UNIVERSIDAD AUTONOMA DE QUERETARO
Facultad de informática

Menú de listas

Estructura de datos

CARLOS NOGUEZ JUAREZ

315398

GRUPO 35

FECHA 17/04/2024

Codigo:

```
#include <iostream>
#include <cstdlib>


using namespace std;

struct Nodo
{
   int data;
   Nodo *next;
};

void insertList(Nodo *&, int);
void mostrarLista(Nodo *);
void buscarLista(Nodo *, int);
bool existeEnLista(Nodo *, int);
void eliminarNodo(Nodo*&, int);
void eliminarLista(Nodo*&);

Nodo *head = NULL;

int main() {
   system("cls");
   int opcion, n;

   do {
      cout << "*********************************" << endl;
      cout << "Menu de opciones:" << endl;
      cout << "1. Insertar elemento" << endl;
      cout << "2. Mostrar lista" << endl;
      cout << "3. Buscar elemento" << endl;
      cout << "4. Eliminar nodo" << endl;
      cout << "5. Eliminar lista" << endl;
      cout << "6. Salir" << endl;
      cout << "Ingrese una opcion: ";
      cin >> opcion;

      switch (opcion) {
         case 1:
```

```
            cout << "Ingrese el contenido de data: ";
            cin >> n;
            if (existeEnLista(head, n)) {
              cout << "El elemento " << n << " ya esta en la lista. No se puede agregar nuevamente." <<
endl;
            } else {
              insertList(head, n);
              cout << "Elemento insertado correctamente." << endl;
            }
            break;
          case 2:
            mostrarLista(head);
            break;
          case 3:
            cout << "Ingresa el elemento a buscar: ";
            cin >> n;
            buscarLista(head, n);
            break;
          case 4:
            eliminarNodo(head, n);
            mostrarLista(head);
            break;
          case 5:
            cout << "Eliminando lista..." << endl;
            eliminarLista(head);
            mostrarLista(head);
            break;
          case 6:
            cout << "Saliendo del programa..." << endl;
            break;
          default:
            cout << "Opcion invalida. Intente nuevamente." << endl;
        }
    } while (opcion != 6);

    return 0;
}

void mostrarLista(Nodo *head) {
    Nodo *actual = head;
    if (actual == NULL) {
        cout << "La lista esta vacia." << endl;
    } else {
        cout << "Elementos de la son: ";
        while (actual != NULL) {
```

```
        cout << "|" << actual->data << "| -> ";
        actual = actual->next;
      }
      cout << "NULL" << endl;
   }
}

void insertList(Nodo *&head, int n) {
   Nodo *new_nodo = new Nodo();
   new_nodo->data = n;
   Nodo *aux1 = head;
   Nodo *aux2 = NULL;

   while ((aux1 != NULL) && (aux1->data < n)) {
      aux2 = aux1;
      aux1 = aux1->next;
   }

   if (head == aux1) {
      head = new_nodo;
   } else {
      aux2->next = new_nodo;
   }

   new_nodo->next = aux1;
}

void buscarLista(Nodo *head, int n) {
   bool flag = false;
   Nodo *actual = new Nodo();
   Nodo *address;
   actual = head;

   while ((actual != NULL) && (actual->data <= n)) {
      if (actual->data == n) {
         flag = true;
         address = actual;
      }
      actual = actual->next;
   }

   if (flag == true) {
      cout << "El elemento " << n << " ha sido encontrado en la posicion " << address << endl;
   } else {
      cout << "El elemento " << n << " no ha sido encontrado en la lista." << endl;
```

```
    }
}

bool existeEnLista(Nodo *head, int n) {
    Nodo *actual = head;
    while (actual != NULL) {
        if (actual->data == n) {
            return true;
        }
        actual = actual->next;
    }
    return false;
}

void eliminarNodo(Nodo *&head, int n){

    cout << "Ingrese un elemento a eliminar: " ;
    cin >> n;
    //PReguntar si la lista no esta vacia
    if(head != NULL) {
        Nodo*aux_delete;
        Nodo*previous = NULL;
        aux_delete = head;

        while ((aux_delete!= NULL) && (aux_delete->data!=n)){
            previous= aux_delete;
            aux_delete=aux_delete->next;
        }
        if(aux_delete==NULL){
            cout << "El elemento no se encuntro en la lista" << endl;
        }
        else if(previous == NULL){  //Elemento a eliminar es el primero de la lista
            head=head->next;
            delete aux_delete;
            cout << "El elemento se elimino correctamente." << endl;
        }
        else{ //El elemento a eliminar esta en otra posicion que no es la primera
            previous->next= aux_delete->next;
            delete aux_delete;
            cout << "El elemento se elimino correctamente." << endl;
        }
    }
}

void eliminarLista(Nodo*&head){
```

```
  while (head != NULL)
  {
    Nodo *aux_delete = head;
    head = aux_delete->next;
    delete aux_delete;
  }

}
```

Ejecucion:

```
Menu de opciones:
1. Insertar elemento
2. Mostrar lista
3. Buscar elemento
4. Eliminar nodo
5. Eliminar lista
6. Salir
Ingrese una opcion: 1
Ingrese el contenido de data: 7
Elemento insertado correctamente.
```

```
Menu de opciones:
1. Insertar elemento
2. Mostrar lista
3. Buscar elemento
4. Eliminar nodo
5. Eliminar lista
6. Salir
Ingrese una opcion: 2
Elementos de la son: |4| -> |6| -> |7| -> NULL
```

```
Menu de opciones:
1. Insertar elemento
2. Mostrar lista
3. Buscar elemento
4. Eliminar nodo
5. Eliminar lista
6. Salir
Ingrese una opcion: 3
Ingresa el elemento a buscar: 6
El elemento 6 ha sido encontrado en la posicion 0x257b1d569e0
```

```
Menu de opciones:
1. Insertar elemento
2. Mostrar lista
3. Buscar elemento
4. Eliminar nodo
5. Eliminar lista
6. Salir
Ingrese una opcion: 4
Ingrese un elemento a eliminar: 6
El elemento se elimino correctamente.
Elementos de la son: |4| -> |7| -> NULL
```