**Actividad 11 – Lista ligada**

**David Madrid Nápoles**

**Estructura de datos I**

# Lineamientos de evaluación

* El programa corre sin errores.
* Se implemento la clase ListaLigada con sus métodos:

ListaLigada();

~listaligada();

void push\_front(const T &dato);

void pop\_front();

void push\_back(const T &dato);

void pop\_back();

size\_t size();

void print();

* Se llevaron a cabo los procedimientos solicitados para realizar las capturas de pantalla como evidencia.

# Desarrollo

## Programa principal (salida main.exe)

Corriendo el main solicitado en la actividad:

Text

Description automatically generated

# Conclusiones

La manera de llevar las listas con ligas hacia el siguiente nodo se me hace interesante, ya que cada nodo contiene dos datos, su valor y algo que lo vincula a siguiente nodo. Y el uso de -> se me hace una manera muy interesante de como se lleva la implementación.

# Referencias

<https://www.youtube.com/watch?v=mmRfQxiP7b8>, Lista Ligada (I), Michel Davalos Boites.

<https://www.youtube.com/watch?v=wDDIH92zM90>, Lista Ligada (II), Michel Davalos Boites.

<https://www.youtube.com/watch?v=QcghZvKTFXA>, Lista Ligada (III), Michel Davalos Boites.

# Código

//main.cpp

#include <iostream>

#include "listaligada.h"

using namespace std;

int main() {

    ListaLigada<int> lista;

    lista.push\_front(10); *// insertar al inicio (frente)*

    lista.push\_front(0); *// insertar al inicio (frente)*

    lista.push\_front(4); *// insertar al inicio (frente)*

    lista.push\_back(7); *// insertar al final (cola)*

    lista.push\_back(8); *// insertar al final (cola)*

    cout << lista.size() << endl; *// imprime la cantidad de nodos (elementos)*

    lista.pop\_back(); *// elimina el ultimo (cola)*

    cout << lista.size() << endl; *// imprime la cantidad de nodos (elementos)*

    lista.print(); *// recorre la lista e imprime cada nodo (elemento)*

    lista.~ListaLigada(); *// se eliminan todos los nodos*

    cout << lista.size() << endl; *// imprime la cantidad de nodos (elementos)*

    return 0;

}

//listaligada.h

#ifndef LISTALIGADA\_H

#define LISTALIGADA\_H

#include <iostream>

using namespace std;

template <class T>

class ListaLigada

{

private:

    struct Nodo

    {

        T dato;

        Nodo \*sig;

        Nodo(const T &*dato*, Nodo \**sig* = nullptr):dato(*dato*), sig(*sig*) {}

    };

    Nodo \*head;

    size\_t cont;

public:

    ListaLigada();

    ~ListaLigada();

    void push\_front(const T &*dato*);

    void pop\_front();

    void push\_back(const T &*dato*);

    void pop\_back();

    size\_t size();

    void print();

};

template <class T>

ListaLigada<T>::ListaLigada()

{

    head = nullptr;

    cont = 0;

}

template <class T>

ListaLigada<T>::~ListaLigada()

{

    while (cont > 0)

    {

        pop\_front();

    }

}

template <class T>

void ListaLigada<T>::push\_front(const T &*dato*)

{

    Nodo \*nodo = new Nodo(*dato*, head);

    head = nodo;

    cont++;

}

template <class T>

void ListaLigada<T>::pop\_front()

{

    if (cont == 0)

    {

        cout << "Lista ligada vacia" << endl;

    }

    else

    {

        Nodo \*temp = head;

        head = head->sig;

        delete temp;

        cont--;

    }

}

template <class T>

void ListaLigada<T>::push\_back(const T &*dato*)

{

    if(cont == 0){

        push\_front(*dato*);

    }else {

        Nodo \*nodo = new Nodo(*dato*);

        Nodo \*temp = head;

        while(temp->sig != nullptr)

        {

            temp = temp->sig;

        }

        temp->sig = nodo;

        cont++;

    }

}

template <class T>

void ListaLigada<T>::pop\_back()

{

    if (cont == 0)

    {

        cout << "Lista ligada vacia" << endl;

    }

    else if(cont == 1)

    {

        pop\_front();

    }

    else {

        Nodo \*temp = head;

        while(temp->sig->sig != nullptr){

            temp = temp->sig;

        }

        delete temp->sig;

        temp->sig = nullptr;

        cont--;

    }

}

template <class T>

size\_t ListaLigada<T>::size() {

    return cont;

}

template <class T>

void ListaLigada<T>::print(){

    Nodo \*temp = head;

    while(temp != nullptr){

        cout << temp->dato << endl;

        temp = temp->sig;

    }

}

#endif