CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Estructuras de datos

Listas doblemente ligadas

Prof.: Noe Ortega Sanchez

Alumno: Edgar Antonio Delgadillo Villegas

Carrera: Ingeniería Informática

Materia: i5886 Estructura de datos I

NRC: 182881

Sección: D17

Calendario: 2021A

"Lista doble ligadas"

Descripción:

En este programa se realizó un breve código de listas doblemente ligadas, a través de un programa de productos, donde pedían el nombre del producto, precio e Id, adicional te permitía eliminar, añadir, mostrar y vaciar los productos deseados.

A continuación mostrare unas breves imágenes de mi código y como lo realice:

Producto.h

Demostración del producto con sus respectivos gettes y setters

```
using namespace std;

//clase Producto: nombre:string, precio:float, id:int

class Producto {
    private:
        string nombre;
        float precio;
        int id;

    public:
        Producto();
        Producto(string,float,int);
        ~Producto();
        int getId(void);
        void setId(int);
        string getNombre(void);
        void setNombre(string);
        float getPrecio(void);
        void setPrecio(float);

};
```

Producto.cpp

```
#include "Producto.h"

Producto::Producto() { }

Producto::~Producto() { }

string Producto::getNombre(void) {return nombre;}

void Producto::setNombre(string nombre_producto) {nombre = nombre_producto;}

int Producto::getId(void) {return id;}

void Producto::setId(int id_producto) {id = id_producto;}

float Producto::getPrecio(void) {return precio;}

void Producto::setPrecio(float precio_producto) {this->precio = precio_producto;};
```

Main

En el main realice la elaboración del menú y como se imprimiría todo, no mostrare todo el código ya que considero pertinente para una mejor explicación y no atiborrar el contenido de puro código, adicional se adjuntara al classroom el (.zip) del código de ser necesario verlo.

```
#include "ListaD.h"

using namespace std;

int main() {
    ListaD *p = new ListaD;
    int opcEliminar = 6;
    int opcAgregar = 6, posicion;


    ///Menu Principal
    int opcLista = 6;
    while (opcLista != 0) {
        system("cls");
        cout << " ------------------------------------------------------------------------------" << endl;
        cout << "|\n                          Menu Principal                                    \n|" << endl;
        cout << " ------------------------------------------------------------------------------" << endl;
        cout << "\n1) Agregar " << endl << endl;
        cout << "2) Buscar " << endl << endl;
        cout << "3) Eliminar " << endl << endl;
        cout << "4) Mostrar " << endl << endl;
```

```
                p->insertaFinal(producto);
                cout << endl;
                system("pause");
            } break;

            case 3: {system("cls");
                cout << " ------------------------------------------------------------------------------------
                cout << "|\n                                    Agregar                                    \n|
                cout << " ------------------------------------------------------------------------------------

                cin.ignore(1000,'\n');
                cout << " Nombre: ";
                getline(cin,nombre);
                producto.setNombre(nombre);

                cout << "\n Precio: ";
                cin >> precio;
                producto.setPrecio(precio);
                cin.ignore(1000,'\n');
```

```
case 2: {
    system("cls");
    Producto producto;
    string nombre;

    cout << " ------------------------------------------------------------------------------------" << endl;
    cout << "|\n                                  Buscar                                    \n|" << endl;
    cout << " ------------------------------------------------------------------------------------" << endl;

    cin.ignore(1000,'\n');
    cout << " Nombre: ";
    getline(cin,nombre);

    p->buscar(nombre);
    cout << endl;
    system("pause");
}
break;
```

```
   case 3: {
            ///Menu Eliminar
            Producto producto;
            string nombre;

            while (opcEliminar != 0) {
                system("cls");
                cout << " ------------------------------------------------------------------------------------" << endl;
                cout << "|\n                                  Eliminar                                  \n|" << endl;
                cout << " ------------------------------------------------------------------------------------" << endl;
                cout << "1) Eliminar producto" << endl << endl;
                cout << "2) Eliminar productos" << endl << endl;
                cout << "0) Salir" << endl;
                cout << "Opcion: ";
                cin.ignore(1000,'\n');
                cin >> opcEliminar;
                cout << endl;
```

```
            break;

        case 5: system("cls");
                cout << " ------------------------------------------------------------------------------" << endl;
                cout << "|\n                    Numero de productos                                    \n|" << endl;
                cout << " ------------------------------------------------------------------------------" << endl;

                p->tamano();
                cout << endl;
                system("pause");
            break;

        case 0: cout << "Presione (enter) para salir " << endl;
                cout << endl;
                system("pause");
            break;

        default: cout << "No existe opcion" << endl;
                system("pause");
```

Nodo.h

```
#ifndef NODO_H
#define NODO_H
#include "Producto.h"

class Nodo
{
        public:
         void eliminar_todo();
         Nodo();
         ~Nodo();
         Producto dato;
         Nodo *sig;
         Nodo *ant;
//     protected:
//
//     private:
};

#endif // NODO_H
```

Aquí en el nodo.h se muestra como use la función elimnar_todo(); para de cierta forma usarla al momento de vaciar, se muestra como use la clase producto para indicar o guardar el dato como si fuese un producto y por último se muestra sus respectivos nodo el siguiente y anterior necesarios para las listas doblemente ligadas

Nodo.cpp

```
void Nodo::eliminar_todo()
{
    if (sig)
        sig->eliminar_todo();
    delete this;
}

Nodo::Nodo() {
    dato.getNombre();
    dato.getPrecio();
    dato.getId();

    sig = nullptr;
    ant = nullptr;
}

Nodo::~Nodo() {

}
```

Lista.h

```
class ListaD
{
    public:
        ListaD();
        ~ListaD();

        void inicializa(void);
        void vaciar();
        void mostrarTodo();
        void ultimo();
        bool vacia();
        int tamano();
        void primero();
        void insertaInicio(Producto);
        void insertaFinal(Producto);
        void insertaPos(Producto,int);
        void eliminar(string);
        void buscar(string);
        Nodo* anterior(Producto);
        Nodo* siguiente(Producto);


 private:
        Nodo *h; //ancla para siguiente
        Nodo *t; //ancla para anterior


};
```

En estos dos, en lista.h y en lista.cpp se muestra que métodos utilice tanto el mostrar inicio como, mostrar final, insertar en posición, eliminar, buscar, inicializa, vaciar, mostrar todo, ultimo, vacía, tamaño, primero entre otros, en el cpp se muestra como lo implemente e hice que funcionara, como comente anteriormente, no mostrare el código completo para no atiborrar el ensayo de puro código pero adicional anexare el (.zip) al classrom de ser necesario verlo.

Lista.cpp

```
ListaD::ListaD() { //
    inicializa();
}

ListaD::~ListaD() { //

}



bool ListaD::vacia() { //
    if (h == nullptr and t == nullptr) {
        return true;
    } else {
        return false;
    }
}
```

```
int ListaD::tamano() { //
    int cont = 0;
    Nodo *aux = h;
    while(aux){
        cont++;
        aux = aux->sig;
    }
    cout << " " << cont << " " << endl;
    return cont;
}

void ListaD::inicializa(void) { //
    h = nullptr;
    t = nullptr;
}



void ListaD::insertaFinal(Producto d) { //
    Nodo *tmp = new Nodo();
    tmp->dato = d;
    Nodo *aux = h;

    if(aux == nullptr){
        h = tmp;
    } else {
        while(aux->sig) {
            aux->ant = aux;
            aux = aux->sig;
        }
```

```
void ListaD::insertaPos(Producto d, int pos) { //
    Nodo *tmp = new Nodo();
    tmp->dato = d;
    Nodo *aux = h, *auxRet;

    int i = 0;

    if(pos>= 1 and pos-1<tamano()) {
        while(i<pos-1 and aux) {
            tmp->ant = aux;
            tmp->sig = aux->sig;
            aux->sig->ant = tmp;
            aux->sig = tmp;
            i++;
        }

        if (aux == h) {
            tmp->sig = h;
            h = tmp;
        } else {
            tmp->ant->sig = tmp;
```

```
void ListaD::insertaInicio(Producto d) { //
    Nodo *tmp = new Nodo();
    tmp->dato = d;
    if (h == nullptr and t == nullptr){
        h = tmp;
        t = tmp;
    } else {
        tmp->sig = h;
        h->ant = tmp;
        h = tmp;
    }
}

void ListaD::mostrarTodo() { //
    Nodo *aux = h;
    while(aux) {
        cout << " Nombre: " << aux->dato.getNombre() << endl;
        cout << " Id:     " << aux->dato.getId() << endl;
        cout << " Precio: " << aux->dato.getPrecio() << endl;
        aux = aux->sig;
    }
}
```

```
void ListaD::vaciar() {
    Nodo *aux;
    if (h != nullptr) {
        aux = h;
        while(aux != nullptr) {
            h = h->sig;
            delete aux;
            aux = h;
        }
    } else {
        cout << "lista vacia" << endl;
    }
}

void ListaD::primero() { //
    if (vacia()) {
        cout << "Lista vacia" << endl;
    } else {
        cout << h <<endl;
    }
}

void ListaD::ultimo() { //
    if (vacia()) {
        cout << "Lista vacia" << endl;
    } else {
        Nodo *aux = h;
        while(aux->sig != nullptr){
            aux = aux->sig;
        }
```

```
Nodo* ListaD::anterior(Producto d) {
    if (vacia()) {
        cout << "Lista vacia" << endl;
    } else {
        Nodo *aux = h;
        Nodo *auxR = nullptr;
        bool band = true;

        if(aux) {
            while(aux and band) {
                if(aux->dato.getNombre() != d.getNombre()) {
                auxR = aux;
                aux = aux->sig;
                } else {
                    band = false;
                }
            }

            if(auxR == nullptr) {
                cout << "Primera posicion " << endl;
                return auxR;
            } else if(aux == nullptr) {
                cout << "no existe " << endl;
                return nullptr;
            } else {
                cout << "Anterior es: " << auxR << endl;
                return auxR;
            }
        }
```

```
Nodo* ListaD::siguiente(Producto d) {
    if (vacia()) {
        cout << "Lista vacia" << endl;
    } else {
        Nodo *auxS = h;
        bool band = true;

        if(auxS) {
            while(auxS and band) {
                if(auxS->dato.getNombre() != d.getNombre()) {
                    auxS = auxS->sig;
                } else {
                    band = false;
                }
            }

            if(auxS == nullptr) {
                cout << "no existe " << endl;
                return nullptr;
            } else if(auxS->sig == nullptr) {
                cout << "Ultima posicion " << endl;
                return auxS;
            } else {
                cout << "Elemento siguiente: "<< auxS->sig << endl;
                return auxS->sig;
            }
        }
    }
    return nullptr;
}
```

Compilando el Programa

```
--------------------------------------------------------------------------------

                          Menu Principal

--------------------------------------------------------------------------------

1) Agregar
2) Buscar
3) Eliminar
4) Mostrar
5) Cantidad(Tamano)
0) Salir
opcion:
```

```
                              Agregar
 ------------------------------------------------------------------------------
1) Agregar al inicio
2) Agregar al final
3) Agregar posicion
0) salir
Opcion:
```

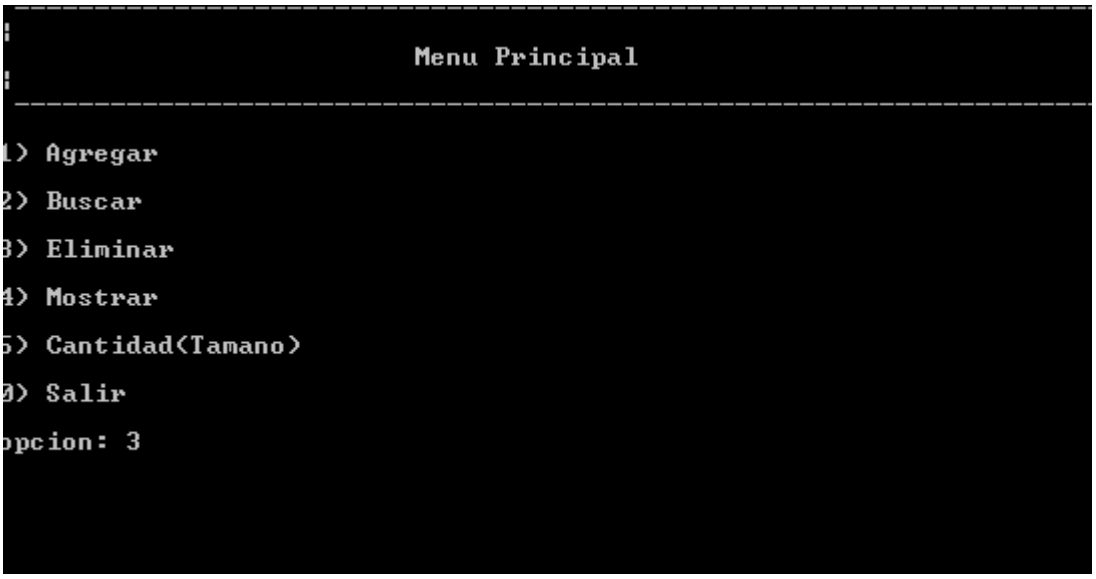
```
 ------------------------------------------------------------------------------
                              Agregar
 ------------------------------------------------------------------------------
 Nombre: pepsi
 Precio: 23
 Id: 2
```

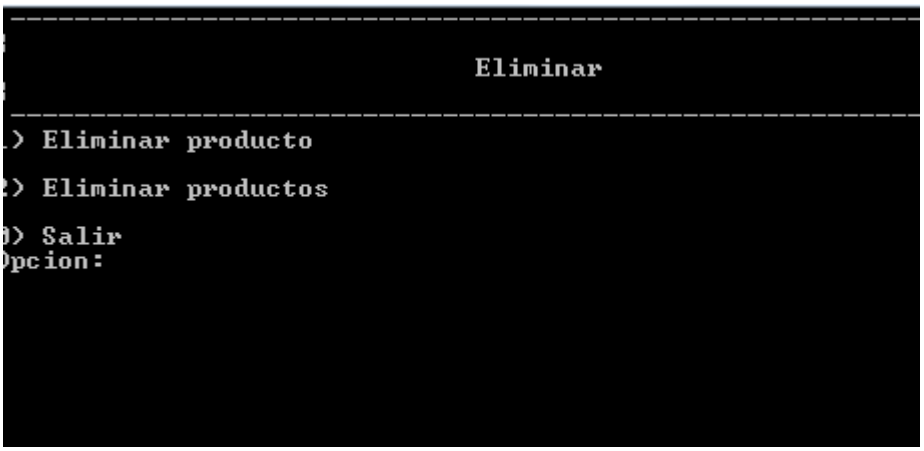
```
                           Menu Principal
 ------------------------------------------------------------------------------
1) Agregar
2) Buscar
3) Eliminar
4) Mostrar
5) Cantidad(Tamano)
0) Salir
opcion: 4
```

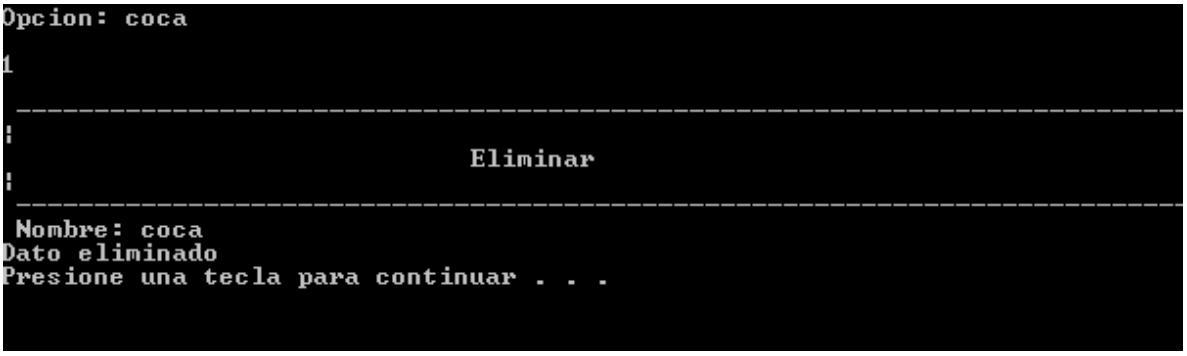
```
Nombre: pepsi
Id:     2
Precio: 23

Presione una tecla para continuar . . .
```

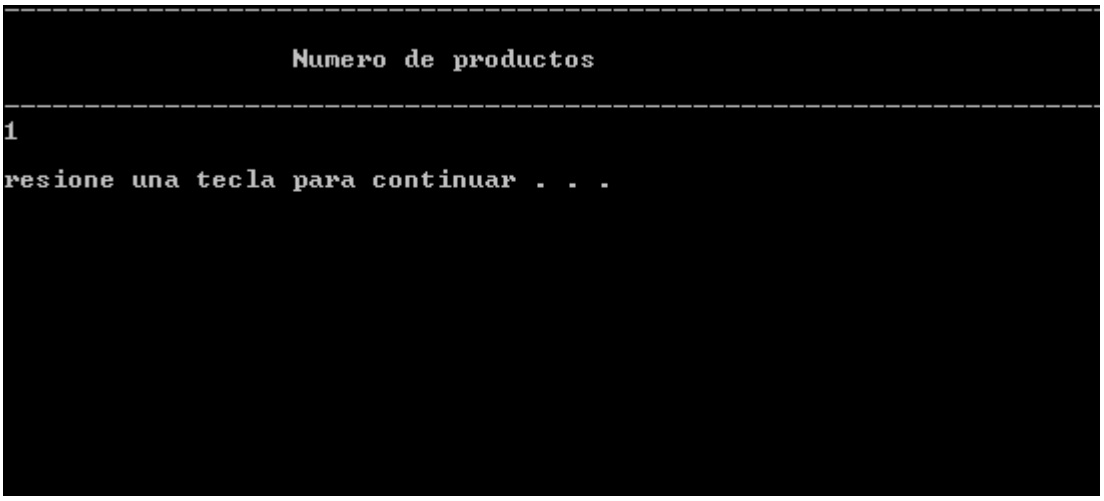
```
--------------------------------------------------------------------------------
                                Buscar
--------------------------------------------------------------------------------
Nombre: pepsi
```

```
--------------------------------------------------------------------------------
                                Buscar
--------------------------------------------------------------------------------
Nombre: pepsi
Nombre: pepsi
Precio: 23
Posicion: 0
Id:     2


Presione una tecla para continuar . . .
```


Menu Principal
1) Agregar
2) Buscar
3) Eliminar
4) Mostrar
5) Cantidad(Tamano)
0) Salir
Opcion: 3


Eliminar
1) Eliminar producto
2) Eliminar productos
0) Salir
Opcion:


Opcion: coca
1
Eliminar
Nombre: coca
Dato eliminado
Presione una tecla para continuar . . .



Opinión;

Este programa por lo menos para mí fue un poco más rápido de hacer, ya que teníamos las bases del último programa, de la lista simple solo fue reutilizarlo pero ahora con doblemente ligadas, para mí fue un poco complicado entender la teoría ya que no se me da muy bien, suelo comprender más con código, pero siento que eso me ayudara para comprender y dejarle ese miedo a la teoría, le batalle algo para entender muy bien el insertar posición pero por lo menos para mí al volver a ver el video de las clases del profe y el juego que hizo de (sig->sig->ant) y adivinar en qué posición quedaba, fue mucho más fácil comprender y una excelente idea. De ahí en más todo fue más sencillo y una excelente manera de comprender.