CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Estructuras de datos

Lista simplemente 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 simplemente ligada"

Descripción:

En este programa se realizó un breve código de listas simplemente ligadas, a través de un programa que registra nombres de alumnos en mi caso, adicional te permitía eliminar, añadir, mostrar y vaciar los productos deseados, usando una lista simple.

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

Main

En el main realice la elaboración del menú y como se imprimiría todo, use distintos casos de switch para poder darle un mejor manejo de la información, 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.

```
using namespace std;

int main()
{
    Lista* Nom = new Lista;
    int opcLista;
    int opcElimniar;
    int opcAgregar, posicion;
    string Nombre;

    do {
        cout<< Nombre << endl;
        cout<<"-------------------------------------------------------------------------------"<<endl;
        cout<<"|                          Nombre de alumnos                                  |"<<endl;
        cout<<"-------------------------------------------------------------------------------"<<endl;
        cout << "" << endl;
        cout << "1) ingresar nombre\n" << endl;
        cout << "2) buscar nombre\n" << endl;
        cout << "3) Eliminar nombre\n" << endl;
```

```
cout << "4) Mostrar todos los nombres\n" << endl;
cout << "5) Ver cantidad de nombres\n" << endl;
cout << "0) Salir \n\n" << endl;
cout << "\t\t\t\tOpcion: ";
cin >> opcLista;
cin.ignore();
system("cls");

switch (opcLista) {
case 1:
    cout<<"-------------------------------------------------------------------------------------"<<endl;
    cout<<"|                                    alumnos                                        |"<<endl;
    cout<<"-------------------------------------------------------------------------------------"<<endl;
    cout << "1) Agregar al inicio\n" << endl;
    cout << "2) Agregar al final\n" << endl;
    cout << "3) Agregar en una posicion deseada\n" << endl;
    cout << "4) Terminar de agregar materias\n" << endl;
    cout << "6) Salir \n\n" << endl;
    cout << "\t\t\t\tOpcion: ";



    cin >> opcAgregar;
    cin.ignore();
    fflush(stdin);
    system("cls");



    switch (opcAgregar) {
    case 1:
    {
        cout<<"-------------------------------------------------------------------------------------"<<endl;
        cout<<"|                              Agregar al Inicio                                    |"<<endl;
        cout<<"-------------------------------------------------------------------------------------"<<endl;
        cout << "\nNombre del Alumno: ";
        fflush(stdin);

        getline(cin, Nombre);

        Nom->insertaInicio(Nombre);
        system("pause");
        system("cls");


    break;
}

case 2:
{

    cout<<"-------------------------------------------------------------------------------------"<<endl;
    cout<<"|                              Agregar al Final                                     |"<<endl;
    cout<<"-------------------------------------------------------------------------------------"<<endl;
    cout << "\nNombre del Alumno: ";
    getline(cin, Nombre);
    Nom->insertaFinal(Nombre);
    system("cls");

} break;
```

```
case 3:
{

    cout<<"-------------------------------------------------------------------------------"<<endl;
    cout<<"|                          Agregar en posicion                                |"<<endl;
    cout<<"-------------------------------------------------------------------------------"<<endl;
    cout << "\nNombre del Alumno: ";
    getline(cin, Nombre);
    cout << "Posicion:\n ";
    fflush(stdin);
    cin >> posicion;
    Nom->insertaPos(Nombre, posicion);
    system("cls");

}   break;


            }   break;

            case 0:

                cout << "Ha salido del menu" << endl;
                system("cls");

                break;

            default:
                cout << "Opcion no valida" << endl;
                break;
            }

            break;
        case 2:
        {
            string N;

            cout << "Nombre ah buscar: ";
            getline(cin, N);
```

```
            cout << "Nombre ah buscar: ";
            getline(cin, N);
            Nom->buscar(N);
            system("cls");
            break;

    }


    case 3:
        cout<<"-------------------------------------------------------------------------------"<<endl;
        cout<<"|                            Eliminar Alumnos                                 |"<<endl;
        cout<<"-------------------------------------------------------------------------------"<<endl;
        cout << "1) Eliminar un Nombre\n" << endl;
        cout << "2) Eliminar todos los Nombres\n" << endl;
        cout << "0) Salir del menu\n" << endl;
        cout << "\t\t\t\tOpcion: ";
        cin >> opcElimniar;
        cin.ignore();
```

```
            ,

        case 4:
            Nom->mostrarTodo();
            system("pause");
            break;

        case 5:
            Nom->tamano();

            break;

        case 6:
            cout << "Ha salido del sistema " << endl;

            break;

        default: cout << "Opcion no valida" << endl;
        }
    } while (opcLista != 0);
```

Nodo.h

```
#ifndef NODO_H
#define NODO_H

#include <iostream>

using namespace std;

class Nodo
{
public:
    Nodo();
    ~Nodo();


    string dato;
    Nodo* sig;
};

#endif // NODO_H
```

Aquí en el nodo se muestra como utilice para dato un string ya que estaría guardando como datos nombre y por último se muestra sus respectivos nodo el siguiente(sig) ya que es necesarios para las listas simplemente ligadas.

Nodo.cpp

```
#include "Nodo.h"

Nodo::Nodo()
{
    dato = "";
    sig = nullptr;
}

Nodo::~Nodo()
{

}
```

Lista.h

```
public:
    Lista();
    ~Lista();
    Nodo* anterior(string);
    Nodo* siguiente(string);
    void elimiar(string);
    void ultimo();
    void primero();
    void incializa(void);
    int tamano();
    void insertaFinal(string);
    void mostrarTodo();
    void vaciar();
    bool vacia();
    void buscar(string);
    void insertaPos(string, int);
    void insertaInicio(string);

private:
    Nodo* h;
```

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

```
int Lista::tamano()
{
    int cont = 0;
    Nodo* aux = h;
    while (aux)
    {
        cont++;
        aux = aux->sig;

    }
    cout << cont << endl;
    return cont;
}
void Lista::incializa(void)
{
    h = nullptr;

}
void Lista::insertaPos(string d, int pos)
{
    Nodo* tpm = new Nodo();
    tpm->dato = d;
    Nodo* aux = h, * auxret = h;

    int i = 0;
    if (pos >= 1 && pos - 1 < tamano())
    {


        while (i < pos - 1 && aux)
        {
            auxret = aux;
            aux = aux->sig;
            i++;
        }
        if (aux == h)
```

```
            i++;
        }
        if (aux == h)
        {
            tpm->sig = h;
            h = tpm;
        }
        else
        {
            auxret->sig = tpm;
            tpm->sig = aux;
        }
    }
    else
    {
        cout << "posision no valida" << endl;
    }
}

void Lista::insertaInicio(string d)
{
    Nodo* tmp = new Nodo;
    tmp->dato = d;

    if (vacia())
    {
        h = tmp;
    }
    else
    {
        tmp->sig = h;
        h = tmp;
    }
}
void Lista::insertaFinal(string d)
{
    Nodo* tpm = new Nodo();
    tpm->dato = d;
    Nodo* aux = h;
```

```
    tpm->dato = d;
    Nodo* aux = h;

    if (aux == nullptr)
    {
        h = tpm;
    }
    else
    {
        while (aux->sig)
        {
            aux = aux->sig;
        }
        aux->sig = tpm;
    }
}
void Lista::elimiar(string d)
{
    Nodo* aux = h;
    Nodo* auxR = nullptr;
    bool band = true;
```

```
    Nodo* aux = h;
    Nodo* auxR = nullptr;
    bool band = true;
    if (aux)
    {
        while (aux && band)
        {
            if (aux->dato != d)
            {
                auxR = aux;
                aux = aux->sig;
            }
            else
            {
                band = false;

            }
        }
        if (aux == nullptr)
        {
            cout << "Dato no encontrado" << endl;
```

```
        if (aux == nullptr)
        {
            cout << "Dato no encontrado" << endl;
        }
        else if (aux == h)
        {
            h = h->sig;
            delete aux;
        }
        else
        {
            auxR->sig = aux->sig;
            delete aux;
        }

    }
    else
    {
        cout << "lista vacia" << endl;
    }
```

```
bool Lista::vacia()
{
    if (h == nullptr)
    {
        //cout << "Si esta vacia" << endl;
        return true;
    }
    else
    {
        //cout << "No esta vacia" << endl;
        return false;
    }
}
Nodo* Lista::siguiente(string d)
{
    if (vacia()) {
        cout << "La lista esta vacia" << endl;
    }
    else {
        Nodo* aux = nullptr;
        Nodo* auxS = h;
        bool band = true;

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

```
void Lista::ultimo()
{
    if (vacia()) {
        cout << "La lista esta vacia" << endl;
    }
    else {
        Nodo* aux = h;
        while (aux->sig != nullptr) {
            aux = aux->sig;
        }
        cout << "Ultimo elemento= " << aux << endl;
    }
}
Nodo* Lista::siguiente(string d)
                band = false;
            }
        }

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

```
void Lista::primero()
{
    if (vacia()) {
        cout << "La lista esta vacia" << endl;
    }
    else {
        cout << "Primer elemento = " << h << endl;
    }
}
Nodo* Lista::anterior(string d)
{
    if (vacia()) {
        cout << "La lista esta vacia" << endl;
    }
    else {
        Nodo* aux = h;
        Nodo* auxR = nullptr;
        bool band = true;

        if (aux) {
        bool band = true;

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

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

```
            else {
                cout << "Elemento anterior a " << d << ": " << auxR << endl;
                return auxR;
            }
        }
    }
    return nullptr;
}

void Lista::buscar(string d) {
    Nodo* aux = h;
    int cont = 1;
    int cont2 = 0;

    if (vacia()) {
        cout << "Error, lista vacia" << endl;
    }
    else {
        while (aux) {
            if (aux->dato == d) {
                cout << "El dato " << d << " se encuentra en la posicion: " << cont - 1 << endl;
```

```
        cout << "Error, lista vacia" << endl;
    }
    else {
        while (aux) {
            if (aux->dato == d) {
                cout << "El dato " << d << " se encuentra en la posicion: " << cont - 1 << endl;
                cont2++;
            }
            aux = aux->sig;
            cont++;
        }
        if (cont2 == 0) {
            cout << "No existe el dato (buscar)" << endl;
        }
        cout << endl << endl;
    }
}
```

```
void Lista::mostrarTodo()
{
    Nodo* aux = h;
    if (aux)
    {
        while (aux) {
            cout << aux->dato << "= " << &aux->dato << endl;
            aux = aux->sig;
        }
    }
    else
    {
        cout << "lista vacia" << endl;
    }
}
void Lista::vaciar()
{
    if (vacia()) {
        cout << "Error,lista vacia" << endl;
    }
    else {
        cout << "Error,lista vacia" << endl;
    }
    else {
        //h->eliminar_todo();
        h = 0;
        cout << "Lista vaciada con exito" << endl;
    }


ista::Lista()

    incializa();


ista::~Lista()
```

Compilando el Programa

```
                        Nombre de alumnos                                 !
-------------------------------------------------------------------------------

> ingresar nombre
> buscar nombre
> Eliminar nombre
> Mostrar todos los nombres
> Ver cantidad de nombres
> Salir

                              Opcion:
```

```
-------------------------------------------------------------------------------
!                                  alumnos                                    !
-------------------------------------------------------------------------------
1> Agregar al inicio
2> Agregar al final
3> Agregar posicion
4> Terminar de agregar
6> Salir

                              Opcion:
```

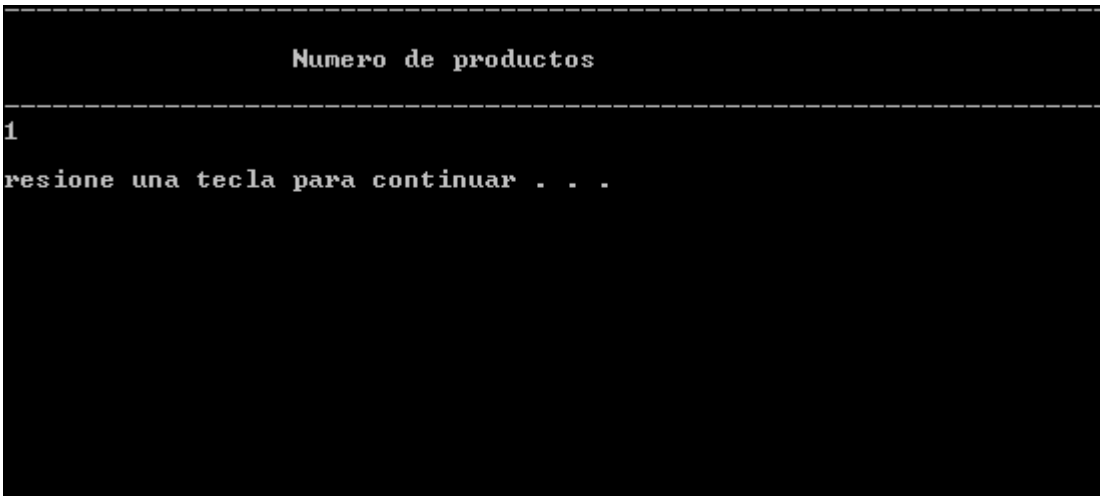
```
                              Agregar al Inicio

Nombre del Alumno: edgar
```

```
Nombre ah buscar: edgar
```

```
                              Eliminar Alumnos

1) Eliminar Nombre
2) Eliminar todos los Nombres
0) Salir

                                Opcion:


lista vacia
Presione una tecla para continuar . . .
```

```
pedro= 0x6c1338
noe= 0x6c1310
edgar= 0x6c12e8
Presione una tecla para continuar . . .
```

```
3
pedro
--------------------------------------------------------------------------------
|                              Nombre de alumnos                               |
--------------------------------------------------------------------------------

1) ingresar nombre

2) buscar nombre

3) Eliminar nombre

4) Mostrar todos los nombres

5) Ver cantidad de nombres

0) Salir

                                  Opcion: 5
```



Opinión;

Este programa por lo menos para mí fue un poco complicado, ya que además de tener demasiada tarea, siento que me revolvió la teoría pero en cuanto volví a mirar los videos y viendo poco a poco todo, ya comencé a entender lo que es una lista y como se implementa.