CUCEI

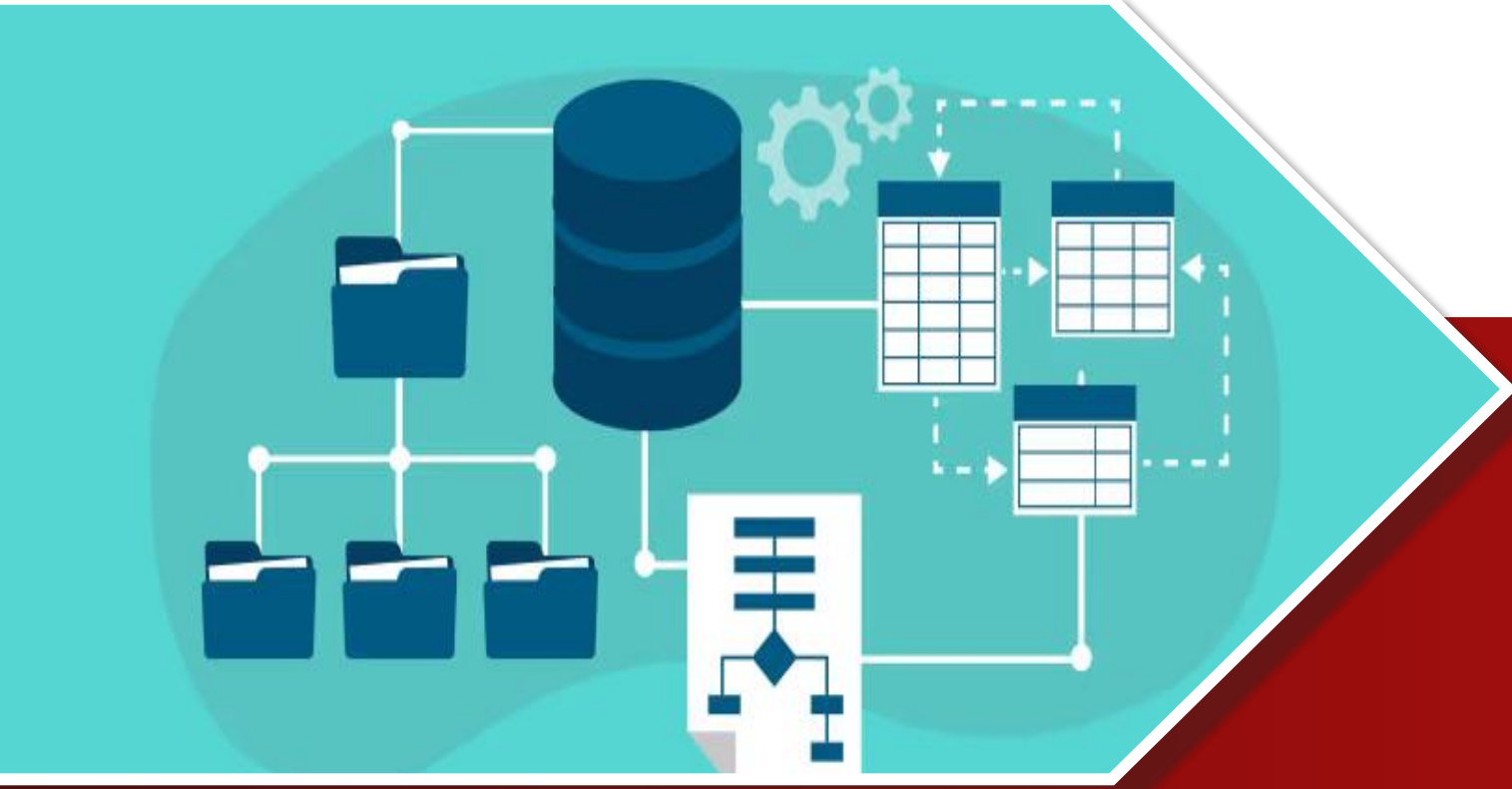
Actividad de Aprendizaje 07

Métodos de Ordenamiento Iterativo

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: 28 de Septiembre de 2025

UNIVERSIDAD DE GUADALAJARA

PIENSA Y TRABAJA

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 conocer y poner en práctica distintos métodos de ordenamiento iterativos, en clase se nos explicó acerca del ordenamiento de Burbuja, InsertSort, SelectSort y ShellSort. Desde su conceptualización hasta su traducción al lenguaje de C++, utilizándolos como métodos de la lista sobre la cual venimos trabajando

Planteamiento del Problema

En la actividad, tenemos que implementar cuatro métodos de ordenamiento a disposición de la elección del usuario para realizar un ordenamiento en base a: el nombre de la canción o el nombre del interprete; para ello, y agilizando mucho el trabajo, utilizaremos comparadores explícitos que ya tienen nuestros objetos declarados e implementados, tenemos como comparación implícita el atributo del ranking de la canción, y como comparadores explícitos los demás atributos que caracterizan a una canción, es decir, los compareBySongName y compareByInterprete, estos como métodos int ya los tenemos implementados y funcionando correctamente. En cuanto a los métodos de la lista de ordenamiento, las conocimos y programamos en clase, por lo que solo resta aplicar mediante polimorfismo, una sobrecarga de estos métodos con la comparación explícita y que se comparen como referencia del 0 para tener métodos fáciles de utilizar y ya adaptados al programa.

Con esto, tendremos la funcionalidad lista para implementar, solo bastaría adaptarles sus respectivas pantallas de la interfaz de usuario desde la clase Menú, para esto, aprovecharemos los múltiples métodos que ahorran mucho código que tenemos aquí como por ejemplo la de crear el Header con el título de la ventana, readChar que programamos en la actividad anterior para hacer un submenú de ordenamiento y manejarlo con un SwitchCase. De la misma manera manejaremos la selección de un método de ordenamiento, en el mismo menú pediremos un char para que se elija el método de ordenamiento y con un switchCase se elegirá uno u otro, después de ordenarlo, no hay realmente necesidad de repetir esta pantalla, por lo que regresamos al menú de ordenamiento, no sin antes avisarle al usuario que el ordenamiento en cuestión fue realizado con éxito.

Confirmaremos el ordenamiento con el menú principal donde se muestra la lista completa de canciones, y aprovecharemos para probar que la búsqueda binaria funcione correctamente y la radiodifusora no nos regañe otra vez.

Programación

Este paso, si bien con un buen planteamiento previo es más fácil, en este caso, las herramientas que necesitábamos en su mayoría ya estaban programadas como los CompareBy o los ordenamientos crudos que hicimos en clase, por lo que el resto solo fue unas pequeñas adaptaciones, en su mayoría las pantallas, que ni verificaciones tuvimos que volver a implementar, ya tenemos métodos que nos dan mayor seguridad para que no se cuele un carácter vacío, una opción incorrecta o demás y nuestro programa funcione de manera inesperada. Dado esto, la implementación de la idea fue bastante sencilla.

Para seguir con la costumbre, corregí un par de bugs que noté recién de la actividad pasada, me gustaría que, si esta actividad se va reciclando, no solo incorporemos las nuevas funcionalidades, sino que también vayamos refinando lo ya existente. En esta ocasión, noté que, en las búsquedas, como en el programa no solo mostramos la primera coincidencia, sino que recopilamos todas ellas en una lista temporal, no se limpia la lista al final, por lo que si buscaste algo que sí existía y le dabas a volver a buscar, esas búsquedas se quedarían y mostrarían en cuanto vuelva al mismo punto, por lo que al final, introduje un deleteAll() para limpiar la lista auxiliar. Como este solucioné un par de bugs simples.

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[ARRAYSIZE];
  int last;

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

  void swapData(T&, T&);

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

  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) {}

template <class T, int ARRAYSIZE>
void List<T, ARRAYSIZE>::copyAll(const List<T, ARRAYSIZE>& other) {
  for (int i = 0; i < other.last; i++)
    this->data[i] = other.data[i];
  this->last = other.last;
}

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) {
  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 true;

  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 true;

  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] = 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)");

  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>
void List<T, ARRAYSIZE>::deleteAll() {
  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++;

  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;
  std::string aux;

  try {
    while (is >> obj) {
      if (!list.isFull())
        list.data[++list.last] = 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->toString();
      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 << "5 Salir\n";

  editOption = readInteger(
      oss.str() + "Elige el atributo que quieras cambiar (1-5): ", 1, 5);

  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:
      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, 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 << 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;
    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;
      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

```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

```
#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

Para mostrar la ejecución del programa, pasaremos por estas nuevas opciones de ordenamiento iterativo y por la búsqueda binario.

Empezando con el Menú después de haber leído un database:

```
                                                    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,        |
```

Tenemos varios registros, y en las opciones:

```
| 38         | 2115    | Nueve Vidas                     | José Madero             | José Madero             | nuevevidas_jm.mp3,      |
| 39         | 2203    | Campeones del Mundo             | José Madero             | José Madero             | campeones_jm.mp3,       |
| 40         | 2320    | Gardenias                       | José Madero             | José Madero             | gardenias2_jm.mp3,      |
| 41         | 2860    | Liminal (Zoé)                   | León Larregui           | Zoé Zoé                 | liminal_zoe.mp3,        |
| 42         | 2437    | Proyectos Invisibles            | Zoé Zoé                 | Proyectos Invisibles    | proyectos_zoe.mp3,      |
| 43         | 2307    | What You Need                   | The Weeknd              | The Weeknd              | whatneed_tw.mp3,        |
| 44         | 1582    | Nada Personal                   | Luis Hernández          | Luis Hernández          | nada_pers.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:
```

Ahora tenemos como comando [I] para ordenar la lista, si le presionamos:

```
================================================
|                ORDENAR EXITOS                |
------------------------------------------------
Existen un total de: 45 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:
```

Cabe destacar que si ingresamos a este menú si registros nos aparecerá lo siguiente:

```
+------------------------------------------------------+
+            No hay Canciones Registradas Aun          +
+              Regresando al Menu...                   +
+------------------------------------------------------+
[Enter] para continuar...
```

O por el contrario, si entramos con una única canción registrada:

```
Solo hay 1 cancion registrada.
No se requiere Ordenamiento.
[Enter] para continuar...
```

Siguiendo con el submenú, intentamos ordenar 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:
```

Aquí es donde el usuario puede seleccionar el tipo de ordenamiento que desee; comencemos con el ordenamiento de Burbuja:

```
================================================================================================================
|                                    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: A
Ordenamiento por Burbuja hecho correctamente!.

Regresando...

[Enter] para continuar...
```

Al hacer darle al [Enter], regresaremos al submenú de ordenamiento. Si regresamos al menú principal veremos esto:

```
================================================================================================================
|                                                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         | 2022    | Midnight City (cover)           | Laura Pergolizzi      | LP Pergolizzi         | midnight_lp.mp3,        |
| 29         | 1888    | Monster                         | Abel Tesfaye          | The Weeknd            | monster_tw.mp3,         |
| 30         | 1742    | Monster (Zoé cover)             | León Larregui         | Zoé Zoé               | monster_zoe.mp3,        |
| 31         | 1582    | Nada Personal                   | Luis Hernández        | Luis Hernández        | nada_pers.mp3,          |
| 32         | 456     | Narcisista por Excelencia       | José Madero           | José Madero           | narc_ex_jm.mp3,         |
| 33         | 1502    | No Hay Mal Que Dure             | León Larregui         | Zoé Zoé               | nomal_dure_zoe.mp3,     |
| 34         | 2115    | Nueve Vidas                     | José Madero           | José Madero           | nuevevidas_jm.mp3,      |
| 35         | 2244    | Popular                         | León Larregui         | Zoé Zoé               | popular_zoe.mp3,        |
```

Observamos como las canciones se ordenaron correctamente por nombre de las mismas. Regresemos al menú de ordenamientos y seleccionemos que queremos que ordene por nombre del Interprete:

```
================================================================================================================
|                                Ordenar por Interprete 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:
```

En este caso, ingresaremos [B] para ordenar por InsertSort:

```
================================================================================================================
|                                Ordenar por Interprete 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: B
Ordenamiento por InserSor hecho correctamente!

Regresando...

[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          | 2203    | Campeones del Mundo           | José Madero           | José Madero           | campeones_jm.mp3,       |
| 1          | 2420    | Dafne                         | José Madero           | José Madero           | dafne_jm.mp3,           |
| 2          | 1673    | Día de Mayo                   | José Madero           | José Madero           | diademayo_jm.mp3,       |
| 3          | 2320    | Gardenias                     | José Madero           | José Madero           | gardenias2_jm.mp3,      |
| 4          | 2099    | Gardenias 87                  | José Madero           | José Madero           | gardenias_jm.mp3,       |
| 5          | 1755    | Hablemos del Campo            | José Madero           | José Madero           | habcamp_jm.mp3,         |
| 6          | 904     | Los Malaventurados No Lloran  | José Madero           | José Madero           | malav_jm.mp3,           |
| 7          | 1987    | Lunes 28                      | José Madero           | José Madero           | lunes28_jm.mp3,         |
| 8          | 1023    | Mercedes                      | José Madero           | José Madero           | mercedes_jm.mp3,        |
| 9          | 456     | Narcisista por Excelencia     | José Madero           | José Madero           | narc_ex_jm.mp3,         |
| 10         | 2115    | Nueve Vidas                   | José Madero           | José Madero           | nuevevidas_jm.mp3,      |
| 11         | 1876    | Quince Mil Días               | José Madero           | José Madero           | quincemil_jm.mp3,       |
| 12         | 1222    | Rayo de Luz                   | José Madero           | José Madero           | rayoluz_jm.mp3,         |
| 13         | 1977    | Belong To The World (cover)   | Laura Pergolizzi      | LP Pergolizzi         | belong_lp.mp3,          |
| 14         | 1901    | Gasolina (cover)              | Laura Pergolizzi      | LP Pergolizzi         | gasolina_lp.mp3,        |
| 15         | 1989    | Lost On You                   | Laura Pergolizzi      | LP Pergolizzi         | lost_lp.mp3,            |
| 16         | 2022    | Midnight City (cover)         | Laura Pergolizzi      | LP Pergolizzi         | midnight_lp.mp3,        |
| 17         | 2111    | Strange                       | Laura Pergolizzi      | LP Pergolizzi         | strange_lp.mp3,         |
| 18         | 1678    | Tokyo                         | Laura Pergolizzi      | LP Pergolizzi         | tokyo_lp.mp3,           |
| 19         | 1582    | Nada Personal                 | Luis Hernández        | Luis Hernández        | nada_pers.mp3,          |
| 20         | 2437    | Proyectos Invisibles          | Zoé Zoé               | Proyectos Invisibles  | proyectos_zoe.mp3,      |
| 21         | 2999    | After Hours                   | Abel Tesfaye          | The Weeknd            | afterh_tw.mp3,          |
| 22         | 2345    | Blinding Lights               | Abel Tesfaye          | The Weeknd            | blinding_tw.mp3,        |
| 23         | 1988    | Callaita (cover)              | Abel Tesfaye          | The Weeknd            | callaita_tw.mp3,        |
| 24         | 1760    | Earned It                     | Abel Tesfaye          | The Weeknd            | earned_tw.mp3,          |
| 25         | 2433    | Friends                       | Abel Tesfaye          | The Weeknd            | friends_tw.mp3,         |
| 26         | 2410    | Heartless                     | Abel Tesfaye          | The Weeknd            | heartless_tw.mp3,       |
| 27         | 2177    | Hit Me Hard                   | The Weeknd            | The Weeknd            | hitme_tw.mp3,           |
| 28         | 2228    | Home                          | The Weeknd            | The Weeknd            | home_tw.mp3,            |
| 29         | 2250    | In Your Eyes                  | Abel Tesfaye          | The Weeknd            | inyoureyes_tw.mp3,      |
| 30         | 1888    | Monster                       | Abel Tesfaye          | The Weeknd            | monster_tw.mp3,         |
| 31         | 1764    | Save Your Tears               | Abel Tesfaye          | The Weeknd            | save_tw.mp3,            |
| 32         | 2122    | The Hills                     | Abel Tesfaye          | The Weeknd            | hills_tw.mp3,           |
| 33         | 2307    | What You Need                 | The Weeknd            | The Weeknd            | whatneed_tw.mp3,        |
| 34         | 789     | El Duelo                      | León Larregui         | Zoé Zoé               | elduelo_zoe.mp3,        |
| 35         | 1421    | Fiebre                        | León Larregui         | Zoé Zoé               | fiebre_zoe.mp3,         |
```

Podemos observar como los interpretes se agrupan y se muestran de manera ordenada.

Probemos ahora el ordenamiento por SelectSort, y para ello, ordenemos por ranking:

```
=============================================================================================================
|                                    Ordenar por Ranking 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:
```

```
=============================================================================================================
|                                    Ordenar por Ranking 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: C
Ordenamiento por DataSelect hecho correctamente!

Regresando...

[Enter] para continuar...
```

Y al regresando al menú principal:

```
=============================================================================================================
|                                              LISTA DE EXITOS                                              |
-------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion          | Nombre del Artista | Nombre del Interprete | Nombre del MP3          |
-------------------------------------------------------------------------------------------------------------
| 0          | 235     | Karmadame                     | León Larregui      | Zoé Zoé               | karmadame_zoe.mp3,      |
| 1          | 302     | Luna                          | León Larregui      | Zoé Zoé               | luna_zoe.mp3,           |
| 2          | 315     | Labios Rotos                  | León Larregui      | Zoé Zoé               | labiosrotos_zoe.mp3,    |
| 3          | 456     | Narcisista por Excelencia     | José Madero        | José Madero           | narc_ex_jm.mp3,         |
| 4          | 789     | El Duelo                      | León Larregui      | Zoé Zoé               | elduelo_zoe.mp3,        |
| 5          | 904     | Los Malaventurados No Lloran  | José Madero        | José Madero           | malav_jm.mp3,           |
| 6          | 1023    | Mercedes                      | José Madero        | José Madero           | mercedes_jm.mp3,        |
| 7          | 1222    | Rayo de Luz                   | José Madero        | José Madero           | rayoluz_jm.mp3,         |
| 8          | 1421    | Fiebre                        | León Larregui      | Zoé Zoé               | fiebre_zoe.mp3,         |
| 9          | 1502    | No Hay Mal Que Dure           | León Larregui      | Zoé Zoé               | nomal_dure_zoe.mp3,     |
| 10         | 1551    | Love                          | León Larregui      | Zoé Zoé               | love_zoe.mp3,           |
| 11         | 1582    | Nada Personal                 | Luis Hernández     | Luis Hernández        | nada_pers.mp3,          |
| 12         | 1673    | Día de Mayo                   | José Madero        | José Madero           | diademayo_jm.mp3,       |
| 13         | 1678    | Tokyo                         | Laura Pergolizzi   | LP Pergolizzi         | tokyo_lp.mp3,           |
| 14         | 1742    | Monster (Zoé cover)           | León Larregui      | Zoé Zoé               | monster_zoe.mp3,        |
| 15         | 1755    | Hablemos del Campo            | José Madero        | José Madero           | habcamp_jm.mp3,         |
| 16         | 1760    | Earned It                     | Abel Tesfaye       | The Weeknd            | earned_tw.mp3,          |
| 17         | 1764    | Save Your Tears               | Abel Tesfaye       | The Weeknd            | save_tw.mp3,            |
| 18         | 1876    | Quince Mil Días               | José Madero        | José Madero           | quincemil_jm.mp3,       |
| 19         | 1888    | Monster                       | Abel Tesfaye       | The Weeknd            | monster_tw.mp3,         |
| 20         | 1901    | Gasolina (cover)              | Laura Pergolizzi   | LP Pergolizzi         | gasolina_lp.mp3,        |
| 21         | 1977    | Belong To The World (cover)   | Laura Pergolizzi   | LP Pergolizzi         | belong_lp.mp3,          |
| 22         | 1987    | Lunes 28                      | José Madero        | José Madero           | lunes28_jm.mp3,         |
| 23         | 1988    | Callaíta (cover)              | Abel Tesfaye       | The Weeknd            | callaita_tw.mp3,        |
| 24         | 1989    | Lost On You                   | Laura Pergolizzi   | LP Pergolizzi         | lost_lp.mp3,            |
| 25         | 2022    | Midnight City (cover)         | Laura Pergolizzi   | LP Pergolizzi         | midnight_lp.mp3,        |
| 26         | 2099    | Gardenias 87                  | José Madero        | José Madero           | gardenias_jm.mp3,       |
| 27         | 2111    | Strange                       | Laura Pergolizzi   | LP Pergolizzi         | strange_lp.mp3,         |
| 28         | 2115    | Nueve Vidas                   | José Madero        | José Madero           | nuevevidas_jm.mp3,      |
| 29         | 2122    | The Hills                     | Abel Tesfaye       | The Weeknd            | hills_tw.mp3,           |
| 30         | 2177    | Hit Me Hard                   | The Weeknd         | The Weeknd            | hitme_tw.mp3,           |
| 31         | 2203    | Campeones del Mundo           | José Madero        | José Madero           | campeones_jm.mp3,       |
| 32         | 2228    | Home                          | The Weeknd         | The Weeknd            | home_tw.mp3,            |
| 33         | 2244    | Popular                       | León Larregui      | Zoé Zoé               | popular_zoe.mp3,        |
| 34         | 2250    | In Your Eyes                  | Abel Tesfaye       | The Weeknd            | inyoureyes_tw.mp3,      |
| 35         | 2307    | What You Need                 | The Weeknd         | The Weeknd            | whatneed_tw.mp3,        |
```

Finalmente, probemos ShellSort, y usemos otra vez el 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: D
Ordenamiento por ShellSort hecho correctamente!

Regresando...

[Enter] para continuar...
```

Y el menú principal vuelva a estar así:

```
================================================================================================================================
|                                                        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    | Callaíta (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         | 2022    | Midnight City (cover)         | Laura Pergolizzi   | LP Pergolizzi         | midnight_lp.mp3,            |
| 29         | 1888    | Monster                       | Abel Tesfaye       | The Weeknd            | monster_tw.mp3,             |
| 30         | 1742    | Monster (Zoé cover)           | León Larregui      | Zoé Zoé               | monster_zoe.mp3,            |
| 31         | 1582    | Nada Personal                 | Luis Hernández     | Luis Hernández        | nada_pers.mp3,              |
| 32         | 456     | Narcisista por Excelencia     | José Madero        | José Madero           | narc_ex_jm.mp3,             |
| 33         | 1502    | No Hay Mal Que Dure           | León Larregui      | Zoé Zoé               | nomal_dure_zoe.mp3,         |
| 34         | 2115    | Nueve Vidas                   | José Madero        | José Madero           | nuevevidas_jm.mp3,          |
| 35         | 2244    | Popular                       | León Larregui      | Zoé Zoé               | popular_zoe.mp3,            |
```

Con un ordenamiento acorde, podemos utilizar de manera correcta el algoritmo de búsqueda binaria que ya teníamos programado, volvamos a la sección de búsqueda:

```
================================================
|                BUSCAR CANCION                |
------------------------------------------------
Existen un total de: 45 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:
```

Para este punto, es de destacar que si dependiendo del atributo sobre el cuál queremos buscar, si tenemos planeado usar binarySearch, tendremos que tener ordenado por dicho atributo, es decir, para este punto tenemos la lista ordenada por el nombre de la canción,

por lo que si intentamos buscar por nombre del interprete, binarySearch puede no funcionar correctamente.

Con ello en consideración, probemos primero la búsqueda por Nombre de la Canción:

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

En este punto, ponemos [B] para señalar una búsqueda Binaria, entonces:

```
================================================================================================================
|                                                LISTA DE EXITOS                                               |
----------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista          | Nombre del Interprete       | Nombre del MP3          |
----------------------------------------------------------------------------------------------------------------
| 0          | 1989    | Lost On You                     | Laura Pergolizzi            | LP Pergolizzi               | lost_lp.mp3,            |
----------------------------------------------------------------------------------------------------------------
Desea Realizar Otra Busqueda?: (1.Si / 2.No):
```

Nos muestra correctamente la coincidencia. Al ser nombre de canción algo más específico, en nuestra lista es la única llamada “Lost On You”, pero si existiera alguna otra coincidencia aparecía aquí.

Para probar binarySearch con nombre del interprete, volvamos a ordenar la lista, usando ShellSort y ordenando por nombre del Interprete la lista queda:

```
================================================================================================================
|                                                LISTA DE EXITOS                                               |
----------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista          | Nombre del Interprete       | Nombre del MP3          |
----------------------------------------------------------------------------------------------------------------
| 0          | 2115    | Nueve Vidas                     | José Madero                 | José Madero                 | nuevevidas_jm.mp3,      |
| 1          | 904     | Los Malaventurados No Lloran    | José Madero                 | José Madero                 | malav_jm.mp3,           |
| 2          | 2099    | Gardenias 87                    | José Madero                 | José Madero                 | gardenias_jm.mp3,       |
| 3          | 1876    | Quince Mil Días                 | José Madero                 | José Madero                 | quincemil_jm.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          | 1023    | Mercedes                        | José Madero                 | José Madero                 | mercedes_jm.mp3,        |
| 8          | 1987    | Lunes 28                        | José Madero                 | José Madero                 | lunes28_jm.mp3,         |
| 9          | 1222    | Rayo de Luz                     | José Madero                 | José Madero                 | rayoluz_jm.mp3,         |
| 10         | 456     | Narcisista por Excelencia       | José Madero                 | José Madero                 | narc_ex_jm.mp3,         |
| 11         | 2320    | Gardenias                       | José Madero                 | José Madero                 | gardenias2_jm.mp3,      |
| 12         | 1755    | Hablemos del Campo              | José Madero                 | José Madero                 | habcamp_jm.mp3,         |
| 13         | 1901    | Gasolina (cover)                | Laura Pergolizzi            | LP Pergolizzi               | gasolina_lp.mp3,        |
| 14         | 1989    | Lost On You                     | Laura Pergolizzi            | LP Pergolizzi               | lost_lp.mp3,            |
| 15         | 2022    | Midnight City (cover)           | Laura Pergolizzi            | LP Pergolizzi               | midnight_lp.mp3,        |
| 16         | 1678    | Tokyo                           | Laura Pergolizzi            | LP Pergolizzi               | tokyo_lp.mp3,           |
| 17         | 2111    | Strange                         | Laura Pergolizzi            | LP Pergolizzi               | strange_lp.mp3,         |
| 18         | 1977    | Belong To The World (cover)     | Laura Pergolizzi            | LP Pergolizzi               | belong_lp.mp3,          |
| 19         | 1582    | Nada Personal                   | Luis Hernández              | Luis Hernández              | nada_pers.mp3,          |
| 20         | 2437    | Proyectos Invisibles            | Zoé Zoé                     | Proyectos Invisibles        | proyectos_zoe.mp3,      |
| 21         | 1760    | Earned It                       | Abel Tesfaye                | The Weeknd                  | earned_tw.mp3,          |
| 22         | 2177    | Hit Me Hard                     | The Weeknd                  | The Weeknd                  | hitme_tw.mp3,           |
| 23         | 2122    | The Hills                       | Abel Tesfaye                | The Weeknd                  | hills_tw.mp3,           |
| 24         | 2999    | After Hours                     | Abel Tesfaye                | The Weeknd                  | afterh_tw.mp3,          |
| 25         | 2228    | Home                            | The Weeknd                  | The Weeknd                  | home_tw.mp3,            |
| 26         | 2250    | In Your Eyes                    | Abel Tesfaye                | The Weeknd                  | inyoureyes_tw.mp3,      |
| 27         | 1988    | Callaita (cover)                | Abel Tesfaye                | The Weeknd                  | callaita_tw.mp3,        |
| 28         | 2410    | Heartless                       | Abel Tesfaye                | The Weeknd                  | heartless_tw.mp3,       |
| 29         | 1888    | Monster                         | Abel Tesfaye                | The Weeknd                  | monster_tw.mp3,         |
| 30         | 2345    | Blinding Lights                 | Abel Tesfaye                | The Weeknd                  | blinding_tw.mp3,        |
| 31         | 2433    | Friends                         | Abel Tesfaye                | The Weeknd                  | friends_tw.mp3,         |
| 32         | 1764    | Save Your Tears                 | Abel Tesfaye                | The Weeknd                  | save_tw.mp3,            |
| 33         | 2307    | What You Need                   | The Weeknd                  | The Weeknd                  | whatneed_tw.mp3,        |
| 34         | 235     | Karmadame                       | León Larregui               | Zoé Zoé                     | karmadame_zoe.mp3,      |
| 35         | 2244    | Popular                         | León Larregui               | Zoé Zoé                     | popular_zoe.mp3,        |
```

Ahora, seleccionemos buscar por nombre del Interete:

```
================================================================
|              BUSCAR POR INTERPRETE DE LA CANCION             |
----------------------------------------------------------------
Ingrese el Nombre del Interprete: José
Ingrese el Apellido: Madero
Desea Realizar Una Busqueda Binaria o Lineal? (L/B): B
```

Al realizar esta búsqueda:

```
                                                    LISTA DE EXITOS                                                    |
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista          | Nombre del Interprete       | Nombre del MP3          |
| 0          | 456     | Narcisista por Excelencia       | José Madero                 | José Madero                 | narc_ex_jm.mp3,         |
| 1          | 904     | Los Malaventurados No Lloran    | José Madero                 | José Madero                 | malav_jm.mp3,           |
| 2          | 1023    | Mercedes                        | José Madero                 | José Madero                 | mercedes_jm.mp3,        |
| 3          | 1222    | Rayo de Luz                     | José Madero                 | José Madero                 | rayoluz_jm.mp3,         |
| 4          | 1673    | Día de Mayo                     | José Madero                 | José Madero                 | diademayo_jm.mp3,       |
| 5          | 1755    | Hablemos del Campo              | José Madero                 | José Madero                 | habcamp_jm.mp3,         |
| 6          | 1876    | Quince Mil Días                 | José Madero                 | José Madero                 | quincemil_jm.mp3,       |
| 7          | 1987    | Lunes 28                        | José Madero                 | José Madero                 | lunes28_jm.mp3,         |
| 8          | 2099    | Gardenias 87                    | José Madero                 | José Madero                 | gardenias_jm.mp3,       |
| 9          | 2115    | Nueve Vidas                     | José Madero                 | José Madero                 | nuevevidas_jm.mp3,      |
| 10         | 2203    | Campeones del Mundo             | José Madero                 | José Madero                 | campeones_jm.mp3,       |
| 11         | 2320    | Gardenias                       | José Madero                 | José Madero                 | gardenias2_jm.mp3,      |
| 12         | 2420    | Dafne                           | José Madero                 | José Madero                 | dafne_jm.mp3,           |
Desea Realizar Otra Busqueda?: (1.Si / 2.No):
```

José Madero es el artista que en la lista ocupa las primeras posiciones, busquemos ahora un interprete a medias de la misma como “The Weeknd”:

```
                    BUSCAR POR INTERPRETE DE LA CANCION
Ingrese el Nombre del Interprete: The
Ingrese el Apellido: Weeknd
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          | 1760    | Earned It                       | Abel Tesfaye                | The Weeknd                  | earned_tw.mp3,          |
| 1          | 1764    | Save Your Tears                 | Abel Tesfaye                | The Weeknd                  | save_tw.mp3,            |
| 2          | 1888    | Monster                         | Abel Tesfaye                | The Weeknd                  | monster_tw.mp3,         |
| 3          | 1988    | Callaita (cover)                | Abel Tesfaye                | The Weeknd                  | callaita_tw.mp3,        |
| 4          | 2122    | The Hills                       | Abel Tesfaye                | The Weeknd                  | hills_tw.mp3,           |
| 5          | 2177    | Hit Me Hard                     | The Weeknd                  | The Weeknd                  | hitme_tw.mp3,           |
| 6          | 2228    | Home                            | The Weeknd                  | The Weeknd                  | home_tw.mp3,            |
| 7          | 2250    | In Your Eyes                    | Abel Tesfaye                | The Weeknd                  | inyoureyes_tw.mp3,      |
| 8          | 2307    | What You Need                   | The Weeknd                  | The Weeknd                  | whatneed_tw.mp3,        |
| 9          | 2345    | Blinding Lights                 | Abel Tesfaye                | The Weeknd                  | blinding_tw.mp3,        |
| 10         | 2410    | Heartless                       | Abel Tesfaye                | The Weeknd                  | heartless_tw.mp3,       |
| 11         | 2433    | Friends                         | Abel Tesfaye                | The Weeknd                  | friends_tw.mp3,         |
| 12         | 2999    | After Hours                     | Abel Tesfaye                | The Weeknd                  | afterh_tw.mp3,          |
Desea Realizar Otra Busqueda?: (1.Si / 2.No):
```

Tenemos completas sus 13 canciones de la lista.

¡Observamos como tanto los distintos métodos iterativos de ordenamiento como la binarySearch funcionan de manera correcta!.

Conclusiones

Los métodos de ordenamiento de búsqueda muchas veces los vemos como algoritmos que ya están ahí, para que simplemente los usemos, pero el hecho de conocer fundamentos de algunos de ellos y programarlos amplia y desarrolla esa habilidad de traducir un determinado planteamiento al código; poder utilizar las herramientas, en este caso de C++, para trasladar lo fundamentado. Algoritmos como BubbleSort o SelectSort son muy intuitivos, de hecho, en proyecto de semestres anteriores, ante la necesidad de ordenar un conjunto de datos y aún sin conocer a fondo un algoritmo de búsqueda fijo y queriendo aceptar el reto antes de investigar una solución que seguramente sería más eficiente, di con una versión similar a SelectSort, y mi lógica fue pensar como si ordenara yo mismo un conjunto de números, primero selecciono el menor de ellos y lo pongo al inicio de otra lista, luego busco el segundo menor y lo pongo siguiente y así sucesivamente.

Sin embargo, ShellSort es un poco más lioso, durante la clase y gracias a la explicación y corrida de escritorio del profesor pude entender cómo opera a cierto nivel, pero el fundamento matemático detrás debe ser muy interesante, también la selección de una secuencia para manejar el tamaño del diferencial y que no sea solamente un factor como ½. Observar como poco a poco quedan en su lugar es satisfactorio, y la aplicación es un muy buen complemento con lo que ya hemos visto en clases pasadas con los comparadores explícitos e implícitos, si ya teníamos la capacidad de elegir reciclando mucho código el atributo sobre el cual queremos realizar una búsqueda pasando una función como parámetro a una sobrecarga del método de búsqueda deseado, aquí aplica lo mismo, utilizamos un polimorfismo y comparadores explícitos para hacer muy fácil el requisito de la tarea, y esto me encanta, porque evidenciamos que aquello con lo que estamos trabajando y sobre lo cual vamos mejorándolo poco a poco, está hecho de buena manera, eficiente y mantenible, un código con objetos completos que, además de funcionar, tienen las herramientas y posibilidades de expandir funcionalidades del código sin tener que pasar por una gran refactorización o un proceso muy lioso.

Los métodos de ordenamiento es todo un mundo con mucha teoría detrás, y embarcarnos desde los intuitivos hasta algunos de los más eficientes me parece ideal para desarrollar ese pensamiento algorítmico y comprender qué hace la computadora y la memoria si pensamos en utilizar uno similar ya creado.