P2_PR1_A_28436_ROGERON_MATEO

Código del programa:

- Estudiante.h:

```
#pragma once

#include <iostream>
#include <string>

struct Estudiante
{
   std::string id;
   std::string nombre;
   int notaFinal;

   bool operator<(const Estudiante& otro)const
   {
      return notaFinal < otro.notaFinal;
   }

   bool operator>(const Estudiante& otro)const
   {
      return notaFinal > otro.notaFinal;
   }

   void imprimir()
   {
      std::cout << "[ID " << id <<", Nombre: "<<nombre<<", Nota: "<< notaFinal << "]\n";
   }

};
```

- ListaSimple:

```
#pragma once

#include "Estudiante.h"
#include <iomanip> // Para formato de tabla

using namespace std;

struct Nodo

{
   Estudiante dato;
   Nodo *siguiente;
   Nodo(const Estudiante &c) : dato(c), siguiente(nullptr) {}

};
```

```
class ListaSimple
{
private:
    Nodo *cabeza;
    bool insertarOrdenado(Nodo *& sortedHead, Nodo *llave)
    {
        if (!sortedHead || llave->dato < sortedHead->dato)
        {
            llave -> siguiente = sortedHead;
            sortedHead = llave;
            return true;
        }

        Nodo *current = sortedHead;
        Nodo *anterior = nullptr;

        while (current != nullptr && current->dato < llave->dato)
        {
            anterior = current;
            current = current->siguiente;
        }

        if(anterior == nullptr)
        {
            return false;
        }

        llave->siguiente = current;
        anterior->siguiente = llave;

        cout << " -> Inserción de la posición: ";
        if(current)
        {
            current->dato.imprimir();
        }
        else
        {
            cout<<" FINAL \n";
        }
        return true;
    }

public:

    ListaSimple() : cabeza(nullptr) {}
    ~ListaSimple()
    {
        // Liberación de memoria (Destructor)
        Nodo *current = cabeza;
        while (current != nullptr)
        {
```

```
            Nodo *siguiente = current->siguiente;
            delete current;
            current = siguiente;
        }
        cabeza = nullptr;
    }

    void insertar(string id, string nombre, int nota)
    {
        Estudiante c= {id,nombre,nota};
        Nodo *nuevoNodo = new Nodo(c);
        nuevoNodo->siguiente = cabeza;
        cabeza = nuevoNodo;
    }

    void mostrar() const
    {
        Nodo *current = cabeza;
        if (!current)
        {
            std::cout << "Lista vacía.\n";
            return;
        }
        while (current != nullptr)
        {
            current->dato.imprimir();
            if (current->siguiente)
            {
                std::cout << " -> ";
            }
            current = current->siguiente;
        }
        std::cout << " -> NULL\n";
    }

    void insertionSort()
    {

        if (!cabeza || !cabeza->siguiente)
        {
            return;
        }

        Nodo *sorted = nullptr;
        Nodo *current = cabeza;

        std::cout <<
"\n=====================================================\n";
        std::cout << "       INICIO DE ORDENAMIENTO (INSERTION SORT)       \n";
        std::cout << "=====================================================\n";

        while(current != nullptr)
```

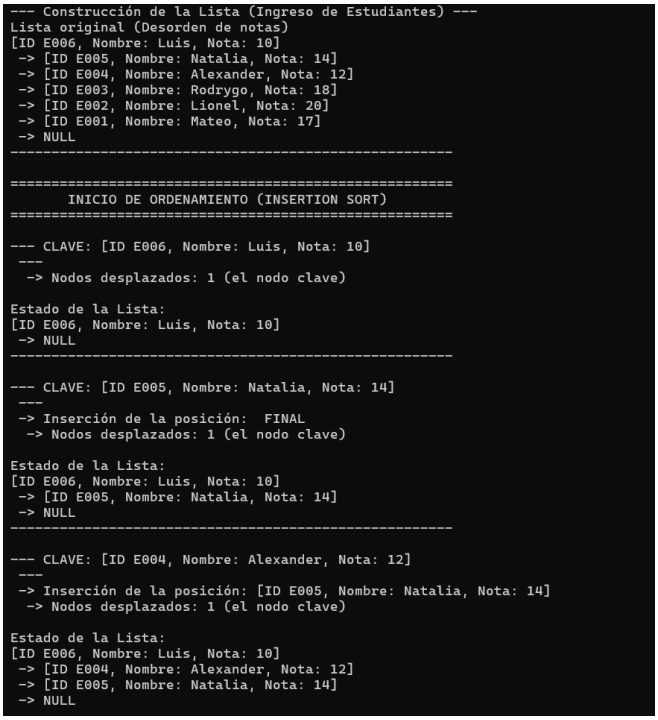
```
        {
            Nodo *key = current;
            current = current->siguiente;
            key->siguiente = nullptr;

            std::cout << "\n--- CLAVE: ";
            key->dato.imprimir();
            std::cout << " ---\n";
            if (insertarOrdenado(sorted, key))
            {
                std::cout << "  -> Nodos desplazados: 1 (el nodo clave)\n";
            }
            else
            {
                std::cout << "  -> Nodos desplazados: 0 (el nodo clave ya estaba en orden)\n";
            }

            cabeza = sorted;

            std::cout << "\nEstado de la Lista: \n";
            mostrar();
            std::cout << "-----------------------------------------------------\n";
        }
    }
};
```

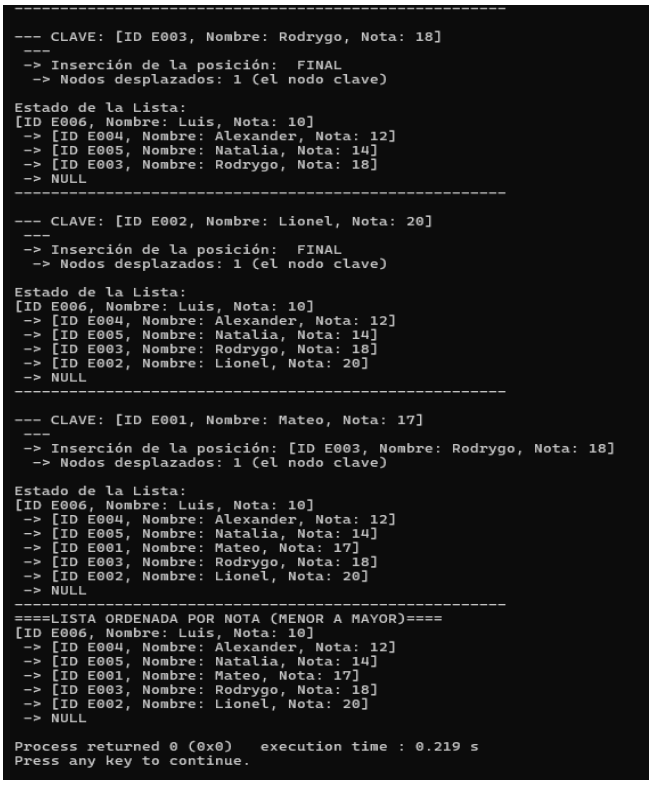
- main.cpp

```
#include <iostream>
#include <locale.h>
#include "ListaSimple.h"

using namespace std;

int main()
{
    setlocale(LC_CTYPE, "Spanish");

    ListaSimple estudiante;

    cout << "--- Construcción de la Lista (Ingreso de Estudiantes) ---\n";

    estudiante.insertar("E001","Mateo",17);
    estudiante.insertar("E002","Lionel",20);
    estudiante.insertar("E003","Rodrygo",18);
    estudiante.insertar("E004","Alexander",12);
    estudiante.insertar("E005","Natalia",14);
    estudiante.insertar("E006","Luis",10);

    cout << "Lista original (Desorden de notas)\n";
    estudiante.mostrar();
    cout << "-----------------------------------------------------\n";
```

```
    estudiante.insertionSort();

    cout <<"====LISTA ORDENADA POR NOTA (MENOR A MAYOR)====\n";
    estudiante.mostrar();
    return 0;
}
```

Capturas del funcionamiento:





EXPLICACIÓN DEL FUNCIONAMIENTO DE SU SOLUCIÓN:

El funcionamiento de ordenamiento se ha hecho desde el InsertionSort, utilizando el apartado de ListaSimple al ser un ordenamiento básico. (En el enunciado también se especifica que: “...tal como se hace cuando una persona ordena cartas en su mano”, siendo una referencia al uso de Insertion Sort)

La razón de la Lista Simple es para la implementación del ordenamiento mediante los punteros, esto para evitar que la memoria utilice demasiados recursos mediante el movimiento físico de un vector, trabajando con una curva de complejidad O(1).

De igual forma optimizaríamos de mejor manera el uso del puntero pues lo único que hacemos es cambiar el sentido en donde la dirección del nodo “***siguiente”*** y “***anterior”*** cambian. Así evitamos mover el dato desde donde está y solo le decimos al programa que ordene mediante sus Nodos.

Los datos fueron quemados para una exposición más simple con respecto al programa, además de un poco más rápida por el tiempo empleado.