CUCEI

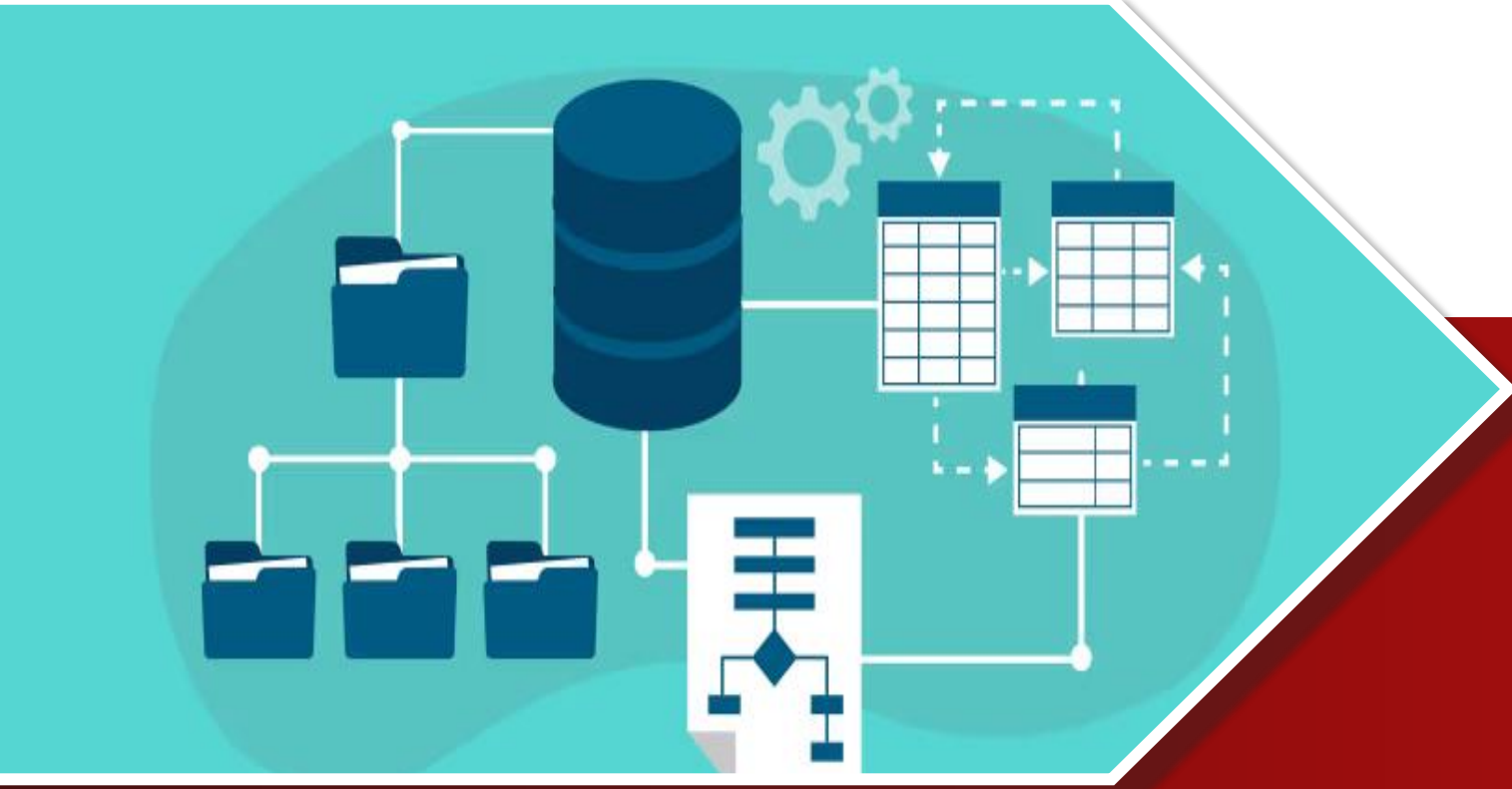
Actividad de Aprendizaje 06

Métodos De Búsqueda

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: 19 de Septiembre de 2025

Contenido

Test de Autoevaluación

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

Introducción y Abordaje del Problema

Esta actividad tiene como objetivo poner en implementación lo visto en clase relacionado con los métodos de búsqueda de búsqueda lineal y búsqueda binaria, además de los comparadores explícitos por medio del paso de funciones como parámetros. Conocimos la complejidad de ambos tipos de búsquedas, el uso de compareBy y demás estrategias para implementar funcionalidades alineadas con POO.

Planteamiento del Problema

Para empezar, ante de ir directamente a implementar lo relacionado con las búsquedas y los comparadores explícitos, hay algunas cosas que quería seguir mejorando del código; en primer lugar, la list.hpp que utilice usaba un toString(), que si bien, era muy estético, dejaba de lado la plantilla de la lista, ya que al hacerlo en formato de tabla, dejaba solo su uso para la clase específica, fue cómodo implementarlo así para el programa, pero para seguir los lineamientos de POO y hacer que la list.hpp sea tan flexible como debería ser con la template.

La función de buscar, en las indicaciones se menciona que será buscar por nombre de la canción y por interprete; planeo hacer un submenú de búsqueda donde se pueda elegir el filtro de búsqueda. Para esto, usaremos otro sistema de comandos como el del menú principal mediante, char, como no necesitamos señalar un número como en el menú principal, leeremos solo el carácter mediante de un cin >> seguido de un cin.ignore(), para esto y con tal de asegurarnos de tener una respuesta válida, haremos una función para leer un carácter llamada readChar, así como tenemos un readInteger o un redLinePrompt.

Con esto, las funciones de búsqueda lineal y binaria serán implementadas en la list.hpp, con su forma implícita y explícita pasando funciones como parámetro, así mismo incluyendo el static int compare, el compareTo y los compareBy correspondientes. Añadiremos también a la canción el atributo del nombre del archivo MP3 como tipo string con todo lo que ello conlleva, desde los toString() hasta los operadores de flujo y el operador de asignación, copyAll etc.

Para estas funciones de búsqueda existe una situación que puede darse, si por ejemplo, buscamos un intérprete, puede haber más de una canción que pueda estar en el

ranking, y las funciones de findData, solo retornan la posición del primer elemento en secuencia, mi planteamiento es que las funciones muestren todas las coincidencias, ya sean con el interprete o con el mismo nombre.

Y con todo esto sobre el papel, vamos a la programación.

Programación

Para recuperar la polivalencia de la lista, haré que el toString() sea un simple recorrido de arreglo, llamando el toString() individual de cada objeto y dando un salto de línea. Para mantener la estética de tabla, modificaré el toString() individual de Song, ahora, tendrá el toString() básico que se imprimirá como líneas de una tabla, y un toStringOnly() para cuando se quiera mostrar como única canción, mostrando una tabla más espaciosa. Podría hacerse también como un solo toString() con un parámetro booleano por defecto en true que determine si se tendrá un string en formato de fila o en formato de tabla individual. Esto también requiere de una adaptación en el menú, completando la tabla, se creará una función llamada tal vez songTable() donde, mediante el toString() de la lista se imprima la tabla en condiciones, de esta manera, adaptamos que la list.hpp tenga esa flexibilidad propia de las plantillas y seguimos con la estética que queríamos desde un principio.

Para la función readChar anteriormente planteada, se debe tener como parámetros un string como prompt, y otra cosa, para un char no tenemos como tal un rango en el que podemos señalar que es válido, si pudiéramos ingresar A o Z, no podemos hacerlo como un rango de ASCII o algo parecido, así que lo que haremos será pasar arreglo (esto realmente es un apuntador al primer elemento), con los caracteres permitidos; entonces, la lógica será una estructura iterativa que se repetirá indefinidamente hasta que se ingrese un carácter válido, esto lo hacemos recorriendo el arreglo mediante aritmética de apuntadores hasta que lleguemos a un carácter nulo, si antes de esto, el carácter coincide, retornamos esa elección, si no, informamos del error y repetimos.

La parte de búsqueda, para recopilar todas las coincidencias, será de la siguiente manera, usaremos dos listas auxiliares, una que será igual al dato (de esta iremos eliminando elementos) y otra vacía donde recopilaremos las coincidencias; el algoritmo será el siguiente: tendremos una iteración que de la lista auxiliar busca la primera coincidencia con el filtro usado, una vez encontrada, la añadimos a la lista recopilatoria; eliminamos el elemento de la primera lista y seguimos así. Todo esto estará encapsulado dentro de un try catch, de manera que cuando se intente hacer un retrieve de la posición -

1, se lanzará una excepción, pero la utilizaremos para salir del bucle en el try catch, y ahora, con una lista con los éxitos de interés recopilados, tal cual en el menú principal, vamos a imprimir en formato de tabla.

Código Fuente

Carpeta Include

list.hpp

```
#ifndef __LIST_H__
#define __LIST_H__

#include <fstream>
#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>&);

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

  bool isEmpty();
  bool isFull();
  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&);
```

```
  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
  int findDataL(const T&, int(const T&, const T&));
  int findDataB(const T&, 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() {
  return this->last == -1;
}

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

// 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>
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);
}

#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 "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 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':
      insertSong();
      break;

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

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

```
      deleteSong(index);
    break;

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

      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 << "No existe un registro de una cancion del interprete: "
          << searchedName.toString();
    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;
    }

    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::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

```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 de este programa, mostraremos los cambios más significativos, considerando que este programa fue antes ya parte de una tarea, comenzando por el menú:

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

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

Seleccione un Comando:
```

Ahora tenemos la opción de Buscar una canción con el comando H, si intentamos acceder a él sin tener registros:

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

Cargaremos un dataset para representar bien las nuevas funcionalidades:

```
================================================================================================================================
|                                                        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    | Dia 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         |
```

Ahora, entremos con el comando [H] al menú de buscar una canción

```
================================================
|                 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:
```

Aquí entra en juego el método de readChar, por lo que no acepta char's diferentes:

```
Opcion invalida
Intentelo nuevamente
[Enter] para continuar...
```

Vayamos a Buscar por el nombre del Interprete:

```
=====================================================================
|               BUSCAR POR INTERPRETE DE LA CANCION                 |
---------------------------------------------------------------------
Ingrese el Nombre del Interprete: Jose
```

Nos pide el nombre y un apellido del interprete para buscar y recopilar coincidencias

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

A este punto, mediante el readChar, se le pide al usuario L o B para determinar entre buscar mediante el algoritmo lineal y el binario. Usaremos la busqueda Lineal, dado que como se indica en la actividad, la busqueda binaria, al usuario poder ingresar un orden deseado y no estar ordenada, puede no funcionar correctamente.

Antes de realizar la búsqueda anterior, si ingresamos un artista que no esté en el ranking se nos mostrará el siguiente mensaje:

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

No existe un registro de una cancion del interprete: Humberto Rodríguez
Desea Realizar Otra Busqueda?: (1.Si / 2.No):
```

Se nos avisa que no hay coincidencias y pregunta si queremos realizar otra búsqueda.

Busquemos ahora a un artista con múltiples canciones en el ranking:

```
================================================================================================================================
|                                                        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):
```

Se nos muestra una tabla donde se recopilan todos sus éxitos presentes en el ranking, ordenados desde la posición del ranking más alta a la más baja.

La búsqueda por nombre de canción funciona similar:

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

```
================================================================================================================================
|                                                        LISTA DE EXITOS                                                       |
--------------------------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion          | Nombre del Artista | Nombre del Interprete | Nombre del MP3          |
--------------------------------------------------------------------------------------------------------------------------------
| 0          | 315     | Labios Rotos                  | León Larregui      | Zoé Zoé               | labiosrotos_zoe.mp3     |
--------------------------------------------------------------------------------------------------------------------------------
Desea Realizar Otra Busqueda?: (1.Si / 2.No):
```

En este caso, solo existe una canción llamada “Labios Rotos”, por lo que solo se recopila ese resultado.

Y con las modificaciones que hicimos los operadores de flujo, las funciones para guardar y cargar datos siguen funcionando correctamente.

Conclusiones

Trabajar con funciones como parámetros era algo que había visto superficialmente en otros lenguajes de programación como Python, pero en C++ nunca, y ahora, usándolas con comparadores explícitos es sumamente útil, más para reutilizar código y trabajar más inteligentemente con funciones polivalentes y objetos muy bien planteados que desempeñan sus funciones individuales de muy buena manera; en semestres pasados, yo habría hecho una función específica como "searchByName" en la lista o en una clase que la incluyera, pero con una sobrecarga de un buscador y comparadores explícitos facilitan mucho esta tarea, aún más si el programa crece y vamos agregando atributos a ciertos objetos, como en este caso, que añadimos un nombre al archivo MP3 dentro de los registros que hacemos en la lista; esta también fue una tarea interesante, más que el hecho de incluir un atributo string en un objeto, el incluirlo dentro de un programa ya hecho aporta a habilidades de mantenimiento e incluso puede ser refactorización de algunas partes del mismo, estoy intentando no solo hacer un programa que funciones, sino intentar seguir metodologías de programación limpias que sean eficientes tanto en su costo computacional como en su mantenimiento y posteriores actualizaciones, y el incluir atributos así en un programa ya diseñado evalúa que tan bien dejamos nuestro código preparado para esto; es por ello que en mis clases como menú, tengo varias funciones útiles y reciclables, como una que da un string con un encabezado, o la lectura de tipos de datos específicos como enteros, string's o en este caso, char's, simplifica mucho el trabajo de entrada de datos para no tener que ponder validaciones y mensajes cada que se pide un dato.

Respecto a los métodos de búsqueda, ambos son intuitivos y fáciles de conceptualizar, una búsqueda tradicional elemento por elemento y otro parecido a un método matemático de bisección, también es atractivo conocer tanto sus costos computacionales (n en el caso de la lineal y log(n) en el caso de la binario con la notación BigO), como las situaciones donde son mejores, y estar preparados para ello es importante; el manejo de datos no solamente es el guardado de información, también, para procesarla, modificarla, eliminarla o demás necesitamos acceder a ella y usualmente no conoceremos la ubicación (o en este caso índice) específico donde se encuentra, son unas de las operaciones básicas que debemos conocer y nos preparan para programas más complejos.