

Codigo: 208623082

Carrera: INNI

Seccion: D05



CUCEI ESTRUCTURA DE DATOS I ACTIVIDAD DE  
APRENDIZAJE VI

## Objetivo:

Haga un programa que realice el ordenamiento de la lista, opcionalmente por nombre de la canción como por nombre del cantante. Para evaluar su funcionamiento la radiodifusora quiere probar con distintos tipos de ordenamiento.

## Resolución:

Para este programa se uso C++. Las implementaciones de los 4 tipos de ordenamientos (burbuja, Shell, inserción y selección) se pusieron en el archivo de la lista. Además de los comparadores booleanos de las canciones.

## Código fuente:

### Lista.h

```
#ifndef LISTA_H_INCLUDED
#define LISTA_H_INCLUDED

#include <cstdlib>
#include <string>
#include <iostream>
#include "cancion.h"
#include "sortbyAuthor.h"
#include "listexception.h"

class List {

private:
    Cancion data[50];
    int last;

    bool isValidPos(const int&);

    void copyAll(const List&);
    void swapData(Cancion&, Cancion&);

public:

    List(); //inicializa
    List(const List&);
```

```

bool isEmpty(); //vacio
bool isFull(); //lleno

void InsertData(const int&, const Cancion&); //inserta

void deleteData(const int&); //elimina

int getFirstpos(); //primero
int getLastpos(); //ultimo
int getPrevpos(const int&); //anterior
int getNextpos(const int&); //siguiente

int findDataLinear(const Cancion&);
int findDataBinary(const Cancion&);

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

void sortDataBubbleAuthor();
void sortDataShellAuthor();
void sortDataInsertAuthor();
void sortDataSelectAuthor();

Cancion retrieve(const int&); //recupera

std::string toString();

void print();
void deleteAll(); //anula

List& operator = (const List&);

};

#endif // LISTA_H_INCLUDED

```

## Lista.cpp

```
#include "lista.h"

using namespace std;

void List::copyAll(const List& l){
    int i=0;
    while(i<=l.last){
        data[i]=l.data[i];
        i++;
    }

    last=l.last;
}

bool List::isValidPos(const int& p){
    return p>=0 and p<=last;
}

List::List() : last(-1) {}

List::List(const List& l){
    copyAll(l);
}

void List::swapData(Cancion& a, Cancion& b){
    Cancion aux=a;
    a=b;
    b=aux;
}

bool List::isEmpty(){
    return last==-1;
}

bool List::isFull(){
    return last==499;
}

void List::InsertData(const int& p, const Cancion& e){

    if(isFull()){
        throw ListException("Desbordamiento de datos, inserteDatos");
    }
}
```

```

if(p != -1 and !isValidPos(p)) {
    throw ListException("Posicion invalida, insertData");
}

int i(last);
while(i>p){
    data[i+1]=data[i];

    i--;
}
data[p+1]= e;
last++;

}

void List::deleteData(const int& p){
    if(!isValidPos(p)){
        throw ListException("Posicion invalida, deleteData");
        return;
    }

    int i(p);
    while(i<last){
        data[i]=data[i+1];
        i++;
    }
    last--;
}

int List::getFirstpos(){
    if(isEmpty()){
        return -1;
    }
    return 0;
}

int List::getLastpos(){
    return last;
}

int List::getPrevpos(const int& p){
    if(p==getFirstpos() or !isValidPos(p)){
        return -1;
    }
    return p-1;
}

```

```

int List::getNextpos(const int& p){
if(p==getLastpos() or !isValidPos(p)){
    return -1;
}
return p+1;
}

```

```

int List::findDataLinear(const Cancion& e){
    int i=0;
    while(i<=last){
        if(data[i]==e){
            return i;
        }
        i++;
    }
    return -1;
}

```

```

int List::findDataBinary(const Cancion& e){

    int i=0,j=last, m;

    while(i<=j){
        m=(i+j)/2;
        if(data[m]==e){
            return m;
        }
        if(e<data[m]){
            j=m-1;
        }
        else{
            i=m+1;
        }
    }
    return -1;
}

```

```

void List::sortDataBubble(){
int i=last;
bool flag;

do{
    flag=false;
    int j=0;

```

```

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

```

```

void List::sortDataShell(){
    float factor(1.0/2.0);
    int dif((last+1)*factor),i,j;

    while(dif>0){
        i=dif;

        while(i<=last){
            j=i;
            while(j>=dif and data[j-dif]>data[j]){
                swapData(data[j-dif],data[j]);
                j-=dif;
            }
            i++;
        }
        dif*=factor;
    }
}

```

```

void List::sortDataInsert(){

int i=1,j;
Cancion aux;

while(i<=last){
    aux=data[i];
    j=i;

    while(j>0 and aux<data[j-1]){
        data[j]=data[j-1];
        j--;
    }
    if(i !=j){

```

```

        data[j]=aux;
    }
    i++;

}
}
void List::sortDataSelect(){

```

```

    int i=0,j,m;

    while(j< last){
        m=i;
        j=i+1;

        while(j<=last){
            if(data[j]<data[m]){
                m=j;
            }
            j++;
        }
        if(i != m){
            swapData(data[i], data[m]);
        }
        i++;
    }
}

```

```

void List::sortDataBubbleAuthor(){

    int i=last;
    bool flag;

    do{
        flag=false;
        int j=0;

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

```



```
}while(flag);
```

```
}
```

```
void List::sortDataShellAuthor(){
```

```
    float factor(1.0/2.0);
```

```
    int dif((last+1)*factor),i,j;
```

```
    while(dif>0){
```

```
        i=dif;
```

```
        while(i<=last){
```

```
            j=i;
```

```
            while(j>=dif and data[j-dif].getnombreAutor(>)data[j].getnombreAutor()){
```

```
                swapData(data[j-dif],data[j]);
```

```
                j-=dif;
```

```
            }
```

```
            i++;
```

```
        }
```

```
        dif*=factor;
```

```
    }
```

```
}
```

```
void List::sortDataInsertAuthor(){
```

```
    int i=1,j;
```

```
    Cancion aux;
```

```
    while(i<=last){
```

```
        aux=data[i];
```

```
        j=i;
```

```
        while(j>0 and aux<data[j-1]){
```

```
            data[j].getnombreAutor()=data[j-1].getnombreAutor();
```

```
            j--;
```

```
        }
```

```
        if(i !=j){
```

```
            data[j]=aux;
```

```
        }
```

```
        i++;
```

```
}
```

```
}
```

```
void List::sortDataSelectAuthor(){
```

```
    int i=0,j,m;
```

```
    while(j< last){
```

```
        m=i;
```

```
        j=i+1;
```

```
        while(j<=last){
```

```
            if(data[j].getnombreAutor()<data[m].getnombreAutor()){
```

```
                m=j;
```

```
            }
```

```
            j++;
```

```
        }
```

```
        if(i != m){
```

```
            swapData(data[i], data[m]);
```

```
        }
```

```
        i++;
```

```
    }
```

```
}
```

```
Cancion List::retrieve(const int& p){
```

```
    if(!isValidPos(p)){
```

```
        throw ListException("Posicion invalida, retrieve");
```

```
    }
```

```
    return data[p];
```

```
}
```

```
string List::toString(){
```

```
    string result;
```

```
    int i(0);
```

```
    while(i<=last){
```

```
        result+= data[i].toString()+ "\n";
```

```
        i++;
```

```
    }
```

```
    result+="\n";
```

```
    return result;
```

```
}
```

```
void List::print(){
    int i(0);

    while(i <=last){
        cout << data[i++] << ", ";
    }
    cout<<endl;

}

void List::deleteAll(){
    last= -1;
}

List& List::operator=(const List& l){
    copyAll(l);

    return *this;
}
```

## Canción.h

```
#ifndef CANCION_H_INCLUDED
#define CANCION_H_INCLUDED

#include <iostream>
#include <string>
#include <cstring>

using namespace std;

class Cancion {

private:
    std::string nombreCancion;
    std::string nombreAutor;

    int rankingCancion;

    friend class Author;

public:

    Cancion();
    Cancion(const Cancion&);

    std::string getnombreCancion() const;

    std::string getnombreAutor() const;

    int getrankingCancion() const;


    std::string toString() const;


    void setnombreCancion(const std::string&);
    void setnombreAutor(const std::string&);


    void setrankingCancion(const int&);

    Cancion& operator =(const Cancion&);
```

```
bool operator == (const Cancion&) const;
bool operator != (const Cancion&) const;
bool operator < (const Cancion&) const;
bool operator <= (const Cancion&) const;
bool operator > (const Cancion&) const;
bool operator >= (const Cancion&) const;
```

```
bool operator == (string&);
bool operator <(Cancion&);
bool operator <=(Cancion&);
bool operator > (Cancion&);
bool operator >= (Cancion&);
```

```
friend std::ostream& operator << (std::ostream&,Cancion&);
friend std::istream& operator >> (std::istream&,Cancion&);
```

```
};
```

```
#endif // CANCION_H_INCLUDED
```

## **Canción.cpp**

```
#include "cancion.h"
#include <string>
#include <iostream>
#include <cstring>
```

```
using namespace std;
```

```
Cancion::Cancion(){}
Cancion::Cancion(const Cancion&
p):nombreCancion(p.nombreCancion),nombreAutor(p.nombreCancion),
rankingCancion(p.rankingCancion) {}
```

```
Cancion& Cancion::operator=(const Cancion& p){
```

```

nombreCancion=p.nombreCancion;
nombreAutor=p.nombreAutor;

rankingCancion=p.rankingCancion;

return *this;
}

string Cancion::getnombreCancion() const{
return nombreCancion;
}

string Cancion::getnombreAutor() const{
return nombreAutor;
}

int Cancion::getrankingCancion() const{
return rankingCancion;
}

string Cancion::toString() const {
    string result;

    result=nombreCancion;
    result+=" | ";
    result+=nombreAutor;
    result+=" | ";
    result+=to_string(rankingCancion);

    return result;
}

void Cancion::setnombreCancion(const string& c){
nombreCancion=c;
}

void Cancion::setnombreAutor(const string& a){
nombreAutor=a;
}

void Cancion::setrankingCancion(const int& e){
rankingCancion=e;
}

```

```
bool Cancion::operator==(string& c){  
    return nombreCancion==c;  
  
}
```

```
bool Cancion::operator<(Cancion& c){  
    return nombreCancion<c.nombreCancion;  
  
}
```

```
bool Cancion::operator<=(Cancion& c){  
    return nombreCancion<=c.nombreCancion;  
  
}
```

```
bool Cancion::operator>(Cancion& c){  
    return nombreCancion>c.nombreCancion;  
}
```

```
bool Cancion::operator>=(Cancion& c){  
    return nombreCancion>=c.nombreCancion;  
}
```

```
bool Cancion::operator ==(const Cancion& p) const{  
    return nombreCancion == p.nombreCancion;  
}
```

```
bool Cancion::operator!=(const Cancion& p) const{  
    return nombreCancion != p.nombreCancion;  
}
```

```
bool Cancion::operator >(const Cancion& p) const{  
    return nombreCancion > p.nombreCancion;  
}
```

```
bool Cancion::operator >=(const Cancion& p) const{  
    return nombreCancion >= p.nombreCancion;  
}
```

```
bool Cancion::operator<(const Cancion& p) const{  
    return nombreCancion < p.nombreCancion;  
}
```

```
bool Cancion::operator<=(const Cancion& p) const{  
    return nombreCancion <= p.nombreCancion;  
}
```

```

ostream& operator << (ostream& os, Cancion& p){
    os<<p.nombreCancion<<endl;
    os<<p.nombreAutor<<endl;
    os<<p.rankingCancion<<endl;

    return os;
}

istream& operator >> (istream& is, Cancion& p){
    string myStr;

    getline(is, p.nombreCancion);
    getline(is, p.nombreAutor);
    p.rankingCancion=stoi(myStr);

    return is;
}

```

## Menú.h

```

#ifndef MENU_H_INCLUDED
#define MENU_H_INCLUDED
#include <iostream>

using namespace std;

class Menu {

public:
    void menu();
    void menuOrdenaCancion();
    void menuOrdenaAutor();

};

void Menu::menu(){

```



```

        cout << "1.- Llenar canciones " << endl;
        cout << "2.- Mostrar lista canciones " << endl;
        cout << "3.- Actualizar cancion " << endl;
        cout << "4.- Eliminar cancion " << endl;
        cout << "5.- Busqueda lineal " << endl;
        cout << "6.- Busqueda binaria " << endl;
        cout << "7.- Ordenar por nombre de cancion " << endl;
        cout << "8.- Ordenar por nombre de autor " << endl;
        cout << "9.- Salir " << endl;

    }

    void Menu::menuOrdenaAutor(){

        cout << "1.- Ordenar por burbuja (mejorada) " << endl;
        cout << "2.- Ordenar por shell " << endl;
        cout << "3.- Ordenar por inserción " << endl;
        cout << "4.- Ordenar por seleccion " << endl;
        cout << "5.- Salir " << endl;

    }

    void Menu::menuOrdenaCancion(){

        cout << "1.- Ordenar por burbuja (mejorada) " << endl;
        cout << "2.- Ordenar por shell " << endl;
        cout << "3.- Ordenar por inserción " << endl;
        cout << "4.- Ordenar por seleccion" << endl;
        cout << "5.- Salir " << endl;

    }

#endif // MENU_H_INCLUDED

```

## Main.cpp

```
#include <iostream>
```

```

#include <random>
#include <chrono>
#include <functional>
#include "menu.h"

#include "lista.h"

using namespace std;

int main()
{
    List myList;
    Menu mymenu;

    Cancion mySong[50];
    string myStr;
    int opcMenu,i,numeroCanciones,rankingCancion,cancionEliminar,pos;
    string codigoBuscado;

    do {
        mymenu.menu();
        cin>>opcMenu;

        switch(opcMenu){
            case 1:
                cout<<"Cuantas canciones quieres llenar ? ";
                cin>>numeroCanciones;getchar();
                for(i=0;i<numeroCanciones;i++) {

                    cout << "Nombre de cancion con el ranking no. "<<i+1<<" ";
                    getline(cin, myStr);
                    mySong[i].setnombreCancion(myStr);

                    cout << "Nombre de cantante: ";
                    getline(cin, myStr);
                    mySong[i].setnombreAutor(myStr);

                    mySong[i].setrankingCancion(i+1);

                    myList.InsertData(myList.getLastpos(),mySong[i]);
                }

```

```

        cout << "Contenido de la lista... " << endl<<endl;
        cout << myLista.toString();
    break;
    case 2: cout << "Contenido de la lista... " << endl<<endl;
        cout << myLista.toString();
    break;
    case 3:
        cout<<"Dame el ranking a actualizar " <<endl;
        cin>> rankingCancion;
        for(i=0;i<rankingCancion;i++){

            if(rankingCancion==mySong[i].getrankingCancion()){
                cout<<" Actualizando existencia de " << mySong[i].getrankingCancion()
<<endl<<endl;
                cout<< " Cual es su ranking actual ";
                getchar();getline(cin,myStr);
                mySong[i].setrankingCancion(stoi(myStr));
                myLista.InsertData(myLista.getLastpos(),mySong[i]);
                myLista.deleteData(rankingCancion-1);

                cout<<"presione entrar para salir... " << endl;
                getchar();

            }
            else{
                getchar();
                cout<<"El producto no existe, presione entrar para salir... " << endl;
                getchar();

            }
        }
    }

    break;
    case 4:
        cout<<"Que ranking de cancion quieres eliminar ";
        cin>>cancionEliminar;

        if(cancionEliminar==mySong[i].getrankingCancion()){

            cout<<"No se ha encontrado la cancion"<< endl;
        }
        else{

            myLista.deleteData(cancionEliminar-1);
            cout<<"Cancion eliminada ...."<< endl;
            getchar();
            cout<<" presione entrar para salir... " << endl;

```

```

        getchar();
    }

    break;
case 5: cout<< "Nombre de cancion a buscar ";
        getchar();getline(cin,myStr);

        mySong[i].setnombreCancion(myStr);

        pos=myLista.findDatalinear(mySong[i]);
        if(pos== -1){

            cout<<"no se encuentra en la lista"<<endl;

        }
        else {

            cout<<"Se encuentra en la posicion "<< pos +1 <<endl;
            cout<<myLista.retrieve(myLista.findDatalinear(mySong[i])).toString()<<endl;

        }

        break;
case 6: cout<< "Nombre de cancion a buscar ";
        getchar();getline(cin,myStr);

        mySong[i].setnombreCancion(myStr);

        pos=myLista.findDatabinary(mySong[i]);
        if(pos== -1){

            cout<<"no se encuentra en la lista"<<endl;

        }
        else {

            cout<<"Se encuentra en la posicion "<< pos +1 <<endl;
            cout<<myLista.retrieve(myLista.findDatabinary(mySong[i])).toString()<<endl;

        }

        break;
case 7: mymenu.menuOrdenaCancion();
        cin>>opcMenu;
        switch(opcMenu){
            case 1: cout<<"ordenando por burbuja..."<<endl<<endl;
                    myLista.sortDataBubble();

```

```

        cout <<myLista.toString();
    break;
    case 2: cout<<"ordenando por shell..."<<endl<<endl;
        myLista.sortDataShell();
        cout <<myLista.toString();
    break;
    case 3: cout<< "Ordenando por insercion..."<<endl<<endl;
        myLista.sortDataInsert();
        cout <<myLista.toString();
    break;
    case 4: cout<<"Ordenando por seleccion..."<<endl<<endl;
        myLista.sortDataSelect();
        cout <<myLista.toString();
    break;
    default: cout<<"Tenias que elegir entre 1 y 4";
}

```

```

break;
case 8: mymenu.menuOrdenaAutor();
    cin>>opcMenu;
    switch(opcMenu){
    case 1: cout<<"ordenando por burbuja..."<<endl<<endl;
        myLista.sortDataBubbleAuthor();
        cout <<myLista.toString();
    break;
    case 2: cout<<"ordenando por shell..."<<endl<<endl;
        myLista.sortDataShellAuthor();
        cout <<myLista.toString();
    break;
    case 3: cout<<"Ordenando por insercion..."<<endl<<endl;
        myLista.sortDataInsertAuthor();
        cout <<myLista.toString();
    break;
    case 4: cout<<"Ordenando por seleccion..."<<endl<<endl;
        myLista.sortDataSelectAuthor();
        cout <<myLista.toString();
    break;
    default: cout<<"Tenias que elegir entre 1 y 4";
}

```

```

break;
case 9: opcMenu=9;
break;
default: cout<< " Elige una opcion valida "<<endl;
    break;

```

```

    }

}

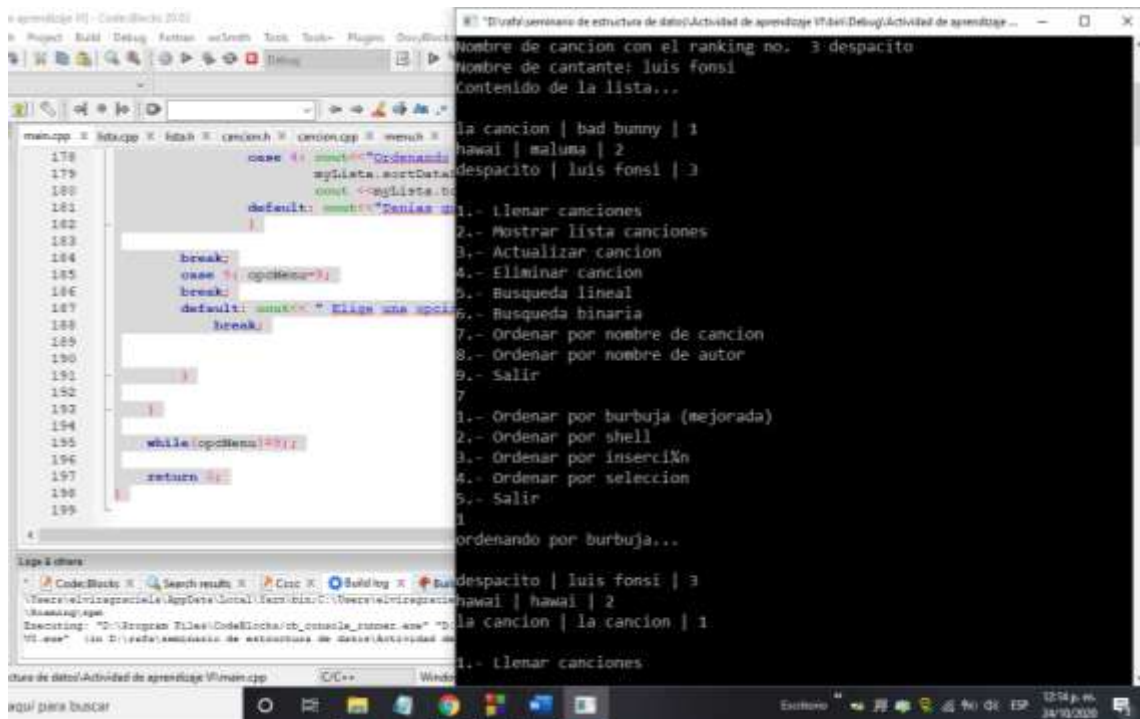
while(opcMenu!=9);

return 0;
}

```

## Capturas de pantalla:

### Busquedas por nombre de cancion



```

C:\Users\seminario de estructura de datos\Actividad de aprendizaje V\bin\Debug\Actividad de aprendizaje V.exe
la cancion | bad bunny | 1
hawai | maluma | 2
despacito | luis fonsi | 3

1.- Llenar canciones
2.- Mostrar lista canciones
3.- Actualizar cancion
4.- Eliminar cancion
5.- Busqueda lineal
6.- Busqueda binaria
7.- Ordenar por nombre de cancion
8.- Ordenar por nombre de autor
9.- Salir
7
1.- Ordenar por burbuja (mejorada)
2.- Ordenar por shell
3.- Ordenar por inserci n
4.- Ordenar por seleccion
5.- Salir
2
ordenando por shell...

despacito | luis fonsi | 3
hawai | hawai | 2
la cancion | la cancion | 1

```

```

C:\Users\seminario de estructura de datos\Actividad de aprendizaje V\bin\Debug\Actividad de aprendizaje V.exe
Nombre de cantante: maluma
Nombre de cancion con el ranking no. 3 despacito
Nombre de cantante: luis fonsi
Contenido de la lista...

la cancion | bad bunny | 1
hawai | maluma | 2
despacito | luis fonsi | 3

1.- Llenar canciones
2.- Mostrar lista canciones
3.- Actualizar cancion
4.- Eliminar cancion
5.- Busqueda lineal
6.- Busqueda binaria
7.- Ordenar por nombre de cancion
8.- Ordenar por nombre de autor
9.- Salir
7
1.- Ordenar por burbuja (mejorada)
2.- Ordenar por shell
3.- Ordenar por inserci n
4.- Ordenar por seleccion
5.- Salir
3
Ordenando por insercion...

despacito | luis fonsi | 3
hawai | maluma | 2
la cancion | bad bunny | 1

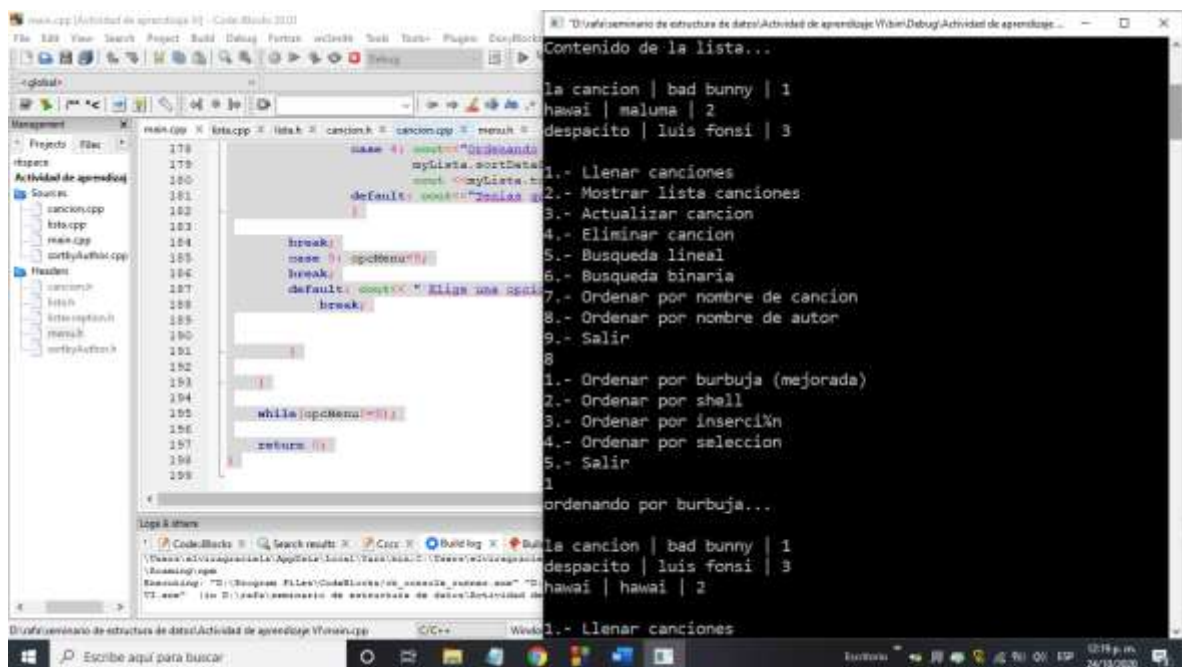
```

```
la cancion | bad bunny | 1
hawai | maluma | 2
despacito | luis fonsi | 3

1.- Llenar canciones
2.- Mostrar lista canciones
3.- Actualizar cancion
4.- Eliminar cancion
5.- Busqueda lineal
6.- Busqueda binaria
7.- Ordenar por nombre de cancion
8.- Ordenar por nombre de autor
9.- Salir
7
1.- Ordenar por burbuja (mejorada)
2.- Ordenar por shell
3.- Ordenar por inserci n
4.- Ordenar por seleccion
5.- Salir
4
Ordenando por seleccion...

despacito | luis fonsi | 3
hawai | maluma | 2
la cancion | la cancion | 1
```

## Ordenamiento por cantante



```
Contenido de la lista...
la cancion | bad bunny | 1
hawai | maluma | 2
despacito | luis fonsi | 3

1.- Llenar canciones
2.- Mostrar lista canciones
3.- Actualizar cancion
4.- Eliminar cancion
5.- Busqueda lineal
6.- Busqueda binaria
7.- Ordenar por nombre de cancion
8.- Ordenar por nombre de autor
9.- Salir
8
1.- Ordenar por burbuja (mejorada)
2.- Ordenar por shell
3.- Ordenar por inserci n
4.- Ordenar por seleccion
5.- Salir
1
Ordenando por burbuja...

la cancion | bad bunny | 1
despacito | luis fonsi | 3
hawai | hawai | 2

1.- Llenar canciones
```



```
8) "D:\seminario de estructuras de datos\Actividad de aprendizaje V\bin\Debug\Actividad de aprendizaje V.exe"
Nombre de cantante: luis fonsi
Contenido de la lista...

lacancion | bad bunny | 1
hawai | maluma | 2
despacito | luis fonsi | 3

1.- Llenar canciones
2.- Mostrar lista canciones
3.- Actualizar cancion
4.- Eliminar cancion
5.- Busqueda lineal
6.- Busqueda binaria
7.- Ordenar por nombre de cancion
8.- Ordenar por nombre de autor
9.- Salir
8
1.- Ordenar por burbuja (mejorada)
2.- Ordenar por shell
3.- Ordenar por insercion
4.- Ordenar por seleccion
5.- Salir
2
ordenando por shell...

lacancion | bad bunny | 1
despacito | luis fonsi | 3
hawai | hawai | 2

1.- Llenar canciones
2
```

```
8) "D:\seminario de estructuras de datos\Actividad de aprendizaje V\bin\Debug\Actividad de aprendizaje V.exe"

la cancion | bad bunny | 1
hawai | maluma | 2
despacito | luis fonsi | 3

1.- Llenar canciones
2.- Mostrar lista canciones
3.- Actualizar cancion
4.- Eliminar cancion
5.- Busqueda lineal
6.- Busqueda binaria
7.- Ordenar por nombre de cancion
8.- Ordenar por nombre de autor
9.- Salir
4
1.- Ordenar por burbuja (mejorada)
2.- Ordenar por shell
3.- Ordenar por insercion
4.- Ordenar por seleccion
5.- Salir
3
ordenando por insercion...

despacito | luis fonsi | 3
hawai | maluma | 2
despacito | luis fonsi | 3

1.- Llenar canciones
2.- Mostrar lista canciones
3.- Actualizar cancion
4.- Eliminar cancion
5.- Busqueda lineal
6.- Busqueda binaria
7.- Ordenar por nombre de cancion
8.- Ordenar por nombre de autor
9.- Salir
```

Contenido de la lista...

la cancion | bad bunny | 1  
hawai | maluma | 2  
despacito | luis fonsi | 3

- 1.- Llenar canciones
- 2.- Mostrar lista canciones
- 3.- Actualizar cancion
- 4.- Eliminar cancion
- 5.- Búsqueda lineal
- 6.- Búsqueda binaria
- 7.- Ordenar por nombre de cancion
- 8.- Ordenar por nombre de autor
- 9.- Salir

8

- 1.- Ordenar por burbuja (mejorada)
- 2.- Ordenar por shell
- 3.- Ordenar por inserción
- 4.- Ordenar por seleccion
- 5.- Salir

4

Ordenando por seleccion...

la cancion | bad bunny | 1  
despacito | luis fonsi | 3  
hawai | hawai | 2

- 1.- Llenar canciones