try {
      myInt = readInteger(oss.str(), 0, 49);
      songList.insertElement(newSong, myInt);
```

```
            oss << myInt << endl;
            break;
          } catch (const DataContainersExceptions::InvalidPosition& msg) {
            system("CLS");
            cout << msg.what() << endl;
            cout << "Intente Nuevamente." << endl;
            enterToContinue();
          } catch (const DataContainersExceptions::MemoryOverflow& msg) {
            system("CLS");
            cout << msg.what() << endl;
            cout << "Regresando..." << endl;
            enterToContinue();
            return;
          }
        }
      }

      oss << "Cancion Agregada con Exito!." << endl;
      oss << "Desea Agregar Otra Cancion? (1. Si / 2. No): ";

      myInt = readInteger(oss.str(), 1, 2);

      oss.str("");
      oss.clear();
    } while (myInt != 2);
}

void Menu::deleteSong(const int& position) {
    system("CLS");
    ostringstream oss;
    int response;

    Song* target = songList.retrieve(position);
    oss << target->toStringOnly();
    oss << "Esta seguro que desea eliminar esta cancion? (1. Si/ 2. No): ";
    response = readInteger(oss.str(), 1, 2);
    if (response == 1) {
      songList.deleteData(position);
      oss << endl << "Cancion Eliminada con Exito!" << endl;
    } else
      oss << endl << "Operacion Cancelada" << endl;

    system("CLS");
    cout << oss.str();

    enterToContinue();
}
```

```
void Menu::deleteAllSongs() {
  system("CLS");
  if (songList.getLastPosition() == -1) {
    cout << "Aun no hay canciones para eliminar" << endl;
    enterToContinue();
    return;
  }

  ostringstream oss;
  int widhtBorder = 50;

  oss << windowHeader(widhtBorder, "ELIMINAR TODAS LAS CANCIONES");

  oss << "Esta seguro que desea eliminar las " <<
songList.getLastPosition() + 1
      << " canciones? (1. Si/ 2. No): ";
  int response = readInteger(oss.str(), 1, 2);
  system("CLS");
  if (response == 1) {
    songList.deleteAll();
    cout << "Canciones eliminadas con Exito!" << endl;
    cout << "Base de Datos Vacia." << endl;
  } else {
    cout << "Operacion Cancelada." << endl;
  }
  enterToContinue();
}

void Menu::editSong(const int& position) {
  ostringstream oss;
  Song* target = songList.retrieve(position);
  int editOption, newRanking;
  string dataString;
  Name newName;
  Song ver;

  oss << target->toStringOnly();
  oss << "6 Salir\n";

  editOption = readInteger(
      oss.str() + "Elige el atributo que quieras cambiar (1-6): ", 1, 6);

  switch (editOption) {
    case 1:
      oss << "Ingrese el Nuevo Ranking de la Cancion: ";
      newRanking = readInteger(oss.str(), 1, 50);
      ver.setRanking(newRanking);
      if (this->songList.findDataL(ver) != -1) {
```

```
          system("CLS");
          cout << "El ranking ya esta ocupado" << endl;
          enterToContinue();
          break;
        }
        target->setRanking(newRanking);
        cout << "Cambio hecho con Exito!";
        break;
      case 2:
        oss << "Ingrese el nuevo nombre de la cancion: ";
        dataString = readLinePrompt(oss.str());
        target->setSongName(dataString);
        cout << "Cambio hecho con Exito!";
        enterToContinue();
        break;
      case 3:
        oss << "Ingrese el nuevo autor de la cancion: ";
        newName = readName(oss.str());
        target->setAuthor(newName);
        cout << "Cambio hecho con Exito!";
        break;
      case 4:
        oss << "Ingrese el nuevo interprete de la cancion: ";
        newName = readName(oss.str());
        target->setInterpreter(newName);
        cout << "Cambio hecho con Exito!";
        break;
      case 5:
        oss << "Ingrese el nuevo nombre del Archivo MP3: ";
        dataString = this->readLinePrompt(oss.str());
        target->setMp3Name(dataString);
        cout << "Cambio Hecho con Exito!";
        break;
      case 6:
        return;
      default:
        break;
    }
    enterToContinue();
}

void Menu::exitProgram() {
    system("CLS");
    int response;
    ostringstream oss;
    if (!this->songList.isEmpty()) {
      oss << windowHeader(50, "SALIR SIN GUARDAR?");
      response = readInteger(
```

```
            oss.str() +
               "Desea Guardar las canciones antes de Salir? (1. Si/ 2. No):
",
            1, 2);
      if (response == 1)
        saveToDisk();
    }

    system("CLS");
    std::cout << "Saliendo del Programa.\nTenga un Lindo Dia :D\n";
    enterToContinue();
}

void Menu::searchMenu() {
    system("CLS");
    if (this->songList.isEmpty()) {
      this->noDataMessage();
      return;
    }

    ostringstream oss;
    char op;
    oss << windowHeader(50, "BUSCAR CANCION");
    oss << "Existen un total de: " << this->songList.getLastPosition() + 1
        << " registradas." << endl;
    oss << "A continuacion se muestran las opciones de busqueda: " << endl;
    oss << "[A] Buscar por Nombre de Cancion" << endl
        << "[B] Buscar por Nombre del Inteprete" << endl
        << "[R] Regresar." << endl
        << "Seleccione una Opcion: ";

    while (op != 'R') {
      system("CLS");
      cout << oss.str();
      cin >> op;
      cin.ignore();

      op = toupper(op);

      switch (op) {
        case 'A':
          this->searchBySongName();
          break;
        case 'B':
          this->searchByIntepreter();
          break;
        case 'R':
          system("CLS");
```

```
        cout << "Regresando...";
        enterToContinue();
        break;
      default:
        system("CLS");
        cout << "Opcion invalida" << endl;
        cout << "Intentelo nuevamente" << endl;
        enterToContinue();
        break;
    }
  }
}

void Menu::searchBySongName() {
  List<Song> songWithTheName;
  List<Song> auxList = this->songList;
  ostringstream oss;
  char response, options[2] = {'B', 'L'};
  string songName;
  Song searchedSong;
  int position, repeat;

  do {
    oss.str("");
    oss.clear();
    system("CLS");
    oss << windowHeader(50, "BUSCAR POR NOMBRE DE CANCION");
    oss << "Ingrese el nombre de la cancion que quiera buscar: ";

    songName = readLinePrompt(oss.str(), false);
    searchedSong.setSongName(songName);
    oss << songName << endl;

    oss << "Desea Realizar Una Busqueda Binaria o Lineal? (L/B): ";

    response = readChar(oss.str(), options);
    cin.ignore();
    oss << response << endl;

    if (response == 'L') {
      try {
        while (true) {
          position = auxList.findDataL(searchedSong,
Song::compareBySongName);
          songWithTheName.insertSortedData(*auxList.retrieve(position));
          auxList.deleteData(position);
        }
      } catch (const DataContainersExceptions::InvalidPosition& ex) {
```

```
          // No hacer nada
        }
      } else {
        try {
          while (true) {
            position = auxList.findDataB(searchedSong,
Song::compareBySongName);
            songWithTheName.insertSortedData(*auxList.retrieve(position));
            auxList.deleteData(position);
          }
        } catch (const DataContainersExceptions::InvalidPosition& ex) {
          // No hacer nada
        }
      }

      if (songWithTheName.isEmpty())
        oss << "\nNo existe un registro de una cancion llamada: " <<
songName
            << endl;
      else {
        oss.str("");
        oss.clear();
        oss << this->windowHeader(146, "LISTA DE EXITOS");
        oss << "| " << "# En Lista" << setw(3) << "| " << "Ranking" <<
setw(3)
            << "| " << "Nombre de la Cancion" << setw(15) << "| "
            << "Nombre del Artista" << setw(12) << "| " << "Nombre del
Interprete"
            << setw(9) << "| " << "Nombre del MP3" << setw(11) << "|" <<
endl;

        oss << setfill('-');
        oss << setw(146) << " ";
        oss << setfill(' ') << endl;
        oss << songWithTheName.toString();

        oss << setfill('-');
        oss << setw(146) << "";
        oss << setfill(' ') << endl;
      }

      songWithTheName.deleteAll();
      oss << "Desea Realizar Otra Busqueda?: (1.Si / 2.No): ";

      repeat = readInteger(oss.str(), 1, 2);

    } while (repeat != 2);
    system("CLS");
```

```
    cout << "Regresando..." << endl;
  this->enterToContinue();
}

void Menu::searchByIntepreter() {
  List<Song> songsOfInterpreter;
  List<Song> auxList = this->songList;
  ostringstream oss;
  char response, options[2] = {'B', 'L'};
  string dataString;
  Name searchedName;
  Song searchedSong;
  int position, repeat;

  do {
    oss.str("");
    oss.clear();
    system("CLS");
    oss << windowHeader(70, "BUSCAR POR INTERPRETE DE LA CANCION");

    oss << "Ingrese el Nombre del Interprete: ";
    searchedName = readName(oss.str());
    oss << searchedName.getFirst() << endl;
    oss << "Ingrese el Apellido: " << searchedName.getLast() << endl;

    searchedSong.setInterpreter(searchedName);

    oss << "Desea Realizar Una Busqueda Binaria o Lineal? (L/B): ";

    response = readChar(oss.str(), options);
    cin.ignore();
    oss << response << endl;

    if (response == 'L') {
      try {
        while (true) {
          position =
              auxList.findDataL(searchedSong,
Song::compareByInterpreter);
          songsOfInterpreter.insertSortedData(*auxList.retrieve(position)
);
          auxList.deleteData(position);
        }
      } catch (const DataContainersExceptions::InvalidPosition& ex) {
        // No hacer nada
      }
    } else {
      try {
```

```
        while (true) {
          position =
              auxList.findDataB(searchedSong,
Song::compareByInterpreter);
          songsOfInterpreter.insertSortedData(*auxList.retrieve(position)
);
          auxList.deleteData(position);
        }
      } catch (const DataContainersExceptions::InvalidPosition& ex) {
        // No hacer nada
      }
    }

    if (songsOfInterpreter.isEmpty())
      oss << "\nNo existe un registro de una cancion del interprete: "
          << searchedName.toString() << endl;
    else {
      oss.str("");
      oss.clear();
      oss << windowHeader(146, "LISTA DE EXITOS");
      oss << "| " << "# En Lista" << setw(3) << "| " << "Ranking" <<
setw(3)
          << "| " << "Nombre de la Cancion" << setw(15) << "| "
          << "Nombre del Artista" << setw(12) << "| " << "Nombre del
Interprete"
          << setw(9) << "| " << "Nombre del MP3" << setw(11) << "|" <<
endl;

      oss << setfill('-');
      oss << setw(146) << " ";
      oss << setfill(' ') << endl;
      oss << songsOfInterpreter.toString();

      oss << setfill('-');
      oss << setw(146) << "";
      oss << setfill(' ') << endl;
    }
    songsOfInterpreter.deleteAll();
    oss << "Desea Realizar Otra Busqueda?: (1.Si / 2.No): ";

    repeat = readInteger(oss.str(), 1, 2);

  } while (repeat != 2);
  system("CLS");
  cout << "Regresando..." << endl;
  enterToContinue();
}
```

```
void Menu::sortMenu() {
  system("CLS");
  if (this->songList.isEmpty()) {
    this->noDataMessage();
    return;
  }

  if (this->songList.getLastPosition() == 0) {
    cout << "Solo hay 1 cancion registrada." << endl;
    cout << "No se requiere Ordenamiento." << endl;
    enterToContinue();
    return;
  }

  ostringstream oss;
  char op;
  oss << windowHeader(50, "ORDENAR EXITOS");
  oss << "Existen un total de: " << this->songList.getLastPosition() + 1
      << " registradas." << endl;
  oss << "A continuacion se muestran las opciones sobre las cuales
ordenar las "
         "canciones: "
      << endl;
  oss << "[A] Ordenar por Nombre de Cancion" << endl
      << "[B] Ordenar por Nombre del Inteprete" << endl
      << "[C] Ordenar por Numero de Ranking" << endl
      << "[R] Regresar." << endl
      << "Seleccione una Opcion: ";

  while (op != 'R') {
    char charsValid[] = {'A', 'B', 'C', 'R'};
    op = readChar(oss.str(), charsValid);
    cin.ignore();

    switch (op) {
      case 'A':
        this->sortBySongName();
        break;
      case 'B':
        this->sortByInterpreter();
        break;
      case 'C':
        this->sortByRanking();
        break;
      case 'R':
        system("CLS");
        cout << "Regresando...";
        enterToContinue();
```

```
        break;
    }
  }
}

void Menu::sortBySongName() {
  system("CLS");
  ostringstream oss;
  char op, validOptions[5] = {'A', 'B', 'C', 'D', 'E'};

  oss << windowHeader(100, "Ordenar por Nombre de la Cancion");
  oss << "A continuacion, eliga un algoritmo de ordenamiento para la
lista: "
      << endl;
  oss << "[A] Ordenamiento por Burbuja" << endl;
  oss << "[B] Ordenamiento por InsertSort" << endl;
  oss << "[C] Ordenamiento por SelectSort" << endl;
  oss << "[D] Ordenamiento por ShellSort" << endl;
  oss << "[E] Regresar" << endl << endl;
  oss << "Ingrese una Opcion: ";

  op = this->readChar(oss.str(), validOptions);
  cin.ignore();

  oss << op << endl;
  switch (op) {
    case 'A':
      this->songList.sortDataBubble(Song::compareBySongName);
      oss << "Ordenamiento por Burbuja hecho correctamente!." << endl;
      break;
    case 'B':
      this->songList.sortDataInsert(Song::compareBySongName);
      oss << "Ordenamiento por InserSor hecho correctamente!" << endl;
      break;
    case 'C':
      this->songList.sortDataSelect(Song::compareBySongName);
      oss << "Ordenamiento por DataSelect hecho correctamente!" << endl;
      break;
    case 'D':
      this->songList.sortDataShell(Song::compareBySongName);
      oss << "Ordenamiento por ShellSort hecho correctamente!" << endl;
      break;
    case 'E':
      system("CLS");
      cout << "Regresando..." << endl;
      enterToContinue();
      return;
      break;
```

```
    }

  system("CLS");
  oss << endl << "Regresando..." << endl << endl;
  cout << oss.str();
  enterToContinue();
}

void Menu::sortByInterpreter() {
  system("CLS");
  ostringstream oss;
  char op, validOptions[5] = {'A', 'B', 'C', 'D', 'E'};

  oss << windowHeader(100, "Ordenar por Interprete de la Cancion");
  oss << "A continuacion, eliga un algoritmo de ordenamiento para la
lista: "
      << endl;
  oss << "[A] Ordenamiento por Burbuja" << endl;
  oss << "[B] Ordenamiento por InsertSort" << endl;
  oss << "[C] Ordenamiento por SelectSort" << endl;
  oss << "[D] Ordenamiento por ShellSort" << endl;
  oss << "[E] Regresar" << endl << endl;
  oss << "Ingrese una Opcion: ";

  op = this->readChar(oss.str(), validOptions);
  cin.ignore();

  oss << op << endl;
  switch (op) {
    case 'A':
      this->songList.sortDataBubble(Song::compareByInterpreter);
      oss << "Ordenamiento por Burbuja hecho correctamente!." << endl;
      break;
    case 'B':
      this->songList.sortDataInsert(Song::compareByInterpreter);
      oss << "Ordenamiento por InserSor hecho correctamente!" << endl;
      break;
    case 'C':
      this->songList.sortDataSelect(Song::compareByInterpreter);
      oss << "Ordenamiento por DataSelect hecho correctamente!" << endl;
      break;
    case 'D':
      this->songList.sortDataShell(Song::compareByInterpreter);
      oss << "Ordenamiento por ShellSort hecho correctamente!" << endl;
      break;
    case 'E':
      system("CLS");
      cout << "Regresando..." << endl;
```

```
        enterToContinue();
        return;
        break;
    }

    system("CLS");
    oss << endl << "Regresando..." << endl << endl;
    cout << oss.str();
    enterToContinue();
}

void Menu::sortByRanking() {
    system("CLS");
    ostringstream oss;
    char op, validOptions[5] = {'A', 'B', 'C', 'D', 'E'};

    oss << windowHeader(100, "Ordenar por Ranking de la Cancion");
    oss << "A continuacion, eliga un algoritmo de ordenamiento para la
lista: "
        << endl;
    oss << "[A] Ordenamiento por Burbuja" << endl;
    oss << "[B] Ordenamiento por InsertSort" << endl;
    oss << "[C] Ordenamiento por SelectSort" << endl;
    oss << "[D] Ordenamiento por ShellSort" << endl;
    oss << "[E] Regresar" << endl << endl;
    oss << "Ingrese una Opcion: ";

    op = this->readChar(oss.str(), validOptions);
    cin.ignore();

    oss << op << endl;
    switch (op) {
      case 'A':
        this->songList.sortDataBubble();
        oss << "Ordenamiento por Burbuja hecho correctamente!." << endl;
        break;
      case 'B':
        this->songList.sortDataInsert();
        oss << "Ordenamiento por InserSor hecho correctamente!" << endl;
        break;
      case 'C':
        this->songList.sortDataSelect();
        oss << "Ordenamiento por DataSelect hecho correctamente!" << endl;
        break;
      case 'D':
        this->songList.sortDataShell();
        oss << "Ordenamiento por ShellSort hecho correctamente!" << endl;
        break;
```

```
    case 'E':
      system("CLS");
      cout << "Regresando..." << endl;
      enterToContinue();
      return;
      break;
  }

  system("CLS");
  oss << endl << "Regresando..." << endl << endl;
  cout << oss.str();
  enterToContinue();
}

void Menu::saveToDisk() {
  system("CLS");
  ostringstream oss;

  if (this->songList.isEmpty()) {
    cout << "+-----------------------------------------------------+"
<< endl;
    cout << "+            No hay Canciones Registradas Aun         +"
<< endl;
    cout << "+              Regresando al Menu...                  +"
<< endl;
    cout << "+-----------------------------------------------------+"
<< endl;
    enterToContinue();
    return;
  }

  int widthBorder = 50;
  string fileName("");
  ofstream file;

  oss << windowHeader(widthBorder, "GUARDAR DATABASE");

  oss << "Ingrese el Nombre que Tendra el Archivo: ";
  fileName = readLinePrompt(oss.str());

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

  if (!file.is_open())
    oss << "No se permite la creacion de archivos." << endl;
  else {
    file << this->songList;
    oss << "Database guardada con Exito!" << endl;
  }
```

```
  system("CLS");
  cout << oss.str();
  enterToContinue();
}

void Menu::readFromDisk() {
  system("CLS");
  ostringstream oss;
  int widthBorder = 100;
  ifstream file;
  string fileName;

  oss << windowHeader(widthBorder, "LEER ARCHIVO");

  oss << "Tenga en Cuenta que los Archivos se Sobreescribiran" << endl;
  oss << "Ingrese el Nombre del Archivo a Cargar sus Datos: ";

  fileName = readLinePrompt(oss.str());
  oss << fileName << endl;

  file.open(fileName);

  if (!file.is_open())
    oss << "El archivo no existe o no pudo ser abierto" << endl;
  else {
    this->songList.deleteAll();
    file >> this->songList;
    oss << "Archivos Cargados Con Exito!" << endl;
  }

  oss << setfill('=') << setw(widthBorder) << "" << endl;
  oss << setfill(' ');

  system("CLS");
  cout << oss.str();
  enterToContinue();
}
```

name.cpp

```
#include "name.hpp"

Name::Name() : first("default"), last("default") {}

Name::Name(const Name& other) : first(other.first), last(other.last) {}

Name::Name(const std::string& f, const std::string& l) : first(f),
last(l) {}

void Name::setFirst(const std::string& first) {
  if (first.empty())
    throw InputExceptions::EmptyString(
        "Nombre no puede estar vacío, setFirst(Name)");
  this->first = first;
}

void Name::setLast(const std::string& last) {
  if (last.empty())
    throw InputExceptions::EmptyString(
        "Apellido no puede estar vacío, setLast(Name)");
  this->last = last;
}

std::string Name::getFirst() const {
  return this->first;
}

std::string Name::getLast() const {
  return this->last;
}

std::string Name::toString() const {
  return this->first + " " + this->last;
}

Name& Name::operator=(const Name& other) {
  this->first = other.first;
  this->last = other.last;

  return *this;
}

bool Name::operator==(const Name& other) const {
  return this->toString() == other.toString();
}

bool Name::operator!=(const Name& other) const {
```

```
    return !(*this == other);
}

bool Name::operator<(const Name& other) const {
  return this->toString() < other.toString();
}

bool Name::operator>(const Name& other) const {
  return this->toString() > other.toString();
}

bool Name::operator<=(const Name& other) const {
  return (*this < other) || (*this == other);
}

bool Name::operator>=(const Name& other) const {
  return (*this > other) || (*this == other);
}

int Name::compareTo(const Name& other) const {
  return this->toString().compare(other.toString());
}

int Name::compare(const Name& nameA, const Name& nameB) {
  return nameA.toString().compare(nameB.toString());
}

std::ostream& operator<<(std::ostream& os, const Name& name) {
  os << name.first << "," << name.last;

  return os;
}

std::istream& operator>>(std::istream& is, Name& name) {
  std::string dataString;
  getline(is, dataString, ',');
  name.first = dataString;
  getline(is, dataString, ',');
  name.last = dataString;

  return is;
}
```

song.cpp

```cpp
#include "song.hpp"

using namespace std;

Song::Song()
    : ranking(-1),
      songName("default"),
      author(),
      interpreter(),
      mp3Name("default") {}

Song::Song(const Song& other)
    : ranking(other.ranking),
      songName(other.songName),
      author(other.author),
      interpreter(other.interpreter),
      mp3Name(other.mp3Name) {}

Song::Song(const int& r,
           const std::string& n,
           const Name& a,
           const Name& i,
           const std::string& m)
    : ranking(r), songName(n), author(a), interpreter(i), mp3Name(m) {}

void Song::setRanking(const int& ranking) {
  if (ranking <= 0)
    throw InputExceptions::InvalidOption("El ranking debe ser positivo");
  this->ranking = ranking;
}

void Song::setSongName(const std::string& songName) {
  if (songName.empty())
    throw InputExceptions::EmptyString("El nombre no puede estar
vacio.");
  this->songName = songName;
}

void Song::setAuthor(const Name& author) {
  this->author = author;  // Name tiene sus propias validaciones
}

void Song::setInterpreter(const Name& interpreter) {
  this->interpreter = interpreter;
}

void Song::setMp3Name(const std::string& mp3Name) {
```

```
    if (mp3Name.empty())
      throw InputExceptions::EmptyString("El nombre no puede estar vacio");
    this->mp3Name = mp3Name;
}

int Song::getRanking() const {
    return this->ranking;
}

std::string Song::getSongName() const {
    return this->songName;
}

Name Song::getAuthor() const {
    return this->author;
}

Name Song::getInterpreter() const {
    return this->interpreter;
}

std::string Song::getMp3Name() const {
    return this->mp3Name;
}

std::string Song::toStringOnly(const int& widthBorder) const {
    ostringstream oss;

    oss << setfill('=') << setw(widthBorder) << "" << endl;
    oss << setfill(' ');

    oss << "| " << setw((widthBorder / 2) + 10) << "INFORMACION DE LA
CANCION"
        << setw((widthBorder / 2) - 12) << "|" << endl;

    oss << setfill('-') << setw(widthBorder) << "" << endl;
    oss << setfill(' ');

    oss << "Posicion en el Ranking: " << ranking << endl;
    oss << "Nombre de la Cancion: " << songName << endl;
    oss << "Nombre del Autor: " << author.toString() << endl;
    oss << "Nombre del Inteprete: " << interpreter.toString() << endl;
    oss << "Nombre del Archivo MP3" << mp3Name << endl;

    oss << endl << setfill('=') << setw(widthBorder) << "" << endl;
    oss << setfill(' ');

    return oss.str();
```

```
}

std::string Song::toString(const int& widthRanking,
                           const int& widthSongName,
                           const int& widthName,
                           const int& widthMP3) const {
  ostringstream oss;

  oss << "| " << this->ranking
      << setw(widthRanking - to_string(this->ranking).size()) << "| "
      << this->songName << setw(widthSongName - this->songName.size()) <<
"| "
      << this->author.toString()
      << setw(widthName - this->author.toString().size()) << "| "
      << this->interpreter.toString()
      << setw(widthName - this->interpreter.toString().size()) << "| "
      << this->mp3Name << setw(widthMP3 - this->mp3Name.size()) << "|";

  return oss.str();
}

Song& Song::operator=(const Song& other) {
  this->ranking = other.ranking;
  this->songName = other.songName;
  this->author = other.author;
  this->interpreter = other.interpreter;
  this->mp3Name = other.mp3Name;

  return *this;
}

bool Song::operator==(const Song& other) const {
  return this->ranking == other.ranking;
}

bool Song::operator!=(const Song& other) const {
  return !(*this == other);
}

bool Song::operator<(const Song& other) const {
  return this->ranking < other.ranking;
}

bool Song::operator>(const Song& other) const {
  return this->ranking > other.ranking;
}

bool Song::operator<=(const Song& other) const {
```

```
    return !(*this > other);
}

bool Song::operator>=(const Song& other) const {
  return !(*this < other);
}

int Song::compareTo(const Song& other) const {
  return this->ranking - other.ranking;
}

int Song::compare(const Song& songA, const Song& songB) {
  return songA.ranking - songB.ranking;
}

int Song::compareBySongName(const Song& songA, const Song& songB) {
  return songA.songName.compare(songB.songName);
}

int Song::compareByAutor(const Song& songA, const Song& songB) {
  return songA.author.compareTo(songB.author);
}

int Song::compareByInterpreter(const Song& songA, const Song& songB) {
  return songA.interpreter.compareTo(songB.interpreter);
}

int Song::compareByMP3Name(const Song& songA, const Song& songB) {
  return songA.mp3Name.compare(songB.mp3Name);
}

std::ostream& operator<<(std::ostream& os, const Song& song) {
  os << song.ranking << "," << song.songName << "," << song.author << ","
     << song.interpreter << "," << song.mp3Name;

  return os;
}

std::istream& operator>>(std::istream& is, Song& song) {
  string dataString;
  getline(is, dataString, ',');
  song.ranking = stoi(dataString);
  getline(is, song.songName, ',');
  is >> song.author;
  is >> song.interpreter;
  getline(is, song.mp3Name);

  return is;}
```

Ejecución del Programa

Como en el programa no añadimos nuevas funcionalidades, este apartado será mostrando un poco del funcionamiento normal del programa para verificar que los cambios realizados a la lista permiten el correcto funcionamiento del programa:

```
================================================================================================================================
|                                                        LISTA DE EXITOS                                                       |
--------------------------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista          | Nombre del Interprete       | Nombre del MP3         |
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
Opciones:
[A] Agregar una Cancion.  [B<n>] Editar una Cancion [C<n>] Eliminar una Cancion. [D<n>]  Mostrar Detalles de Cancion.  [E]    Eliminar Todas las Canciones.

[F] Guardar la Database   [G] Leer del Disco   [H] Buscar una Cancion    [I] Ordenar Lista  [J] Salir.

Seleccione un Comando:
```

En el menú principal, vamos a agregar una canción:

```
================================================================================================================================
|                                                        INSERTAR EXITO                                                        |
--------------------------------------------------------------------------------------------------------------------------------
Ingrese el Nombre de la Cancion: Erase una Bestia
Ingrese el Ranking de la Cancion: 5
Ingrese el Nombre del Autor: Jose
Ingrese el Apellido: Madero
Ingrese el Nombre del Interprete: Jose
Ingrese el Apellido: Madero
Ingrese el nombre del Archivo MP3: EbestJM.mp3
Cancion Agregada con Exito!.
Desea Agregar Otra Cancion? (1. Si / 2. No):
```

Al regresar al menú principal;

```
================================================================================================================================
|                                                        LISTA DE EXITOS                                                       |
--------------------------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista          | Nombre del Interprete       | Nombre del MP3         |
--------------------------------------------------------------------------------------------------------------------------------
| 0          | 5       | Erase una Bestia                | Jose Madero                 | Jose Madero                 | EbestJM.mp3            |
--------------------------------------------------------------------------------------------------------------------------------
Opciones:
[A] Agregar una Cancion.  [B<n>] Editar una Cancion [C<n>] Eliminar una Cancion. [D<n>]  Mostrar Detalles de Cancion.  [E]    Eliminar Todas las Canciones.

[F] Guardar la Database   [G] Leer del Disco   [H] Buscar una Cancion    [I] Ordenar Lista  [J] Salir.

Seleccione un Comando:
```

Podemos editarla con el comando B<N> (B0 en este caso):

```
================================================================
|                INFORMACION DE LA CANCION                     |
----------------------------------------------------------------
Posicion en el Ranking: 5
Nombre de la Cancion: Erase una Bestia
Nombre del Autor: Jose Madero
Nombre del Inteprete: Jose Madero
Nombre del Archivo MP3EbestJM.mp3

================================================================
6 Salir
Elige el atributo que quieras cambiar (1-6):
```

Cambiemos el nombre del MP3:

```
============================================================
|                  INFORMACION DE LA CANCION                |
------------------------------------------------------------
Posicion en el Ranking: 5
Nombre de la Cancion: Erase una Bestia
Nombre del Autor: Jose Madero
Nombre del Inteprete: Jose Madero
Nombre del Archivo MP3EbestJM.mp3

============================================================
6 Salir
Ingrese el nuevo nombre del Archivo MP3: EraseBestiaJM.mp3
Cambio Hecho con Exito![Enter] para continuar...
```

Y los cambios se reflejan en el menú principal:

```
================================================================================================================
|                                                LISTA DE EXITOS                                               |
----------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion | Nombre del Artista | Nombre del Interprete | Nombre del MP3 |
----------------------------------------------------------------------------------------------------------------
| 0          | 5       | Erase una Bestia     | Jose Madero        | Jose Madero           | EraseBestiaJM.mp3 |
----------------------------------------------------------------------------------------------------------------
```

Para hacer más real este funcionamiento, carguemos datos con el comando [G] para leer del disco:

```
================================================================================================================
|                                                LEER ARCHIVO                                                  |
----------------------------------------------------------------------------------------------------------------
Tenga en Cuenta que los Archivos se Sobreescribiran
Ingrese el Nombre del Archivo a Cargar sus Datos: database.csv
Archivos Cargados Con Exito!
================================================================================================================
[Enter] para continuar...
```

Y al regresar al menú principal;

```
================================================================================================================
|                                                LISTA DE EXITOS                                               |
----------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion | Nombre del Artista | Nombre del Interprete | Nombre del MP3 |
----------------------------------------------------------------------------------------------------------------
| 0          | 1023    | Mercedes             | José Madero        | José Madero           | mercedes_jm.mp3, |
| 1          | 235     | Karmadame            | León Larregui      | Zoé Zoé               | karmadame_zoe.mp3, |
| 2          | 789     | El Duelo             | León Larregui      | Zoé Zoé               | elduelo_zoe.mp3, |
| 3          | 2177    | Hit Me Hard          | The Weeknd         | The Weeknd            | hitme_tw.mp3, |
| 4          | 1421    | Fiebre               | León Larregui      | Zoé Zoé               | fiebre_zoe.mp3, |
| 5          | 2250    | In Your Eyes         | Abel Tesfaye       | The Weeknd            | inyoureyes_tw.mp3, |
| 6          | 1502    | No Hay Mal Que Dure  | León Larregui      | Zoé Zoé               | nomal_dure_zoe.mp3, |
| 7          | 2228    | Home                 | The Weeknd         | The Weeknd            | home_tw.mp3, |
| 8          | 315     | Labios Rotos         | León Larregui      | Zoé Zoé               | labiosrotos_zoe.mp3, |
| 9          | 2244    | Popular              | León Larregui      | Zoé Zoé               | popular_zoe.mp3, |
| 10         | 1987    | Lunes 28             | José Madero        | José Madero           | lunes28_jm.mp3, |
| 11         | 1755    | Hablemos del Campo   | José Madero        | José Madero           | habcamp_jm.mp3, |
| 12         | 1673    | Día de Mayo          | José Madero        | José Madero           | diademayo_jm.mp3, |
| 13         | 904     | Los Malaventurados No Lloran | José Madero | José Madero           | malav_jm.mp3, |
| 14         | 456     | Narcisista por Excelencia | José Madero   | José Madero           | narc_ex_jm.mp3, |
| 15         | 1222    | Rayo de Luz          | José Madero        | José Madero           | rayoluz_jm.mp3, |
| 16         | 2099    | Gardenias 87         | José Madero        | José Madero           | gardenias_jm.mp3, |
| 17         | 1876    | Quince Mil Días      | José Madero        | José Madero           | quincemil_jm.mp3, |
| 18         | 1551    | Love                 | León Larregui      | Zoé Zoé               | love_zoe.mp3, |
| 19         | 2503    | Soñé                 | León Larregui      | Zoé Zoé               | sone_zoe.mp3, |
| 20         | 302     | Luna                 | León Larregui      | Zoé Zoé               | luna_zoe.mp3, |
| 21         | 1988    | Callaíta (cover)     | Abel Tesfaye       | The Weeknd            | callaita_tw.mp3, |
| 22         | 2345    | Blinding Lights      | Abel Tesfaye       | The Weeknd            | blinding_tw.mp3, |
| 23         | 1764    | Save Your Tears      | Abel Tesfaye       | The Weeknd            | save_tw.mp3, |
| 24         | 2999    | After Hours          | Abel Tesfaye       | The Weeknd            | afterh_tw.mp3, |
| 25         | 2410    | Heartless            | Abel Tesfaye       | The Weeknd            | heartless_tw.mp3, |
| 26         | 2022    | Midnight City (cover) | Laura Pergolizzi  | LP Pergolizzi         | midnight_lp.mp3, |
| 27         | 1989    | Lost On You          | Laura Pergolizzi   | LP Pergolizzi         | lost_lp.mp3, |
| 28         | 2111    | Strange              | Laura Pergolizzi   | LP Pergolizzi         | strange_lp.mp3, |
| 29         | 1678    | Tokyo                | Laura Pergolizzi   | LP Pergolizzi         | tokyo_lp.mp3, |
| 30         | 2122    | The Hills            | Abel Tesfaye       | The Weeknd            | hills_tw.mp3, |
| 31         | 1760    | Earned It            | Abel Tesfaye       | The Weeknd            | earned_tw.mp3, |
| 32         | 2433    | Friends              | Abel Tesfaye       | The Weeknd            | friends_tw.mp3, |
| 33         | 1888    | Monster              | Abel Tesfaye       | The Weeknd            | monster_tw.mp3, |
| 34         | 1977    | Belong To The World (cover) | Laura Pergolizzi | LP Pergolizzi       | belong_lp.mp3, |
| 35         | 1901    | Gasolina (cover)     | Laura Pergolizzi   | LP Pergolizzi         | gasolina_lp.mp3, |
```

El archivo .csv insertado es el mismo que en actividades anteriores por lo que sigue siendo compatible.

Eliminemos una canción con [C<n>]:

```
============================================================
|                  INFORMACION DE LA CANCION                |
------------------------------------------------------------
Posicion en el Ranking: 2022
Nombre de la Cancion: Midnight City (cover)
Nombre del Autor: Laura Pergolizzi
Nombre del Inteprete: LP Pergolizzi
Nombre del Archivo MP3midnight_lp.mp3,

============================================================
Esta seguro que desea eliminar esta cancion? (1. Si/ 2. No):
```

Al darle que sí:

```
============================================================
|                  INFORMACION DE LA CANCION                |
------------------------------------------------------------
Posicion en el Ranking: 2022
Nombre de la Cancion: Midnight City (cover)
Nombre del Autor: Laura Pergolizzi
Nombre del Inteprete: LP Pergolizzi
Nombre del Archivo MP3midnight_lp.mp3,

============================================================
Esta seguro que desea eliminar esta cancion? (1. Si/ 2. No):
Cancion Eliminada con Exito!
[Enter] para continuar...
```

Y al volver al menú principal, esta canción que ocupaba el puesto 26, ya no está presente:

```
| 22 | 2345 | Blinding Lights | Abel Tesfaye | The Weeknd | blinding_tw.mp3,
| 23 | 1764 | Save Your Tears | Abel Tesfaye | The Weeknd | save_tw.mp3,
| 24 | 2999 | After Hours | Abel Tesfaye | The Weeknd | afterh_tw.mp3,
| 25 | 2410 | Heartless | Abel Tesfaye | The Weeknd | heartless_tw.mp3,
| 26 | 1989 | Lost On You | Laura Pergolizzi | LP Pergolizzi | lost_lp.mp3,
| 27 | 2111 | Strange | Laura Pergolizzi | LP Pergolizzi | strange_lp.mp3,
```

Mostrando los detalles de una canción con [D<n>] para la canción 0 “Mercedes” por ejemplo:

```
============================================================
|                  INFORMACION DE LA CANCION                |
------------------------------------------------------------
Posicion en el Ranking: 1023
Nombre de la Cancion: Mercedes
Nombre del Autor: José Madero
Nombre del Inteprete: José Madero
Nombre del Archivo MP3mercedes_jm.mp3,

============================================================
[Enter] para continuar...
```

La opción [F] para guardar al disco:

```
================================================
|                GUARDAR DATABASE              |
------------------------------------------------
Ingrese el Nombre que Tendra el Archivo: Database guardada con Exito!
[Enter] para continuar...
```

Sigue funcionando y generando el archivo .csv correspondiente con toda la información:

La opción [H] para el submenú de búsqueda:

```
================================================
|                  BUSCAR CANCION              |
------------------------------------------------
Existen un total de: 44 registradas.
A continuacion se muestran las opciones de busqueda:
[A] Buscar por Nombre de Cancion
[B] Buscar por Nombre del Inteprete
[R] Regresar.
Seleccione una Opcion:
```

Buscando una canción por nombre:

```
================================================
|          BUSCAR POR NOMBRE DE CANCION        |
------------------------------------------------
Ingrese el nombre de la cancion que quiera buscar: Soñé
Desea Realizar Una Busqueda Binaria o Lineal? (L/B): L
```

(Binaria aún no porque la lista no está ordenada):

```
================================================================================================================================
|                                                    LISTA DE EXITOS                                                           |
--------------------------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion          | Nombre del Artista        | Nombre del Interprete     | Nombre del MP3       |
--------------------------------------------------------------------------------------------------------------------------------
| 0          | 2503    | Soñé                          | León Larregui             | Zoé Zoé                   | sone_zoe.mp3,        |
--------------------------------------------------------------------------------------------------------------------------------
Desea Realizar Otra Busqueda?: (1.Si / 2.No):
```

En el menú, yendo a la opción de ordenamientos:

```
==================================================
|                 ORDENAR EXITOS                 |
--------------------------------------------------
Existen un total de: 44 registradas.
A continuacion se muestran las opciones sobre las cuales ordenar las canciones:
[A] Ordenar por Nombre de Cancion
[B] Ordenar por Nombre del Inteprete
[C] Ordenar por Numero de Ranking
[R] Regresar.
Seleccione una Opcion:
```

Ordenando por nombre de la canción:

```
================================================================================================================
|                                    Ordenar por Nombre de la Cancion                                          |
----------------------------------------------------------------------------------------------------------------
A continuacion, eliga un algoritmo de ordenamiento para la lista:
[A] Ordenamiento por Burbuja
[B] Ordenamiento por InsertSort
[C] Ordenamiento por SelectSort
[D] Ordenamiento por ShellSort
[E] Regresar

Ingrese una Opcion:
```

Utilicemos ShellSort:

```
================================================================================================================
|                                    Ordenar por Nombre de la Cancion                                          |
----------------------------------------------------------------------------------------------------------------
A continuacion, eliga un algoritmo de ordenamiento para la lista:
[A] Ordenamiento por Burbuja
[B] Ordenamiento por InsertSort
[C] Ordenamiento por SelectSort
[D] Ordenamiento por ShellSort
[E] Regresar

Ingrese una Opcion: D
Ordenamiento por ShellSort hecho correctamente!

Regresando...

[Enter] para continuar...
```

Y al volver al menú:

```
================================================================================================================
|                                                LISTA DE EXITOS                                               |
----------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista | Nombre del Interprete | Nombre del MP3          |
----------------------------------------------------------------------------------------------------------------
| 0          | 2999    | After Hours                     | Abel Tesfaye       | The Weeknd            | afterh_tw.mp3,          |
| 1          | 1977    | Belong To The World (cover)     | Laura Pergolizzi   | LP Pergolizzi         | belong_lp.mp3,          |
| 2          | 2345    | Blinding Lights                 | Abel Tesfaye       | The Weeknd            | blinding_tw.mp3,        |
| 3          | 1988    | Callaita (cover)                | Abel Tesfaye       | The Weeknd            | callaita_tw.mp3,        |
| 4          | 2203    | Campeones del Mundo             | José Madero        | José Madero           | campeones_jm.mp3,       |
| 5          | 2420    | Dafne                           | José Madero        | José Madero           | dafne_jm.mp3,           |
| 6          | 1673    | Día de Mayo                     | José Madero        | José Madero           | diademayo_jm.mp3,       |
| 7          | 1760    | Earned It                       | Abel Tesfaye       | The Weeknd            | earned_tw.mp3,          |
| 8          | 789     | El Duelo                        | León Larregui      | Zoé Zoé               | elduelo_zoe.mp3,        |
| 9          | 1421    | Fiebre                          | León Larregui      | Zoé Zoé               | fiebre_zoe.mp3,         |
| 10         | 2433    | Friends                         | Abel Tesfaye       | The Weeknd            | friends_tw.mp3,         |
| 11         | 2320    | Gardenias                       | José Madero        | José Madero           | gardenias2_jm.mp3,      |
| 12         | 2099    | Gardenias 87                    | José Madero        | José Madero           | gardenias_jm.mp3,       |
| 13         | 1901    | Gasolina (cover)                | Laura Pergolizzi   | LP Pergolizzi         | gasolina_lp.mp3,        |
| 14         | 1755    | Hablemos del Campo              | José Madero        | José Madero           | habcamp_jm.mp3,         |
| 15         | 2410    | Heartless                       | Abel Tesfaye       | The Weeknd            | heartless_tw.mp3,       |
| 16         | 2177    | Hit Me Hard                     | The Weeknd         | The Weeknd            | hitme_tw.mp3,           |
| 17         | 2228    | Home                            | The Weeknd         | The Weeknd            | home_tw.mp3,            |
| 18         | 2250    | In Your Eyes                    | Abel Tesfaye       | The Weeknd            | inyoureyes_tw.mp3,      |
| 19         | 235     | Karmadame                       | León Larregui      | Zoé Zoé               | karmadame_zoe.mp3,      |
| 20         | 315     | Labios Rotos                    | León Larregui      | Zoé Zoé               | labiosrotos_zoe.mp3,    |
| 21         | 2860    | Liminal (Zoé)                   | León Larregui      | Zoé Zoé               | liminal_zoe.mp3,        |
| 22         | 904     | Los Malaventurados No Lloran    | José Madero        | José Madero           | malav_jm.mp3,           |
| 23         | 1989    | Lost On You                     | Laura Pergolizzi   | LP Pergolizzi         | lost_lp.mp3,            |
| 24         | 1551    | Love                            | León Larregui      | Zoé Zoé               | love_zoe.mp3,           |
| 25         | 302     | Luna                            | León Larregui      | Zoé Zoé               | luna_zoe.mp3,           |
| 26         | 1987    | Lunes 28                        | José Madero        | José Madero           | lunes28_jm.mp3,         |
| 27         | 1023    | Mercedes                        | José Madero        | José Madero           | mercedes_jm.mp3,        |
| 28         | 1888    | Monster                         | Abel Tesfaye       | The Weeknd            | monster_tw.mp3,         |
| 29         | 1742    | Monster (Zoé cover)             | León Larregui      | Zoé Zoé               | monster_zoe.mp3,        |
| 30         | 1582    | Nada Personal                   | Luis Hernández     | Luis Hernández        | nada_pers.mp3,          |
| 31         | 456     | Narcisista por Excelencia       | José Madero        | José Madero           | narc_ex_jm.mp3,         |
| 32         | 1502    | No Hay Mal Que Dure             | León Larregui      | Zoé Zoé               | nomal_dure_zoe.mp3,     |
| 33         | 2115    | Nueve Vidas                     | José Madero        | José Madero           | nuevevidas_jm.mp3,      |
| 34         | 2244    | Popular                         | León Larregui      | Zoé Zoé               | popular_zoe.mp3,        |
| 35         | 2437    | Proyectos Invisibles            | Zoé Zoé            | Proyectos Invisibles  | proyectos_zoe.mp3,      |
```

Y ahora, probemos la búsqueda binaria por nombre de la canción:

```
================================================
|          BUSCAR POR NOMBRE DE CANCION        |
------------------------------------------------
Ingrese el nombre de la cancion que quiera buscar: Rayo de Luz
Desea Realizar Una Busqueda Binaria o Lineal? (L/B): B
```

```
================================================================================================================
|                                                LISTA DE EXITOS                                               |
----------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista        | Nombre del Interprete     | Nombre del MP3        |
----------------------------------------------------------------------------------------------------------------
| 0          | 1222    | Rayo de Luz                     | José Madero               | José Madero               | rayoluz_jm.mp3,       |
----------------------------------------------------------------------------------------------------------------
Desea Realizar Otra Busqueda?: (1.Si / 2.No):
```

Ahora, intentemos con [E] eliminar todas las canciones:

```
================================================
|          ELIMINAR TODAS LAS CANCIONES        |
------------------------------------------------
Esta seguro que desea eliminar las 1 canciones? (1. Si/ 2. No): 1
```

```
Canciones eliminadas con Exito!
Base de Datos Vacia.
[Enter] para continuar...
```

Y al menú queda vacío:

```
================================================================================================================
|                                                LISTA DE EXITOS                                               |
----------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista        | Nombre del Interprete     | Nombre del MP3        |
----------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------
Opciones:
[A] Agregar una Cancion.  [B<n>] Editar una Cancion [C<n>] Eliminar una Cancion. [D<n>]  Mostrar Detalles de Cancion.  [E]    Eliminar Todas las Canciones.

[F] Guardar la Database   [G] Leer del Disco   [H] Buscar una Cancion    [I] Ordenar Lista  [J] Salir.

Seleccione un Comando:
```

Volvamos a leer la database y hagamos una modificación, como eliminar la canción en la posición 2:

```
================================================================================================================
|                                                LISTA DE EXITOS                                               |
----------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista        | Nombre del Interprete     | Nombre del MP3        |
----------------------------------------------------------------------------------------------------------------
| 0          | 1023    | Mercedes                        | José Madero               | José Madero               | mercedes_jm.mp3,      |
| 1          | 235     | Karmadame                       | León Larregui             | Zoé Zoé                   | karmadame_zoe.mp3,    |
| 2          | 2177    | Hit Me Hard                     | The Weeknd                | The Weeknd                | hitme_tw.mp3,         |
| 3          | 1421    | Fiebre                          | León Larregui             | Zoé Zoé                   | fiebre_zoe.mp3,       |
| 4          | 2250    | In Your Eyes                    | Abel Tesfaye              | The Weeknd                | inyoureyes_tw.mp3,    |
| 5          | 1502    | No Hay Mal Que Dure             | León Larregui             | Zoé Zoé                   | nomal_dure_zoe.mp3,   |
| 6          | 2228    | Home                            | The Weeknd                | The Weeknd                | home_tw.mp3,          |
| 7          | 315     | Labios Rotos                    | León Larregui             | Zoé Zoé                   | labiosrotos_zoe.mp3,  |
| 8          | 2244    | Popular                         | León Larregui             | Zoé Zoé                   | popular_zoe.mp3,      |
| 9          | 1987    | Lunes 28                        | José Madero               | José Madero               | lunes28_jm.mp3,       |
| 10         | 1755    | Hablemos del Campo              | José Madero               | José Madero               | habcamp_jm.mp3,       |
| 11         | 1673    | Dia de Mayo                     | José Madero               | José Madero               | diademayo_jm.mp3,     |
| 12         | 904     | Los Malaventurados No Lloran    | José Madero               | José Madero               | malav_jm.mp3,         |
| 13         | 456     | Narcisista por Excelencia       | José Madero               | José Madero               | narc_ex_jm.mp3,       |
| 14         | 1222    | Rayo de Luz                     | José Madero               | José Madero               | rayoluz_jm.mp3,       |
| 15         | 2099    | Gardenias 87                    | José Madero               | José Madero               | gardenias_jm.mp3,     |
| 16         | 1876    | Quince Mil Dias                 | José Madero               | José Madero               | quincemil_jm.mp3,     |
| 17         | 1551    | Love                            | León Larregui             | Zoé Zoé                   | love_zoe.mp3,         |
| 18         | 2503    | Soñé                            | León Larregui             | Zoé Zoé                   | sone_zoe.mp3,         |
| 19         | 302     | Luna                            | León Larregui             | Zoé Zoé                   | luna_zoe.mp3,         |
| 20         | 1988    | Callaita (cover)                | Abel Tesfaye              | The Weeknd                | callaita_tw.mp3,      |
| 21         | 2345    | Blinding Lights                 | Abel Tesfaye              | The Weeknd                | blinding_tw.mp3,      |
| 22         | 1764    | Save Your Tears                 | Abel Tesfaye              | The Weeknd                | save_tw.mp3,          |
| 23         | 2999    | After Hours                     | Abel Tesfaye              | The Weeknd                | afterh_tw.mp3,        |
| 24         | 2410    | Heartless                       | Abel Tesfaye              | The Weeknd                | heartless_tw.mp3,     |
| 25         | 2022    | Midnight City (cover)           | Laura Pergolizzi          | LP Pergolizzi             | midnight_lp.mp3,      |
| 26         | 1989    | Lost On You                     | Laura Pergolizzi          | LP Pergolizzi             | lost_lp.mp3,          |
| 27         | 2111    | Strange                         | Laura Pergolizzi          | LP Pergolizzi             | strange_lp.mp3,       |
| 28         | 1678    | Tokyo                           | Laura Pergolizzi          | LP Pergolizzi             | tokyo_lp.mp3,         |
| 29         | 2122    | The Hills                       | Abel Tesfaye              | The Weeknd                | hills_tw.mp3,         |
| 30         | 1760    | Earned It                       | Abel Tesfaye              | The Weeknd                | earned_tw.mp3,        |
| 31         | 2433    | Friends                         | Abel Tesfaye              | The Weeknd                | friends_tw.mp3,       |
| 32         | 1888    | Monster                         | Abel Tesfaye              | The Weeknd                | monster_tw.mp3,       |
| 33         | 1977    | Belong To The World (cover)     | Laura Pergolizzi          | LP Pergolizzi             | belong_lp.mp3,        |
| 34         | 1901    | Gasolina (cover)                | Laura Pergolizzi          | LP Pergolizzi             | gasolina_lp.mp3,      |
| 35         | 1742    | Monster (Zoé cover)             | León Larregui             | Zoé Zoé                   | monster_zoe.mp3,      |
```

Y con registros, si intentamos salir con [F]:

```
==================================================
|               SALIR SIN GUARDAR?               |
--------------------------------------------------
Desea Guardar las canciones antes de Salir? (1. Si/ 2. No):
```

Nos dice que si queremos guardar, y si le damos que sí nos lleva al menú de guardar al disco, una vez terminado aquello o al darle que no:

```
Saliendo del Programa.
Tenga un Lindo Dia :D
[Enter] para continuar...
```

Y el programa finaliza.

Podemos observar como a pesar de las adaptaciones que hicimos para que la lista ya no contiene en su arreglo estático objetos del tipo canción, sino que ahora almacenando apuntadores a los mismos siguen en pie todas sus funcionalidades de manera correcta.

Conclusiones

Los apuntadores o punteros es un tema que personalmente me fascina, es una manera que le da al programador mucho control sobre su sistema y nos abre muchas posibilidades; ya había trabajado con ellos en Programación Orientada a Objetos, en combinación con herencia y métodos virtuales puros para utilizar polimorfismo sobre los apuntadore, no directamente sobre los objetos, daba mucho dinamismo y posibilidad; aquí, abre un gran abanico de margen de acción para nosotros, desde el hecho de aligerar mucho el costo computacional para funciones como swapData() hasta, por fin, poder hacer estructuras de datos con memoria dinámica totalmente, o sea, ahora tenemos las herramientas para no depender de un arreglo y estar anclados a un cierto tamaño de arreglo previamente definido, sino que podremos hacer que nuestras estructuras ocupen solamente el espacio que la información requiere. Cabe señalar que una lista con arreglo estático y una lista dinámica, ambas con la misma cantidad de elementos, acabará por ocupar más memoria la dinámica por el hecho de que, en adición a la información, almacena las direcciones de memoria, pero fuera de ello; manejando bien los punteros (esto se logra entendiendo qué son, que no “apuntan” y como funciona una referencia), podremos hacer muchas más cosas.

Listas simplemente ligadas, doblemente ligadas, almacenar herencias que tienen la sobreescritura de un método virtual puro, pilas y colas mucho más fáciles de implementar y sin un tope aparente en la memoria, los punteros son esenciales para un manejo optimizado del lenguaje C++ y un gran avance en nuestro conocimiento sobre las estructuras de datos; inclusive nos abren la puerta a estructuras aún más complejas como árboles, árboles AVL o grafos, cada una con sus áreas de estudio, modelos e implementaciones.

Y antes de emocionarme demasiado y como demostró la realización de esta actividad, los punteros, si bien no son muy complicados en la teoría, en la práctica hay que manejarlos con mucho cuidado y siendo responsables en su asignación y liberación, porque así como nos permiten muchas cosas, hacen que sea más factible caer en errores más difíciles de rastrear, fallos de acceso de memoria que impliquen el cierre brusco del programa o fugas de memoria sin que nos demos cuenta.

Hay todavía mucho que practicar y conocer, pero estoy emocionado por ello.