CUCEI

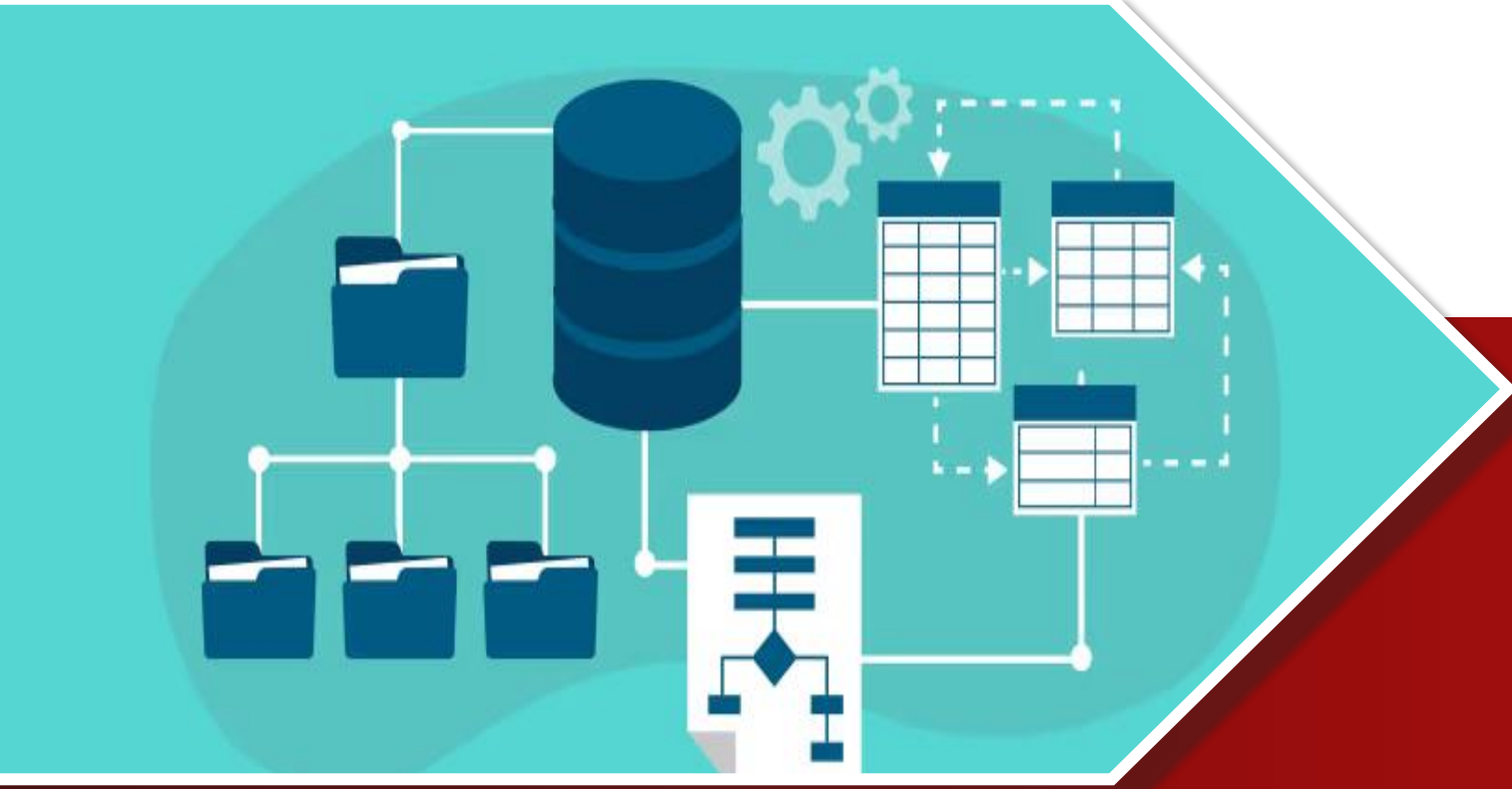
Actividad de Aprendizaje 04

Archivos

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: 06 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

El objetivo de esta actividad fue poner en práctica lo visto en clase con los operadores de flujo y el manejo de archivos a través de implementar mejoras en la actividad pasada, de la lista de éxitos musicales, en este caso, hacer que la lista pudiera guardarse en archivos para mantener su información y posteriormente poder recuperar esos datos para que los objetos recuperaran su estado anterior con ello.

Nuevamente, antes de ponernos a programar, antes viene una fase de planeación para visualizar cómo es que quedarán estas mejoras aplicadas a la estructura de trabajo que ya tenemos; por lo que igual que en los anteriores trabajos, dividimos este primer parte en planeación y programación.

Planeación

Para el manejo de archivos, implementaremos los operadores de flujo en nuestras clases involucradas, iremos de la más simple a la clase contenedor List, de esta manera, iremos vigilando mediante test que haremos que se vayan manejando correctamente; además, por lo visto en clase, el manejo de enteros como tal puede ser muy problemático a la hora de entrada y/o salida de los datos, por lo que manejaremos cadenas de caracteres por medio para evadir lo más posible estos problemas. Para el formato o la convención en la que representaremos los datos en el archivo, en lugar de hacerlo con solo saltos de líneas, lo haremos mediante separación de comas para que sea compatible con un archivo .csv, al poder ser exportado en este formato, podemos hacer su manejo y visualización más flexible fuera del mismo programa, por ejemplo, en herramientas como Excel se podrá ver y actualizar más fácilmente, aunque no descarta que pueda ser exportado en un simple archivo .txt. Siguiendo el ejemplo de otros programas, además de tener pantallas que serán sencillas para la escritura y lectura de un database, cuando se quiera salir del programa y registros realizados, vamos a preguntar si se quiere guardar el registro, en caso de que se haya hecho algún cambio. En cuestión de archivos, será todo lo que implementaremos, pero aprovechando que regresamos al programa, vamos a realizarle algunas mejoras en él.

Los cambios que planeé para el programa fueron 3 más importantes: primero, crearé un método de encabezado, no lo había realizado porque no había tantas pantallas que lo requerían, pero ahora que crece, vamos a hacer una función a la cuál se le pasé el

título de la pantalla, el tamaño de anchura que quiere que se tenga y creará el encabezado, reemplazaremos los códigos similares para que se optimice y reutilizarlo lo más posible.

Otro cambio relevante es que pasaremos de que el menú tenga de composición a una lista a que la tenga por referencia, esto para seguir mejor los lineamientos de POO, ya que un menú "usa" una lista, más no "tiene" una lista, cambiaremos lo que sea necesario para ello.

Por último y uno muy relevante, en su programación inicial, despistadamente, después de que una pantalla finalizará, volvía a llamar a mainMenu() en cada una en lugar de un return, el problema es que cargábamos cada vez la pila de llamadas con ello, un error inocente que corregiré aquí; para ello, planeo lo siguiente: en el programa tengo un método que maneja la elección del usuario del tipo void handleOption(), tiene un bucle do while que no deja de ejecutarse hasta cierta opción; lo que haré será pasar el bucle al mainMenu() como un buble while(), mainMenu() cada vez que se retorné usará el toString() de List para ir actualizando la lista de canciones si hay algún cambio y sí se vean reflejados, imprimirá las opciones y luego pasará a handleOption() para redigirala, esta útlima pasará a ser una función booleana, que siempre regresará true (para que el bucle while siga ejecutándose) a menos que se seleccione la opción de salir. De esta manera, sustituiremos todos las llamadas a mainMenu() por return's para no sobrecargar la pila de funciones y tenemos una implementación bastante más eficaz que además de funcionar, cuida los recursos y el costo computacional del programa.

Como adición a todo esto, también completaremos los objetos con los operadores relacionales que no habían sido incluidos, para las clases como Name o Song, la primera siendo por orden alfabético (usando el toString()) y para la segunda, utilizaremos el ranking como medida para determinar sus comparaciones.

Con todo este plan en mente antes de teclear, pasemos ahora sí a Visual Studio Code para plasmar todas estas ideas.

Programación

Sabiendo ya lo que íbamos a hacer, comencé primeramente por completar los objetos, es decir, añadirle sus operadores relacionales, que fue aquello que les faltó la última actualización de esta actividad y comenzando por el nombre, agregamos los operadores de flujo y verificando en cada paso que estos funcionaran correctamente, siguiendo un patrón de separación por comas para que sea compatible con .csv. Después, seguimos con la clase List, y para maximizar aún más esta compatibilidad con plataformas como Excel, agregué una línea de encabezados a la escritura de archivos, con su correspondiente adaptación para que la lectura tomará eso en cuenta. Funcionando bien aquello, diseñé el método de encabezado que tenía planteado para hacer más mantenible y expandible el programa, y con ello, pasé a diseñar las pantalas de escritura y entrada de datos, considerando que es el usuario quien pone el nombre del database a guardar, en adición a esto, añadí en un método de salir, trasladé el mensaje a este y le puse que si el sistema tenía guardadas canciones (si no estaba vacío) preguntará si se quería guardar en un archivo los registros.

Con ello hecho, pasé a hacer que la clase Menú tuviera por referencia y no por valor la List, y realicé los cambios del mainMenú() y de handleOption() para no apilar funciones en la memoria. De ahí, fue corregir algunos problemas que se iban dando, como errores lógicos y demás, quedando en un programa funcional y más robusto que el anterior y que cumple con los requisitos de la actividad.

Código Fuente

Carpeta include

exception.hpp

```
#ifndef __EXCEPTIONS_H__
#define __EXCEPTIONS_H__

#include <exception>
#include <string>

class Exception : public std::exception {
 private:
  std::string msg;

 public:
  Exception() noexcept : msg("Error Indefinido") {}
  Exception(const Exception& ex) noexcept : msg(ex.msg) {}
  Exception(const std::string& m) : msg(m) {}
  Exception& operator=(const Exception& ex) noexcept {
    msg = ex.msg;
    return *this;
  }

  virtual ~Exception() {}
  virtual const char* what() const noexcept { return msg.c_str(); }
};

class OperationCanceledException : public Exception {
 public:
  OperationCanceledException() noexcept : Exception("Operacion
Cancelada") {}

  OperationCanceledException(const OperationCanceledException& ex)
noexcept
      : Exception(ex) {}

  OperationCanceledException(const std::string& m) : Exception(m) {}
  OperationCanceledException& operator=(
      const OperationCanceledException& ex) noexcept {
    Exception::operator=(ex);  // reutiliza asignación de la base
    return *this;
  }

  virtual ~OperationCanceledException() {}
};

#endif  // __EXCEPTIONS_H__
```

list.hpp

```
#ifndef __LIST_H__
#define __LIST_H__

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

#include "exceptions.hpp"

template <class T, int ARRAYSIZE = 50>
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 isRankingAvalible(const int&) const;
  bool isValidPosition(const int&) const;

  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>&);
};

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 Exception("Lista Llena, InsertElement(List)");

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

  if (!isRankingAvalible(newData.getRanking()))
    throw Exception("Ranking no se puede repetir, 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 Exception("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 Exception("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 {
  int widthRanking = 10, widthSongName = 40, widthName = 35;
  int widthBorder = (widthRanking * 2) + widthSongName + (widthName * 2)
+ 1;
```

```
    std::stringstream oss;
  oss << std::setfill('=') << std::setw(widthBorder) << "" << std::endl;
  oss << std::setfill(' ');

  oss << "|" << std::setw(widthBorder / 2) << "LISTA DE EXITOS"
      << std::setw(widthBorder / 2) << "|" << std::endl;

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

  oss << std::left << std::setw(widthRanking) << "| N Lista";
  oss << std::left << std::setw(widthRanking) << "| Ranking";
  oss << std::left << std::setw(widthSongName) << "| Nombre de la
Cancion";
  oss << std::left << std::setw(widthName) << "| Nombre del Autor";
  oss << std::left << std::setw(widthName) << "| Nombre del Interprete";
  oss << "|" << std::endl;

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

  for (int i = 0; i <= last; i++) {
    oss << "| " << std::left << std::setw(widthRanking - 2) << i << "| "
        << std::left << std::setw(widthRanking - 2) <<
data[i].getRanking()
        << "| " << std::left << std::setw(widthSongName - 2)
        << data[i].getSongName() << "| " << std::left
        << std::setw(widthName - 2) << data[i].getAuthor().toString() <<
"| "
        << std::left << std::setw(widthName - 2)
        << data[i].getInterpreter().toString() << "| " << std::endl;
  }

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

  return oss.str();
}

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

template <class T, int ARRAYSIZE>
bool List<T, ARRAYSIZE>::isRankingAvalible(const int& ranking) const {
  for (int i = 0; i <= last; i++)
    if (data[i].getRanking() == ranking)
```

```
      return false;
  return true;
}

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;

  os << "Ranking, Nombre de la Cancion, Nombre del Autor, Apellido del
Autor, "
        "Nombre del Interprete, Apellido del Interprete\n";

  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;
  std::getline(is, aux);

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

#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;

  int readInteger(std::string, const int&, const int&);
  Name readName(std::string);
  std::string readLinePrompt(const std::string&, bool = false);

  bool handleOption(const std::string&);
  std::string windowHeader(const int&, const std::string&);

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

  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 "exceptions.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;

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

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

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

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

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

  std::string toString() const;

  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;

    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;

  return 0;
}
```

menu.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();
}

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 Exception("Numero Fuera de Rango");
      break;
    } catch (std::invalid_argument) {
      system("CLS");
      cout << "Entrada invalida" << endl;
      cout << "Intente nuevamente" << endl;
      system("PAUSE");
    } catch (Exception msg) {
      system("CLS");
      cout << msg.what() << endl;
      system("PAUSE");
    }
  }

  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 (result == "0")
      throw OperationCanceledException();
    if (!allowEmpty && result.empty()) {
      system("CLS");
      cout << "No puede estar vacio.\nIntentelo nuevamente." << endl;
      system("PAUSE");
      continue;
    }
    return result;
  }
}

string Menu::windowHeader(const int& widthBorder, const string& prompt) {
  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)) << "" << "|" <<
endl;
  oss << setfill('-') << setw(widthBorder) << "" << endl;
  oss << setfill(' ');

  return oss.str();
}

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

  system("CLS");
  std::cout << promt;
  std::getline(std::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";
        system("PAUSE");
        break;
      }
      if (!songList.isValidPosition(index)) {
        system("CLS");
        std::cout << "Posicion de lista invalida\n";
```

```
      system("PAUSE");
      break;
    }
    editSong(index);
    break;

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

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

```
      case 'F':
        saveToDisk();
        break;
      case 'G':
        readFromDisk();
        break;
      case 'H':
        exitProgram();
        return false;

      default:
        system("CLS");
        std::cout << "Comando invalido\nIntentelo nuevamente.\n";
        system("PAUSE");
        break;
    }  // switch

    return true;
}

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

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

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

        oss << songList.toString();
        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"
               "[F] Guardar la Database [G] Leer del Disco   [H] Salir.\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: ";

  while (true) {
    try {
      myString = readLinePrompt(oss.str());
      newSong.setSongName(myString);
      break;
    } catch (Exception msg) {
      system("CLS");
      cout << msg.what() << endl;
      cout << "Vuelva a intentarlo\n";
      system("PAUSE");
    }
  }

  oss << newSong.getSongName() << endl;
  oss << "Ingrese el Ranking de la Cancion: ";

  while (true) {
    try {
      system("CLS");
      myInt = readInteger(oss.str(), 0, 1000);
      if (!songList.isRankingAvalible(myInt))
        throw std::invalid_argument("Ranking ya utilizado");
      newSong.setRanking(myInt);
      break;
    } catch (Exception msg) {
      system("CLS");
      cout << msg.what() << endl;
      system("PAUSE");
    } catch (std::invalid_argument msg) {
      system("CLS");
      cout << msg.what() << endl;
      system("PAUSE");
    }
  }

  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);

    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 (Exception msg) {
          system("CLS");
          cout << msg.what() << endl;
          cout << "Intente Nuevamente." << endl;
          system("PAUSE");
        }
      }
    }

    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->toString();
    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();

    system("PAUSE");
}

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

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

```
 string dataString;
Name newName;

oss << target->toString();
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);
    if (!songList.isRankingAvalible(newRanking)) {
      system("CLS");
      cout << "El ranking ya esta ocupado" << endl;
      system("PAUSE");
      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!";
    system("PAUSE");
    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;
  }
  system("PAUSE");
}
```

```
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";
  system("PAUSE");
}

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

  if (this->songList.isEmpty()) {
    cout << "+-----------------------------------------------------+"
<< endl;
    cout << "+            No hay Canciones Registradas Aun         +"
<< endl;
    cout << "+              No hay datos que Guardar               +"
<< endl;
    cout << "+-----------------------------------------------------+"
<< endl;
    system("PAUSE");
    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();
  system("PAUSE");
}

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();
  system("PAUSE");
}
```

name.cpp

```
#include "name.hpp"

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

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

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

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

void Name::setLast(const std::string& last) {
  if (last.empty())
    throw Exception("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);
}
```

```cpp
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);
}

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()
{}

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

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

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

void Song::setSongName(const std::string& songName) {
  if (songName.empty())
    throw Exception("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;
}

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::toString() const {
  ostringstream oss;
  int widthBorder = 60;

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

  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;

  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);
}

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

  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;

  return is;
}
```

Ejecución del Programa

Para la captura de pantallas, tomaremos en cuenta las nuevas funcionalidades, ya que las demás están plasmadas en la actividad anterior.

Recordando, este era el menú principal:

```
================================================================================================================
|                                                LISTA DE EXITOS                                               |
----------------------------------------------------------------------------------------------------------------
| N Lista | Ranking | Nombre de la Cancion                | Nombre del Autor               | Nombre del Interprete           |
----------------------------------------------------------------------------------------------------------------

================================================================================================================
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] Salir.
Seleccione un Comando:
```

Si intentamos acceder a la opción de guardar la database sin tener canciones registradas aparecerá el siguiente mensaje:

```
+-----------------------------------------------------+
+              No hay Canciones Registradas Aun       +
+                No hay datos que Guardar             +
+-----------------------------------------------------+
Presione una tecla para continuar . . .
```

Así que usemos un dataset de canciones como lo hacíamos en la anterior actividad, agregaremos unas cuantas canciones:

```
================================================================================================================
|                                                LISTA DE EXITOS                                               |
----------------------------------------------------------------------------------------------------------------
| N Lista | Ranking | Nombre de la Cancion                | Nombre del Autor               | Nombre del Interprete           |
----------------------------------------------------------------------------------------------------------------
| 0       | 1       | Etrusto Unico                       | Jose Madero                    | Jose Madero                     |
| 1       | 7       | Rayo de Luz                         | Jose Madero                    | Jose Madero                     |
| 2       | 12      | Los Malaventurados No Lloran        | Jose Madero                    | Panda Band                      |
| 3       | 22      | Soñé                                | Jose Madero                    | Panda Band                      |
| 4       | 5       | Cita en el Quirófano                | Jose Madero                    | Panda Band                      |
| 5       | 17      | Cuando No Es Como Debería Ser       | Jose Madero                    | Panda Band                      |
| 6       | 24      | Labios Rotos                        | Leon Larregui                  | Zoe Band                        |
| 7       | 2       | Vía Láctea                          | Leon Larregui                  | Zoe Band                        |
| 8       | 30      | Paula                               | Leon Larregui                  | Zoe Band                        |
| 9       | 14      | Arrullo de Estrellas                | Leon Larregui                  | Zoe Band                        |
| 10      | 47      | Nada                                | Leon Larregui                  | Zoe Band                        |
| 11      | 28      | La Ley del Monte                    | Vicente Fernandez              | Vicente Fernandez               |
| 12      | 41      | Mujeres Divinas                     | Vicente Fernandez              | Vicente Fernandez               |
| 13      | 3       | Volver Volver                       | Vicente Fernandez              | Vicente Fernandez               |
| 14      | 33      | Hermoso Cariño                      | Vicente Fernandez              | Vicente Fernandez               |
| 15      | 19      | El Rey                              | Vicente Fernandez              | Vicente Fernandez               |
| 16      | 8       | Julieta                             | Pablo Preciado                 | Latin Mafia                     |
| 17      | 21      | No Me Conozco                       | Miloh Smith                    | Latin Mafia                     |
| 18      | 29      | Patadas de Ahogado                  | Miloh Smith                    | Latin Mafia                     |
| 19      | 38      | Morir De Amor                       | Pablo Preciado                 | Latin Mafia                     |
| 20      | 44      | Eres un Ángel                       | Miloh Smith                    | Latin Mafia                     |
| 21      | 6       | Amorfoda                            | Benito Martinez                | Bad Bunny                       |
| 22      | 40      | Ojitos Lindos                       | Benito Martinez                | Bad Bunny                       |
| 23      | 15      | Dákiti                              | Benito Martinez                | Bad Bunny                       |
| 24      | 25      | Callaita                            | Benito Martinez                | Bad Bunny                       |
| 25      | 49      | Tití Me Preguntó                    | Benito Martinez                | Bad Bunny                       |
| 26      | 18      | Eres                                | Leon Larregui                  | Zoe Band                        |
| 27      | 11      | Love                                | Leon Larregui                  | Zoe Band                        |
| 28      | 23      | Déjate Conectar                     | Leon Larregui                  | Zoe Band                        |
| 29      | 34      | Final Feliz                         | Leon Larregui                  | Zoe Band                        |
| 30      | 9       | Azul                                | Leon Larregui                  | Zoe Band                        |
| 31      | 13      | Solo a Terceros                     | Jose Madero                    | Panda Band                      |
| 32      | 35      | Miércoles                           | Jose Madero                    | Panda Band                      |
| 33      | 27      | Disculpa los Malos Pensamientos     | Jose Madero                    | Panda Band                      |
| 34      | 10      | Ya No Jalaba                        | Jose Madero                    | Panda Band                      |
| 35      | 36      | Un Hombre con Suerte                | Vicente Fernandez              | Vicente Fernandez               |
```

Con un dataset así, vamos a ingresar el comando [F] para guardar en un archivo:

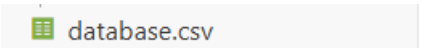
```
=================================================
|                GUARDAR DATABASE                |
-------------------------------------------------
Ingrese el Nombre que Tendra el Archivo:
```

Le daremos el nombre de database.csv:

```
=================================================
|                GUARDAR DATABASE                |
-------------------------------------------------
Ingrese el Nombre que Tendra el Archivo: Database guardada con Exito!
Presione una tecla para continuar . . .
```

Si todo sale bien, se nos avisa que el archivo fue guardado con éxito, y volveremos al menú principal.

Si revisamos la carpeta raíz, tendremos un archivo como el siguiente:



Y con el, tenemos el registro de canciones:

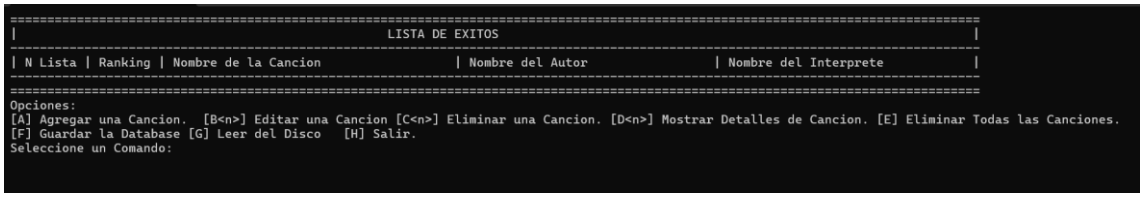
```
Ranking, Nombre de la Cancion, Nombre del Autor, Apellido del Autor, Nombre del Interprete, Apellido del Interprete
1,Etrusto Unico,Jose,Madero,Jose,Madero,
7,Rayo de Luz,Jose,Madero,Jose,Madero,
12,Los Malaventurados No Lloran,Jose,Madero,Panda,Band,
22,So��,Jose,Madero,Panda,Band,
5,Cita en el Quir�fano,Jose,Madero,Panda,Band,
17,Cuando No Es Como Deber�a Ser,Jose,Madero,Panda,Band,
24,Labios Rotos,Leon,Larregui,Zoe,Band,
2,V�a L�ctea,Leon,Larregui,Zoe,Band,
30,Paula,Leon,Larregui,Zoe,Band,
14,Arrullo de Estrellas,Leon,Larregui,Zoe,Band,
47,Nada,Leon,Larregui,Zoe,Band,
28,La Ley del Monte,Vicente,Fernandez,Vicente,Fernandez,
41,Mujeres Divinas,Vicente,Fernandez,Vicente,Fernandez,
3,Volver Volver,Vicente,Fernandez,Vicente,Fernandez,
33,Hermoso Cari�o,Vicente,Fernandez,Vicente,Fernandez,
19,El Rey,Vicente,Fernandez,Vicente,Fernandez,
8,Julieta,Pablo,Preciado,Latin,Mafia,
21,No Me Conozco,Miloh,Smith,Latin,Mafia,
29,Patadas de Ahogado,Miloh,Smith,Latin,Mafia,
38,Morir De Amor,Pablo,Preciado,Latin,Mafia,
44,Eres un �ngel,Miloh,Smith,Latin,Mafia,
6,Amorfoda,Benito,Martinez,Bad,Bunny,
40,Ojitos Lindos,Benito,Martinez,Bad,Bunny,
15,D�kiti,Benito,Martinez,Bad,Bunny,
```

Vemos como se tiene un encabezado y los datos separados con comas, y este archivo es compatible con aplicaciones como Excel:

	A	B	C	D	E	F
1	Ranking	Nombre de la Cancion	Nombre del Autor	Apellido del Autor	Nombre del Interprete	Apellido del Interprete
2	1	Etrusto Unico	Jose	Madero	Jose	Madero
3	7	Rayo de Luz	Jose	Madero	Jose	Madero
4	12	Los Malaventurados No Lloran	Jose	Madero	Panda	Band
5	22	Soπ,	Jose	Madero	Panda	Band
6	5	Cita en el QuirЄfano	Jose	Madero	Panda	Band
7	17	Cuando No Es Como Deberյa Ser	Jose	Madero	Panda	Band
8	24	Labios Rotos	Leon	Larregui	Zoe	Band
9	2	V|a L ctea	Leon	Larregui	Zoe	Band
10	30	Paula	Leon	Larregui	Zoe	Band
11	14	Arrullo de Estrellas	Leon	Larregui	Zoe	Band
12	47	Nada	Leon	Larregui	Zoe	Band
13	28	La Ley del Monte	Vicente	Fernandez	Vicente	Fernandez
14	41	Mujeres Divinas	Vicente	Fernandez	Vicente	Fernandez
15	3	Volver Volver	Vicente	Fernandez	Vicente	Fernandez
16	33	Hermoso Cariπo	Vicente	Fernandez	Vicente	Fernandez
17	19	El Rey	Vicente	Fernandez	Vicente	Fernandez
18	8	Julieta	Pablo	Preciado	Latin	Mafia
19	21	No Me Conozco	Miloh	Smith	Latin	Mafia
20	29	Patadas de Ahogado	Miloh	Smith	Latin	Mafia
21	38	Morir De Amor	Pablo	Preciado	Latin	Mafia
22	44	Eres un µngel	Miloh	Smith	Latin	Mafia
23	6	Amorfoda	Benito	Martinez	Bad	Bunny
24	40	Ojitos Lindos	Benito	Martinez	Bad	Bunny
25	15	D kiti	Benito	Martinez	Bad	Bunny

Esto hace mucho más fácil su visualización y edición fuera del propio programa, haciéndolo también bastante portable.

Después de esto, regresamos al menú principal. Para la siguiente funcionalidad de lectura, vamos a eliminar todas las canciones con el comando [E]:



Y ahora, usemos el comando para leer [G]:

si utilizamos un nombre de un archivo inexistente:

Ahora, pongamos el nombre database.csv que fue con el que realmente se guardó el archivo:

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

Y regresando al menú principal:

```
================================================================================================================
|                                                LISTA DE EXITOS                                               |
----------------------------------------------------------------------------------------------------------------
| N Lista | Ranking | Nombre de la Cancion                | Nombre del Autor          | Nombre del Interprete     |
----------------------------------------------------------------------------------------------------------------
| 0       | 1       | Etrusto Unico                       | Jose Madero               | Jose Madero               |
| 1       | 7       | Rayo de Luz                         | Jose Madero               | Jose Madero               |
| 2       | 12      | Los Malaventurados No Lloran        | Jose Madero               | Panda Band                |
| 3       | 22      | Soñé                                | Jose Madero               | Panda Band                |
| 4       | 5       | Cita en el Quirófano                | Jose Madero               | Panda Band                |
| 5       | 17      | Cuando No Es Como Debería Ser       | Jose Madero               | Panda Band                |
| 6       | 24      | Labios Rotos                        | Leon Larregui             | Zoe Band                  |
| 7       | 2       | Vía Láctea                          | Leon Larregui             | Zoe Band                  |
| 8       | 30      | Paula                               | Leon Larregui             | Zoe Band                  |
| 9       | 14      | Arrullo de Estrellas                | Leon Larregui             | Zoe Band                  |
| 10      | 47      | Nada                                | Leon Larregui             | Zoe Band                  |
| 11      | 28      | La Ley del Monte                    | Vicente Fernandez         | Vicente Fernandez         |
| 12      | 41      | Mujeres Divinas                     | Vicente Fernandez         | Vicente Fernandez         |
| 13      | 3       | Volver Volver                       | Vicente Fernandez         | Vicente Fernandez         |
| 14      | 33      | Hermoso Cariño                      | Vicente Fernandez         | Vicente Fernandez         |
| 15      | 19      | El Rey                              | Vicente Fernandez         | Vicente Fernandez         |
| 16      | 8       | Julieta                             | Pablo Preciado            | Latin Mafia               |
| 17      | 21      | No Me Conozco                       | Miloh Smith               | Latin Mafia               |
| 18      | 29      | Patadas de Ahogado                  | Miloh Smith               | Latin Mafia               |
| 19      | 38      | Morir De Amor                       | Pablo Preciado            | Latin Mafia               |
| 20      | 44      | Eres un Ángel                       | Miloh Smith               | Latin Mafia               |
| 21      | 6       | Amorfoda                            | Benito Martinez           | Bad Bunny                 |
| 22      | 40      | Ojitos Lindos                       | Benito Martinez           | Bad Bunny                 |
| 23      | 15      | Dákiti                              | Benito Martinez           | Bad Bunny                 |
| 24      | 25      | Callaita                            | Benito Martinez           | Bad Bunny                 |
| 25      | 49      | Tití Me Preguntó                    | Benito Martinez           | Bad Bunny                 |
| 26      | 18      | Eres                                | Leon Larregui             | Zoe Band                  |
| 27      | 11      | Love                                | Leon Larregui             | Zoe Band                  |
| 28      | 23      | Déjate Conectar                     | Leon Larregui             | Zoe Band                  |
| 29      | 34      | Final Feliz                         | Leon Larregui             | Zoe Band                  |
| 30      | 9       | Azul                                | Leon Larregui             | Zoe Band                  |
| 31      | 13      | Solo a Terceros                     | Jose Madero               | Panda Band                |
| 32      | 35      | Miércoles                           | Jose Madero               | Panda Band                |
| 33      | 27      | Disculpa los Malos Pensamientos     | Jose Madero               | Panda Band                |
| 34      | 10      | Ya No Jalaba                        | Jose Madero               | Panda Band                |
| 35      | 36      | Un Hombre con Suerte                | Vicente Fernandez         | Vicente Fernandez         |
```

Así, aunque cerremos el programa, nuestros objetos pueden recuperar su estado leyendo estos archivos; como otra funcionalidad, si salimos del programa con registros, antes de que se cierre aparece esto:

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

Si le damos que sí nos regresa a la pantalla de escribir en el disco antes de salir:

```
==================================================
|                GUARDAR DATABASE                |
--------------------------------------------------
Ingrese el Nombre que Tendra el Archivo:
```

Y si salemos, finalmente:

```
Saliendo del Programa.
Tenga un Lindo Dia :D
Presione una tecla para continuar . . .
```

Y nuestro programa finaliza.

Conclusiones

El manejo de archivos es una herramienta tan útil como interesante, aprendí bastante tanto en la clase como en esta práctica, mucha parte de la lógica de cómo funciona el programa y realmente qué representan los símbolos << y >>, más allá de utilizarlos para los cout's y cin's del programa; reconocerlos como operadores de flujo de los cuales podemos definirlos para los objetos que planteamos da mucha libertad para manejar la memoria post cierre de programa. Realizarla en formato .csv para que sea compatible con otras herramientas me hace imaginar cómo podríamos hacerlo más compatible con otras herramientas de gestión de información, nosotros como programadores tenemos ese control y estándares.

Además, adquirí cierta noción de por qué los enteros y flotantes pueden causar problemas si se manejan sin cuidado ocasionando errores inesperados y más difíciles de seguir y depurar; por ello, es mejor capturarlas como cadenas de caracteres y después convertirlas al tipo deseado. El manejo de archivos es conocimiento que tarde o temprano, un programador y un informático debe de aprender; no solo procesamos información, sino que hay gran utilidad explícita en preservar dicha información en formatos diversos.

Con esto, ahora puedo completar mucho más mis programas POO, haciendo que un objeto tenga la estructura básica e indispensable, desde sus atributos, constructores, etc., hasta los operadores de flujo. Será una gran herramienta para mis futuros proyectos y para actualizar los ya existentes.