

Instituto Politécnico Nacional

Escuela Superior de Computo.

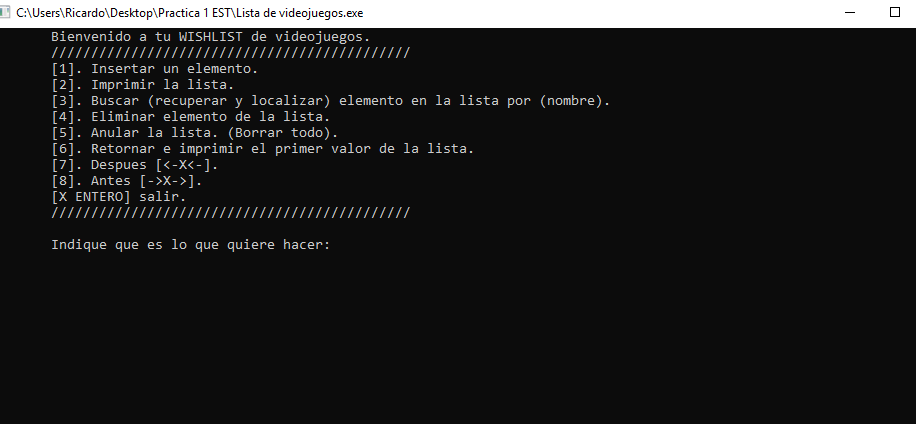
**Practica 1.**

Unidad de aprendizaje: Estructura de datos.

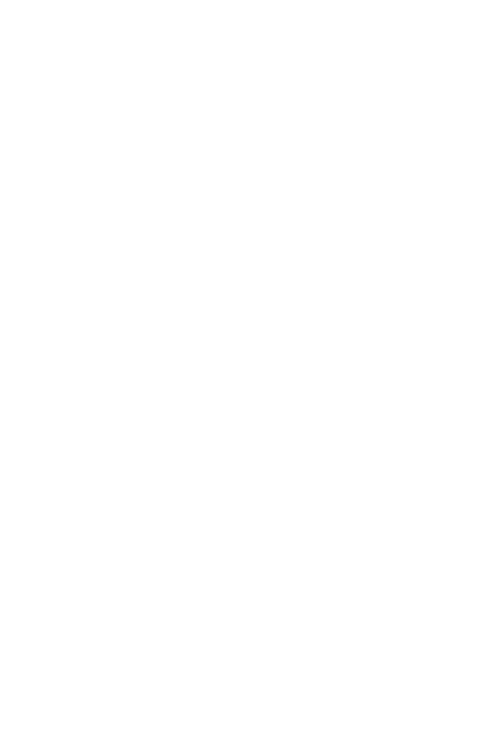
Profesora: José Sánchez Juárez

Alumno:

* Chavarría Vázquez Luis Enrique.



1CM5

Instituto Politécnico Nacional

Escuela Superior de Computo.

Estructura de datos.

Practica 1.

Listas Enlazadas.

Grupo: 1CM5.

Calificación: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Objetivo

Construir las listas enlazadas utilizando apuntadores, y aplicarlas en la solución de problemas reales.

2. Teoría

Las ventajas de usar listas enlazadas es que no se requiere memoria contigua y no es necesario hacer desplazamientos debido a inserciones y relleno de vacios creados por la eliminación de elementos.

Indice.

[Resultados de la practica. 5](#_Toc1328254)

[Codigo. 5](#_Toc1328255)

[Mostrando cada una de las operaciones que realiza el programa. 9](#_Toc1328256)

[Insertar Elemento. 9](#_Toc1328257)

[Imprimir elementos. 10](#_Toc1328258)

[Localizar y recuperar elemento. 10](#_Toc1328259)

[Eliminar un elemento. 11](#_Toc1328260)

[Buscar el primer elemento de la lista. 11](#_Toc1328261)

[Operación de antes y despues. 11](#_Toc1328262)

[Eliminar todos los elementos de la lista (Anular la lista). 11](#_Toc1328263)

[Extras. 12](#_Toc1328264)

[Cuestionario. 12](#_Toc1328265)

[Codigo en c++. (Incluye menu, creación de la lista y prototipos de funciones) 12](#_Toc1328266)

[Codigo en pascal. 15](#_Toc1328267)

[2. Escriba el pseudoc´odigo y el c´odigo de la operaci´on de ISERTAR(x, ↑p, ↑NL), presentando un ejemplo de la lista propuesta. 15](#_Toc1328268)

[Codigo en C++. 15](#_Toc1328269)

[Codigo en pascal. 15](#_Toc1328270)

[3. Escriba el pseudoc´odigo y el c´odigo de la operaci´on de LOCALIZA(x, ↑NL), presentando un ejemplo de la lista propuesta. 16](#_Toc1328271)

[Codigo en C++. 16](#_Toc1328272)

[Codigo en Pascal 16](#_Toc1328273)

[4. Escriba el pseudoc´odigo y el c´odigo de la operaci´on de RECUPERA(↑p, ↑NL), presentando un ejemplo de la lista propuesta. 16](#_Toc1328274)

[Codigo en C++. 16](#_Toc1328275)

[Codigo en Pascal. 16](#_Toc1328276)

[5. Implementar las dem´as operaciones. 16](#_Toc1328277)

[ Imprimir lista. 16](#_Toc1328278)

[ Anular lista. 16](#_Toc1328279)

[ Eliminar elemento de la lista 16](#_Toc1328280)

[ Antes y después. 16](#_Toc1328281)

[ Mostrar primer elemento de la lista. 17](#_Toc1328282)

[6. Implementar una lista num´erica y ordenarla. 17](#_Toc1328283)

# Resultados de la practica.

En esta sección del reporte de practica se presentan los resultados en codigo C++ y algunas capturas de pantalla del funcionamiento del programa.

## Codigo.

#include <stdio.h>

#include <iostream>

using namespace std;

//Estructura básica

struct Nodo{

string nombre;

string genero;

int costo;

Nodo \*sig;

Nodo \*ant;

};

//Prototipado de las funciones y declaraciones de parametros

void insertarElemento(Nodo \*&lista, string nombre, string genero, int costo);

void imprimirElementos(Nodo \*&lista);

void recuperaElementos(Nodo \*lista, string nombre, string genero, int costo);

void eliminarElementos(Nodo \*&lista, string nombre);

void anularElementos(Nodo \*&lista, string &nombre, string &genero, int &costo);

void buscarPrimero(Nodo \*&lista);

void localizarElemento(Nodo \*lista, string nombre);

void despues(Nodo \*&lista);

void antes(Nodo \*&lista);

int main(){

//Declaraciones para el menu

int opcion,close = 1;

//Declaraciones para las funciones

int costo;

string nombre;

string genero;

Nodo \*lista = NULL;

while(close){

cout << "°\tBienvenido a tu WISHLIST de videojuegos." << endl;

cout << "°\t/////////////////////////////////////////////" << endl;

cout << "°\t[1]. Insertar un elemento." << endl;

cout << "°\t[2]. Imprimir la lista." << endl;

cout << "°\t[3]. Buscar (recuperar y localizar) elemento en la lista por (nombre)." << endl;

cout << "°\t[4]. Eliminar elemento de la lista." << endl;

cout << "°\t[5]. Anular la lista. (Borrar todo)." << endl;

cout << "°\t[6]. Retornar e imprimir el primer valor de la lista." << endl;

cout << "°\t[7]. Despues [<-X<-]." << endl;

cout << "°\t[8]. Antes [->X->]." << endl;

cout << "°\t[X ENTERO] salir." << endl;

cout << "°\t/////////////////////////////////////////////" << endl;

cout << "°\n\tIndique que es lo que quiere hacer: ";

cin >> opcion;

switch(opcion) {

case 1:

cout << "\n\tDigite un Nombre: "; /\*\*/ cin >> nombre;

cout << "\n\tDigite un Genero: "; /\*\*/ cin >> genero;

cout << "\n\tDigite un Costo: "; /\*\*/ cin >> costo;

insertarElemento(lista, nombre,genero,costo);

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tEl elemento " << nombre << " ha sido insertado correctamente\n" << endl;

break;

case 2:

cout << "°°°°°°\tLista Impresa" << endl;

imprimirElementos(lista);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tLa lista fue impresa.\n"<< endl;

break;

case 3:

cout << "°°°°°°\tBusqueda de elemento" << endl;

cout << "°°°°°°\tDanos el nombre del juego que buscas: "; cin >> nombre;

recuperaElementos(lista,nombre,genero,costo);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tElemento " << nombre << " estuvo en proceso de busqueda.\n" << endl;

break;

case 4:

cout << "°°°°°°\tBorrar un elemento" << endl;

cout << "°°°°°°\tDanos el nombre del juego que buscas: "; cin >> nombre;

eliminarElementos(lista,nombre);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tElemento " << nombre << " fue eliminado.\n" << endl;

break;

case 5:

cout << "°°°°°°\tBorrar toda la lista" << endl;

anularElementos(lista,nombre,genero,costo);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tToda la lista ha sido eliminada.\n" << endl;

break;

case 6:

cout << "°°°°°°\tBuscar el primer elemento" << endl;

buscarPrimero(lista);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tEl primer elemento ha sido impreso.\n" << endl;

break;

case 7:

cout << "°°°°°°\tAntes y Despues\n" << endl;

despues(lista);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tHemos impreso un elemento despues."<< endl;

break;

case 8:

cout << "°°°°°°\tAntes y Despues\n" << endl;

antes(lista);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tHemos impreso un elemento antes."<< endl;

break;

default:

cout << "Estas seguro de que quiere salir?? // SI = 0 // NO = 1 // :";

int final;

cin >> final;

if(final == 0){

close = 0;

}else{

close = 1;

system("CLS");

}

break;

}

}

}

void insertarElemento(Nodo \*&lista, string nombre, string genero, int costo){

Nodo \*nuevo\_nodo = new Nodo();

nuevo\_nodo->costo = costo;

nuevo\_nodo->genero = genero;

nuevo\_nodo->nombre = nombre;

Nodo \*aux1 = lista;

Nodo \*aux2;

while((aux1 != NULL)){

aux2 = aux1;

aux1 = aux1 -> sig;

}

if(lista == aux1){

lista = nuevo\_nodo;

}else{

aux2->sig = nuevo\_nodo;

}

nuevo\_nodo->sig = aux1;

}

void imprimirElementos(Nodo \*&lista){

Nodo \*actual = new Nodo();

Nodo \*aux = lista;

actual = lista;

while(actual != NULL){

cout << "\t °°°Direccion: " << aux << endl;

cout << "\t °°°Nombre: " << actual -> nombre << endl;

cout << "\t °°°Genero: " << actual -> genero << endl;

cout << "\t °°°Costo: " << actual -> costo << endl;

cout <<"////////////////////////////////////////////\n";

actual = actual -> sig;

}

}

void buscarPrimero(Nodo \*&lista){

for(int i = 1; i <= 1; i++){

Nodo \*aux = lista;

cout << "\t °°°Direccion: " << aux << endl;

cout << "\t °°°Nombre: " << aux -> nombre << endl;

cout << "\t °°°Genero: " << aux -> genero << endl;

cout << "\t °°°Costo: " << aux -> costo << endl;

cout <<"////////////////////////////////////////////\n";

}

}

void recuperaElementos(Nodo \*lista, string nombre, string genero, int costo){

Nodo \*actual = new Nodo();

actual = lista;

while(actual != NULL){

if(actual -> nombre == nombre){

cout << "\t °°°Nombre: " << actual -> nombre << endl;

cout << "\t °°°Genero: " << actual -> genero << endl;

cout << "\t °°°Costo: " << actual -> costo << endl;

cout <<"////////////////////////////////////////////\n";

}

actual = actual -> sig;

}

}

void eliminarElementos(Nodo \*&lista, string nombre){

if(lista != NULL){

Nodo \*aux\_borrar;

Nodo \*ant = NULL;

aux\_borrar = lista;

while((aux\_borrar != NULL) && (aux\_borrar->nombre != nombre)){

ant = aux\_borrar;

aux\_borrar = aux\_borrar -> sig;

}

if(aux\_borrar == NULL){

cout <<"\t°°°°°Este elemento no existe" << endl;

}else if(ant == NULL){

lista = lista -> sig;

delete aux\_borrar;

}else{

ant -> sig = aux\_borrar ->sig;

delete aux\_borrar;

}

}

}

void anularElementos(Nodo \*&lista, string &nombre, string &genero, int &costo){

while(lista != NULL){

Nodo \*aux = lista;

nombre = aux -> nombre;

genero = aux -> genero;

costo = aux -> costo;

lista = aux -> sig;

delete aux;

}

}

void despues(Nodo \*&lista){

Nodo \*actual = new Nodo();

Nodo \*aux = lista;

actual = lista;

cout << "\n°°°°°Despues =>°°°°°" << endl;

while(actual != NULL){

cout << "[[Direccion: " << aux << "]";

cout << "[N: " << actual -> nombre << "]]->";

actual = actual -> sig;

}

cout << " [NULL]\n";

}

void antes(Nodo \*&lista){

Nodo \*actual = new Nodo();

Nodo \*aux = lista;

actual = lista;

cout << "\n°°°°°Antes <=°°°°°" << endl;

while(actual != NULL){

cout << "[[Direccion: " << aux << "]";

cout << "[N: " << actual -> nombre << "]]<-";

actual = actual -> sig;

}

cout << " [NULL]\n";

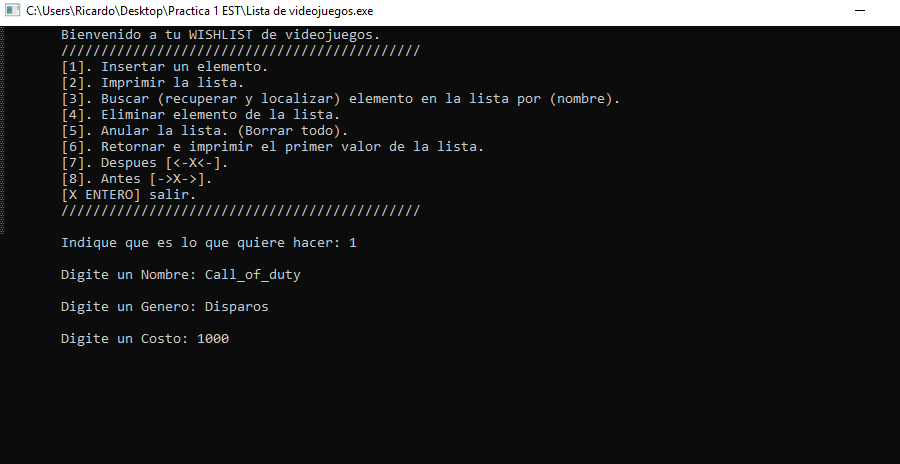
}

## Mostrando cada una de las operaciones que realiza el programa.

### Insertar Elemento.

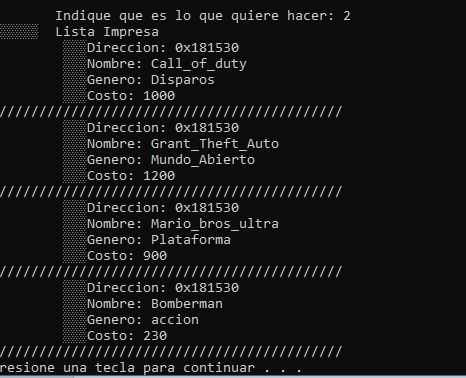
Basicamente lo que hacemos aquí es poner un elemento en nuestra lista, la cual es una WISHLIST o lista de videojuegos que nos gustaría comprar en un futuro.

Para ello ingresamos el valor de 1 en nuestro programa.



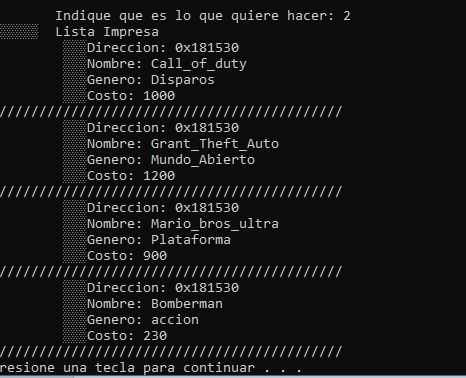
### Imprimir elementos.

Ingresando 2 en nuestro programa podemos desplegar cada uno de los elementos ingresados en nuestra lista.



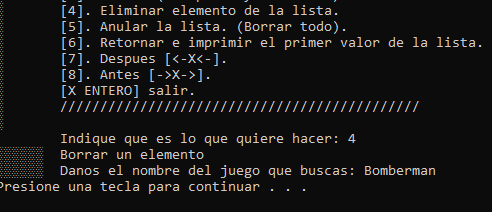
### Localizar y recuperar elemento.

Aquí se implementaron las dos operaciones de manera junta, por lo que el programa al recibir 3 nos dara tanto la dirrecion en memoria de nuestro elemento como también el valor de cada una de las características del mismo.



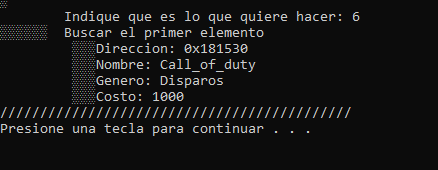
### Eliminar un elemento.

Con el numero 4 podemos eliminar un elemento de nuestra lista con simplemente escribir su nombre



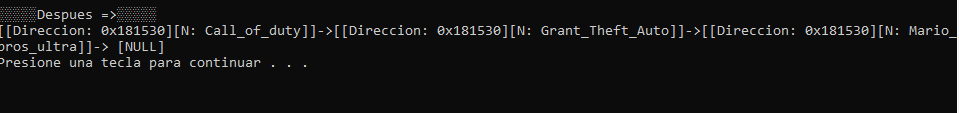
### Buscar el primer elemento de la lista.

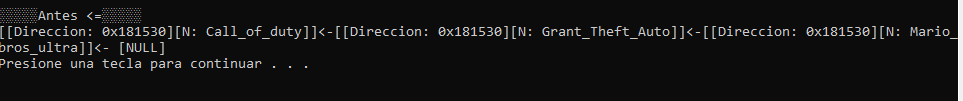
Con el valor 6 podemos buscar solamente el primer elemento de nuestra lista, retornarlo y mostrarlo por medio de una impresión.



### Operación de antes y despues.

Con estas operaciones lo que hacemos es imprimir los elementos de atrás hacia delante y de delante hacia atrás con ayuda de los apuntadores sig y ant.





### Eliminar todos los elementos de la lista (Anular la lista).

Con esta operación básicamente borramos todos y cada uno de los elementos que tenemos en nuestra lista. Simplemente debemos dar como valor en el programa el numero 5.



### Extras.

Se ha agregado un activityLog para saber cual fue la ultima operación que realizamos con la lista y facilitar el manipularla.

# Cuestionario.

1. Escriba el pseudoc´odigo y c´odigo de la construcci´on de la lista enlazada propuesta.

### Codigo en c++. (Incluye menu, creación de la lista y prototipos de funciones)

#include <stdio.h>

#include <iostream>

using namespace std;

//Estructura básica

struct Nodo{

string nombre;

string genero;

int costo;

Nodo \*sig;

Nodo \*ant;

};

//Prototipado de las funciones y declaraciones de parametros

void insertarElemento(Nodo \*&lista, string nombre, string genero, int costo);

void imprimirElementos(Nodo \*&lista);

void recuperaElementos(Nodo \*lista, string nombre, string genero, int costo);

void eliminarElementos(Nodo \*&lista, string nombre);

void anularElementos(Nodo \*&lista, string &nombre, string &genero, int &costo);

void buscarPrimero(Nodo \*&lista);

void localizarElemento(Nodo \*lista, string nombre);

void despues(Nodo \*&lista);

void antes(Nodo \*&lista);

int main(){

//Declaraciones para el menu

int opcion,close = 1;

//Declaraciones para las funciones

int costo;

string nombre;

string genero;

Nodo \*lista = NULL;

while(close){

cout << "°\tBienvenido a tu WISHLIST de videojuegos." << endl;

cout << "°\t/////////////////////////////////////////////" << endl;

cout << "°\t[1]. Insertar un elemento." << endl;

cout << "°\t[2]. Imprimir la lista." << endl;

cout << "°\t[3]. Buscar (recuperar y localizar) elemento en la lista por (nombre)." << endl;

cout << "°\t[4]. Eliminar elemento de la lista." << endl;

cout << "°\t[5]. Anular la lista. (Borrar todo)." << endl;

cout << "°\t[6]. Retornar e imprimir el primer valor de la lista." << endl;

cout << "°\t[7]. Despues [<-X<-]." << endl;

cout << "°\t[8]. Antes [->X->]." << endl;

cout << "°\t[X ENTERO] salir." << endl;

cout << "°\t/////////////////////////////////////////////" << endl;

cout << "°\n\tIndique que es lo que quiere hacer: ";

cin >> opcion;

switch(opcion) {

case 1:

cout << "\n\tDigite un Nombre: "; /\*\*/ cin >> nombre;

cout << "\n\tDigite un Genero: "; /\*\*/ cin >> genero;

cout << "\n\tDigite un Costo: "; /\*\*/ cin >> costo;

insertarElemento(lista, nombre,genero,costo);

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tEl elemento " << nombre << " ha sido insertado correctamente\n" << endl;

break;

case 2:

cout << "°°°°°°\tLista Impresa" << endl;

imprimirElementos(lista);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tLa lista fue impresa.\n"<< endl;

break;

case 3:

cout << "°°°°°°\tBusqueda de elemento" << endl;

cout << "°°°°°°\tDanos el nombre del juego que buscas: "; cin >> nombre;

recuperaElementos(lista,nombre,genero,costo);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tElemento " << nombre << " estuvo en proceso de busqueda.\n" << endl;

break;

case 4:

cout << "°°°°°°\tBorrar un elemento" << endl;

cout << "°°°°°°\tDanos el nombre del juego que buscas: "; cin >> nombre;

eliminarElementos(lista,nombre);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tElemento " << nombre << " fue eliminado.\n" << endl;

break;

case 5:

cout << "°°°°°°\tBorrar toda la lista" << endl;

anularElementos(lista,nombre,genero,costo);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tToda la lista ha sido eliminada.\n" << endl;

break;

case 6:

cout << "°°°°°°\tBuscar el primer elemento" << endl;

buscarPrimero(lista);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tEl primer elemento ha sido impreso.\n" << endl;

break;

case 7:

cout << "°°°°°°\tAntes y Despues\n" << endl;

despues(lista);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tHemos impreso un elemento despues."<< endl;

break;

case 8:

cout << "°°°°°°\tAntes y Despues\n" << endl;

antes(lista);

system("pause");

system("CLS");

cout << "\n°°°°°Actividad reciente°°°°°\tHemos impreso un elemento antes."<< endl;

break;

default:

cout << "Estas seguro de que quiere salir?? // SI = 0 // NO = 1 // :";

int final;

cin >> final;

if(final == 0){

close = 0;

}else{

close = 1;

system("CLS");

}

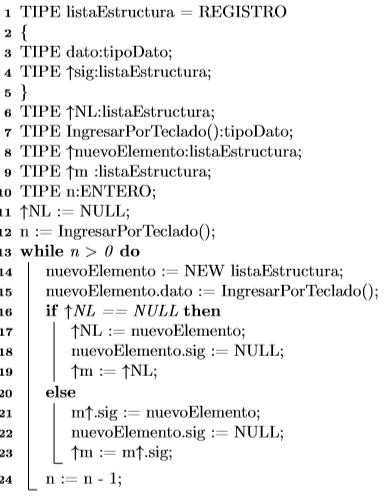
break;

}

}

}

### Codigo en pascal.



## 2. Escriba el pseudoc´odigo y el c´odigo de la operaci´on de ISERTAR(x, ↑p, ↑NL), presentando un ejemplo de la lista propuesta.

### Codigo en C++.

void insertarElemento(Nodo \*&lista, string nombre, string genero, int costo){

Nodo \*nuevo\_nodo = new Nodo();

nuevo\_nodo->costo = costo;

nuevo\_nodo->genero = genero;

nuevo\_nodo->nombre = nombre;

Nodo \*aux1 = lista;

Nodo \*aux2;

while((aux1 != NULL)){

aux2 = aux1;

aux1 = aux1 -> sig;

}

if(lista == aux1){

lista = nuevo\_nodo;

}else{

aux2->sig = nuevo\_nodo;

}

nuevo\_nodo->sig = aux1;

}

### Codigo en pascal.

## 3. Escriba el pseudoc´odigo y el c´odigo de la operaci´on de LOCALIZA(x, ↑NL), presentando un ejemplo de la lista propuesta.

### Codigo en C++.

void localizaElementos(Nodo \*lista, string nombre, string genero, int costo){

Nodo \*actual = new Nodo();

actual = lista;

while(actual != NULL){

if(actual -> nombre == nombre){

cout << "\t °°°Direccion: " << aux << endl;

cout << "\t °°°Nombre: " << actual -> nombre << endl;

cout <<"////////////////////////////////////////////\n";

}

actual = actual -> sig;

}

}

### Codigo en Pascal

## 4. Escriba el pseudoc´odigo y el c´odigo de la operaci´on de RECUPERA(↑p, ↑NL), presentando un ejemplo de la lista propuesta.

### Codigo en C++.

void recuperaElementos(Nodo \*lista, string nombre, string genero, int costo){

Nodo \*actual = new Nodo();

actual = lista;

while(actual != NULL){

if(actual -> nombre == nombre){

cout << "\t °°°Nombre: " << actual -> nombre << endl;

cout << "\t °°°Genero: " << actual -> genero << endl;

cout << "\t °°°Costo: " << actual -> costo << endl;

cout <<"////////////////////////////////////////////\n";

}

actual = actual -> sig;

}

}

### Codigo en Pascal.

## 5. Implementar las dem´as operaciones.

### Imprimir lista.

#### Codigo en c++.

void imprimirElementos(Nodo \*&lista){

Nodo \*actual = new Nodo();

Nodo \*aux = lista;

actual = lista;

while(actual != NULL){

cout << "\t °°°Direccion: " << aux << endl;

cout << "\t °°°Nombre: " << actual -> nombre << endl;

cout << "\t °°°Genero: " << actual -> genero << endl;

cout << "\t °°°Costo: " << actual -> costo << endl;

cout <<"////////////////////////////////////////////\n";

actual = actual -> sig;

}

}

#### Codigo en Pascal.

### Anular lista.

#### Codigo en c++.

void anularElementos(Nodo \*&lista, string &nombre, string &genero, int &costo){

while(lista != NULL){

Nodo \*aux = lista;

nombre = aux -> nombre;

genero = aux -> genero;

costo = aux -> costo;

lista = aux -> sig;

delete aux;

}

}

#### Codigo en Pascal.

### Eliminar elemento de la lista

#### Codigo en c++.

void eliminarElementos(Nodo \*&lista, string nombre){

if(lista != NULL){

Nodo \*aux\_borrar;

Nodo \*ant = NULL;

aux\_borrar = lista;

while((aux\_borrar != NULL) && (aux\_borrar->nombre != nombre)){

ant = aux\_borrar;

aux\_borrar = aux\_borrar -> sig;

}

if(aux\_borrar == NULL){

cout <<"\t°°°°°Este elemento no existe" << endl;

}else if(ant == NULL){

lista = lista -> sig;

delete aux\_borrar;

}else{

ant -> sig = aux\_borrar ->sig;

delete aux\_borrar;

}

}

}

#### Codigo en Pascal.

### Antes y después.

#### Codigo en c++.

void despues(Nodo \*&lista){

Nodo \*actual = new Nodo();

Nodo \*aux = lista;

actual = lista;

cout << "\n°°°°°Despues =>°°°°°" << endl;

while(actual != NULL){

cout << "[[Direccion: " << aux << "]";

cout << "[N: " << actual -> nombre << "]]->";

actual = actual -> sig;

}

cout << " [NULL]\n";

}

void antes(Nodo \*&lista){

Nodo \*actual = new Nodo();

Nodo \*aux = lista;

actual = lista;

cout << "\n°°°°°Antes <=°°°°°" << endl;

while(actual != NULL){

cout << "[[Direccion: " << aux << "]";

cout << "[N: " << actual -> nombre << "]]<-";

actual = actual -> sig;

}

cout << " [NULL]\n";

}

#### Codigo en Pascal.

### Mostrar primer elemento de la lista.

#### Codigo en c++.

void buscarPrimero(Nodo \*&lista){

for(int i = 1; i <= 1; i++){

Nodo \*aux = lista;

cout << "\t °°°Direccion: " << aux << endl;

cout << "\t °°°Nombre: " << aux -> nombre << endl;

cout << "\t °°°Genero: " << aux -> genero << endl;

cout << "\t °°°Costo: " << aux -> costo << endl;

cout <<"////////////////////////////////////////////\n";

}

}

#### Codigo en Pascal.

## 6. Implementar una lista num´erica y ordenarla.

### Codigo en C++ (Ordenamiento de burbuja)

#include <iostream>

#include <stdlib.h>

using namespace std;

struct nodo{

int nro; // en este caso es un numero entero

struct nodo \*sgte;

};

typedef struct nodo \*Tlista;

Tlista inicio, fin;

void generarLista( Tlista &inicio, Tlista &fin, int n )

{

Tlista q, t;

for(int i=0; i<n; i++)

{

q = new(struct nodo);

q->nro = rand()%51;

if(inicio==NULL)

{

q->sgte = inicio;

inicio = q;

fin = q;

}

else

{

q->sgte = fin->sgte;

fin->sgte = q;

fin = q;

}

}

cout<<"\n\n\tLista de numeros generados... "<<endl;

}

void reportarLista(Tlista inicio)

{

while(inicio != NULL)

{

cout <<" " << inicio->nro ;

inicio = inicio->sgte;

}

}

void ordenarLista(Tlista lista)

{

Tlista actual , siguiente;

int t;

actual = lista;

while(actual->sgte != NULL)

{

siguiente = actual->sgte;

while(siguiente!=NULL)

{

if(actual->nro > siguiente->nro)

{

t = siguiente->nro;

siguiente->nro = actual->nro;

actual->nro = t;

}

siguiente = siguiente->sgte;

}

actual = actual->sgte;

siguiente = actual->sgte;

}

cout<<"\n\n\tLista ordenada..."<<endl;

}

void menu()

{

cout<<"\n\t\tORDENAMIENTO DE UNA LISTA ENLAZADA SIMPLE\n\n";

cout<<" 1. GENERAR NUMEROS "<<endl;

cout<<" 2. MOSTRAR NUMEROS "<<endl;

cout<<" 3. ORDENAR NUMEROS "<<endl;

cout<<" 4. SALIR "<<endl;

cout<<"\n INGRESE OPCION: ";

}

int main()

{

inicio = NULL;

fin = NULL;

int op; // opcion del menu

int num; // elemenento a ingresar

do

{

menu(); cin>> op;

switch(op)

{

case 1:

cout<< "\n Cantidad de numeros: "; cin>> num;

generarLista( inicio, fin, num );

break;

case 2:

cout<<"\n\n LISTA:\n\n";

reportarLista( inicio );

break;

case 3:

ordenarLista( inicio );

break;

}

cout<<endl<<endl;

system("pause"); system("cls");

}while(op!=4);

system("pause");

return 0;

}