CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Estructuras de datos

Ordenamientos

Prof.: Noe Ortega Sanchez

Alumno: Edgar Antonio Delgadillo Villegas

Carrera: Ingeniería Informática

Materia: i5886 Estructura de datos I

NRC: 182881

Sección: D17

Calendario: 2021A

"Ordenamientos"

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, entre otras cosas, la principal elaboración de este programa es para mostrar los métodos de ordenamiento quicksort y select sort.

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.

Main

```
int main()
{

    int opc, opc1, pos, id;
    float precio;
    string pro;
    Nodo *inicio, *fin;
    Producto Producto;
    listaD *lista = new listaD();
    do
    {
        system("cls");
        cout<<"--------------------------------------------------------------------------------"<<endl;
        cout<<"                                MENU                                "<<endl;
        cout<<endl<<"--------------------------------------------------------------------------------"<<endl;
        cout <<"1) Insertar"<<endl;
        cout<<"2) Eliminar"<<endl;
        cout<<"3) Buscar"<<endl;
        cout<<"4) Vacia"<<endl;
        cout<<"5) Primero"<<endl;
        cout<<"6) Ultimo"<<endl;
        cout<<"7) Anterior"<<endl;
        cout<<"8) Siguiente"<<endl;
        cout<<"9) Tamanio"<<endl;
        cout<<"10) vaciar"<<endl;
```

```
cout<< 9) tamanio <<endl;
cout<<"10) vaciar"<<endl;
cout<<"11) Select sort"<<endl;
cout<<"12) Quick sort"<<endl;
cout<<"13) Mostrar lista"<<endl;
cout<<"14) Salir"<<endl<<endl;
cout<<"Opcion: ";
cin >> opc;
switch(opc)
{
case 1:
    {
        cout << endl;
        cout << "Opcion a insertar"<<endl;
        cout<<"1) Inicio"<<endl;
        cout<<"2) Final"<<endl<<endl;
        cout<<"Opcion: ";
        cin >> opc1;
        if(opc1==1)
        {
            cout << endl;
            cout << "Producto: ";
            cin.ignore();
            getline(cin, pro);
            cout << "ID: ";
            cin>>id;
            cout << "Precio: ";
            cin>>precio;
            Producto.setProductos(pro);
            Producto.setId(id);
            Producto.setPrecio(precio);
            cout << endl;
            lista->insertaInicio(Producto);
            lista->mostrarTodo();
        }
        else if(opc1==2)
        {
            cout << endl;
            cout << "Producto: ";
            cin.ignore();
            getline(cin, pro);
            cout << "ID: ";
            cin>>id;
            cout << "Precio: ";
            cin>>precio;
            Producto.setProductos(pro);
            Producto.setPrecio(precio);
            Producto.setId(id);
            cout << endl;
            lista->insertaFinal(Producto);
            lista->mostrarTodo();
        }
        else
        {
            cout << endl;
            cout << "No existe opcion...." << endl << endl;
        }
    break;
    }
```

```
            cout << endl;
            cout << "No existe opcion...." << endl << endl;
        }
    break;
    }
case 2:
    {
        cout << endl;
        cout << "¿Eliminar:    ";
        cin.ignore();
        getline(cin, pro);
        cout << endl;
        lista->eliminar(pro);
        lista->mostrarTodo();
    break;
    }
case 3:
    {
        cout << endl;
        cout << "Producto:   ";
        cin.ignore();
        getline(cin, pro);
        cout << endl;
        lista->Buscar(pro);
        cout << endl;
    break;
    }
case 4:
    {
```

```
case 4:
    {
        cout << endl;
        if(lista->vacia())
            cout << "Lista vacia" << endl;
        else
            cout << "Lista vacia" << endl;
    cout << endl;
    break;
    }
case 5:
    {
        cout << endl;
        inicio=lista->primero();
        if(inicio)
            cout << ", Precio: " << inicio->dato.getPrecio() << ", ID: " << inicio->dato.getId()<< "Producto: " << inicio->dato.getProductos()  << " " << en
        cout << endl;
    break;
    }
case 6:
    {
        cout << endl;
        fin=lista->ultimo();
        if(fin)
            cout  << ", Precio: $" << fin->dato.getPrecio() << ", ID: " << fin->dato.getId()<< "Producto: " << fin->dato.getProductos() << " " << endl;
        cout << endl;
    break;
    }
case 7:
```

```
        case 8:
            {
                cout << endl;
                cout << "Deme el nombre de un producto: ";
                cin.ignore();
                getline(cin, pro);
                cout << endl;
                lista->siguiente(pro);
                cout << endl;
            break;
            }
        case 9:
            {
                cout << endl;
                lista->tamano();
                cout << endl;
            break;
            }
        case 10:
            {
                cout << endl;
                lista->eliminarTodo();
                cout << endl;
            break;
            }
        case 11:
            {
                cout << endl;
                lista->select();
            break;

                lista->select();
            break;
            }
        case 12:
            {
                 cout << endl;
                 lista->quick(lista->primero(),lista->ultimo());
            break;
            }
        case 13:
            {
                system("cls");
                 cout << endl;
                 lista->mostrarTodo();
                    system("pause");
            break;
            }
        case 14:
            {
                cout << endl;
            break;
            }
        default:
            cout << endl << "Opcion no existe....." << endl << endl;
        }
    }while(opc!=14);
    return 0;
}
```

Nodo.h

Aquí en el nodo 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 para la Pila ya que es una lista simple y anterior y siguiente para la cola ya que es una lista doble por último.

```
using namespace std;

class listaD
{
    public:
        listaD();
        ~listaD();
        void insertaPos(Producto, int);
        void eliminar(string);
        void Buscar(string);
        void insertaInicio(Producto);
        void quick(Nodo*, Nodo*);
        void select();
        void insertaFinal(Producto);
        bool vacia();
        Nodo* anterior(string);
        Nodo* siguiente(string);
        int tamano();
        void Inicializa();
        Nodo* primero();
        Nodo* ultimo();
```

```
        Nodo* primero();
        Nodo* ultimo();
        void eliminarTodo();
        void mostrarTodo();
        Nodo* particion(Nodo*, Nodo*);
        void intercambio(Nodo*, Nodo*);


    private:
        Nodo *h;
        Nodo *t;
};

#endif // lista_H
```

Nodo.cpp

```
#include "Nodo.h"

Nodo::Nodo()
{
    sig=nullptr;
    ant=nullptr;
    //ctor
}

Nodo::~Nodo()
{
    //dtor
}
```

Lista

en lista.h, en lista.cpp se muestra que métodos utilice tanto el quicksort, como el selectsort, vaciar, mostrar, tope 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.h y lista.cpp

```
#include "lista.h"
#include "Producto.h"
#include <iostream>

using namespace std;



void listaD::insertaPos(Producto pro, int pos)
{
    Nodo *tmp = new Nodo();
    tmp->dato = pro;
    Nodo *aux = h;
    bool band = true;
    int i = 1;
    if(vacia())
    {
        h=tmp;
        t=tmp;
    }
    else
```

```
{
    while(aux && band)
    {
        if(i == pos)
        {
            band = false;
        }
        else
        {
            aux = aux->sig;
        }
        i++;
    }
    if(aux==nullptr)
    {
        t->sig=tmp;
        tmp->ant=t;
        t=tmp;
        cout << "Posicion no valida, se ha agregado al final." << endl << endl;
    }
    else if(aux->ant==nullptr)

         cout << "Posicion no valida, se ha agregado al final." << endl << endl;
     }
     else if(aux->ant==nullptr)
     {
         tmp->sig=h;
         h->ant=tmp;
         h=tmp;
     }
     else
     {
         tmp->sig=aux->ant->sig;
         aux->ant->sig=tmp;
         tmp->ant=aux->ant;
         aux->ant=tmp;
     }
  }
}
```

```
void listaD::insertaInicio(Producto pro)
{
    Nodo *tmp = new Nodo();
    tmp->dato = pro;
    if (vacia())
    {
        h = tmp;
        t = tmp;
    }
    else
    {
        tmp->sig = h;
        h->ant = tmp;
        h = tmp;
    }
}
```

```
Nodo* listaD::siguiente(string pro)
{
    Nodo *aux = h;
    bool band=true;
    if(h)
    {
        while(aux && band)
        {
            if(aux->dato.getProductos()!=pro)
            {
                aux=aux->sig;
            }
            else
            {
                band=false;
            }
        }
        if(aux==nullptr)
        {
            cout << "Producto no existe" << endl;
```

```
void listaD::insertaFinal(Producto pro)
{
    Nodo *tmp = new Nodo();
    tmp->dato = pro;
    if(vacia())
    {
        h = tmp;
        t = tmp;
    }
    else
    {
        t->sig = tmp;
        tmp->ant = t;
        t = tmp;
    }
}

void listaD::eliminar(string pro)
{
    Nodo *aux=h;
    bool band=true;

    if(aux)
    {
        while(aux && band)
        {
            if(aux->dato.getProductos()==pro)
            {
                band=false;
            }
            else
            {
                aux=aux->sig;
            }
        }
        if((aux==h)and(aux==t))
        {
```

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

    }
```

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

void listaD::Inicializa()
{
    h = nullptr;
    t = nullptr;
}

Nodo* listaD::primero()
{
    Nodo *primero=new Nodo();
    if(vacia())
    {
        cout << "Lista vacia" << endl;
        return nullptr;
    }
    else
        primero=h;
        return primero;
```

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

void listaD::select()
{
    Nodo *tmp,*tmp2=new Nodo();
    Producto aux;
    if(vacia())
    {
        cout<<"La lista esta vacia"<<endl;
    }
    else
    {
        tmp=h;
        while(tmp)
        {
            tmp2=tmp->sig;
            while(tmp2)
            {
                if ((tmp->dato.getId())>(tmp2->dato.getId()))
                {
```

```
    if(vacia())
    {
        cout<<"La lista esta vacia"<<endl;
    }
    else
    {
        tmp=h;
        while(tmp)
        {
            tmp2=tmp->sig;
            while(tmp2)
            {
                if ((tmp->dato.getId())>(tmp2->dato.getId()))
                {
                    aux = tmp2->dato;
                    tmp2->dato = tmp->dato;
                    tmp->dato = aux;
                }
                tmp2=tmp2->sig;
            }
            tmp=tmp->sig;
        }
    }
}
```

```
Nodo* listaD::particion(Nodo *cabeza, Nodo *cola)
{
    Nodo *pivote = cola;
    Nodo *i;
    i=cabeza->ant;
    Nodo *j=cabeza;
    while(j!=cola)
    {
        if (j->dato.getId() <= pivote->dato.getId())
        {
            if(i==nullptr)
            {
                i=cabeza;
            }
            else
            {
                i=i->sig;
            }
        }
        j = j->sig;
    }
    if(i==nullptr)
    {
        i=cabeza;
    }
    else
    {
        i=i->sig;
    }


    }
    if(i==nullptr)
    {
        i=cabeza;
    }
    else
    {
        i=i->sig;
    }
    intercambio(i, pivote);
    return i;
}

void listaD::quick(Nodo *cabeza, Nodo *cola)
{
    if (cola != nullptr && cabeza != cola && cabeza != cola->sig)
    {
        Nodo *p = particion(cabeza, cola);
        quick(cabeza, p->ant);
        quick(p->sig, cola);
    }

}

listaD::listaD()
{
    Inicializa();
    //ctor
}

listaD::~listaD()
{
```

```
using namespace std;

class listaD
{
    public:
        listaD();
        ~listaD();
        void insertaPos(Producto, int);
        void eliminar(string);
        void Buscar(string);
        void insertaInicio(Producto);
        void quick(Nodo*, Nodo*);
        void select();
        void insertaFinal(Producto);
        bool vacia();
        Nodo* anterior(string);
        Nodo* siguiente(string);
        int tamano();
        void Inicializa();
        Nodo* primero();
        int tamano();
        void Inicializa();
        Nodo* primero();
        Nodo* ultimo();
        void eliminarTodo();
        void mostrarTodo();
        Nodo* particion(Nodo*, Nodo*);
        void intercambio(Nodo*, Nodo*);


    private:
        Nodo *h;
        Nodo *t;
};
```

Producto.h y producto.cpp

Aquí se muestra como use la clase libro para indicar o guardar el dato como si fuese un producto, con sus respectivos nodos dependiendo si es cola o pila.

```
#include "Producto.h"

Producto::Producto()
{
    Productos="";
    precio=0;
    id=0;
    //ctor
}

Producto::~Producto()
{
    //dtor
}
```

```
class Producto
{
    public:
        void setId(int numId) {id=numId;}
        void setPrecio(float numPrecio) {precio=numPrecio;}
        int getId(void) {return id;}
        float getPrecio(void) {return precio;}
        string getProductos(void){return Productos;}
        Producto(string,float,int);
        Producto();
        ~Producto();
        void setProductos(string nom) {Productos=nom;}

    private:
        float precio;
        int id;
        string Productos;

};
```

Compilando el Programa

Solo mostrare como utilice los métodos quicksort y select sort ya que esos son los objetivos de esta tarea

```
--------------------------------------------------------------------------------
                                    MENU
--------------------------------------------------------------------------------

1) Insertar
2) Eliminar
3) Buscar
4) Vacia
5) Primero
6) Ultimo
7) Anterior
8) Siguiente
9) Tamanio
10) vaciar
11) Select sort
12) Quick sort
13) Mostrar lista
14) Salir

Opcion: 13
```

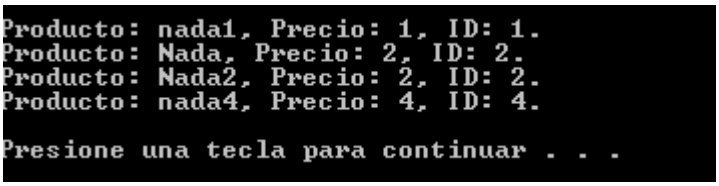
```
Producto: Nada2, Precio: 2, ID: 2.
Producto: Nada, Precio: 2, ID: 2.
Producto: nada1, Precio: 1, ID: 1.
```

```
--------------------------------------------------------------------------------
                                    MENU
--------------------------------------------------------------------------------

1) Insertar
2) Eliminar
3) Buscar
4) Vacia
5) Primero
6) Ultimo
7) Anterior
8) Siguiente
9) Tamanio
10) vaciar
11) Select sort
12) Quick sort
13) Mostrar lista
14) Salir

Opcion: 12
```

```
Producto: nada1, Precio: 1, ID: 1.
Producto: Nada, Precio: 2, ID: 2.
Producto: Nada2, Precio: 2, ID: 2.
```

Arriba se muestra el quick sort y abajo mostrare la continuación del ordenamiento pero con select sort

```
Producto: nada4, Precio: 4, ID: 4.
Producto: nada1, Precio: 1, ID: 1.
Producto: Nada, Precio: 2, ID: 2.
Producto: Nada2, Precio: 2, ID: 2.

Presione una tecla para continuar . . .
```



Opinión;

Este programa fue algo un poco complicado para mí por lo menos al momento de entender los ordenamientos pero poco a poco y volviendo a ver los videos de usted profesor me ayudaron aun que siento que necesito ver más videos de estos ordenamientos y como funciona, aunque algo que si me gusto de la clase es ver cómo nos comentó y enseño los videos de gente bailando con ordenamientos fue algo divertido y que jamás me hubiera imaginado que alguien haría.