UNIVERSIDAD AUTONOMA GABRIEL RENE MORENO

FACULTAD DE INGENIERIA Y CIENCIAS DE LA COMPUTACION Y TELECOMUNICACIONES

ESTRUCTURA DE DATOS II

CONTENIDO:

LAB-6. ELIMINAR NODOS, LISTAS DOBLES

PORCENTAJE TERMINADO: 100%

Grupo 14

Garcia Taboada Brayan Albaro

Fecha de presentación : Jueves , 11 de abril de 2024

Fecha Presentada : Jueves , 11 de abril de 2024

Días de Atraso : 0

ELIMINAR LOS ELEMENTOS DE UNA LISTA DOBLEMENTE ENCADENADA

TRABAJO INDIVIDUAL.

1. L1.eliminarPrim() : Método que elimina el elemento de la primer posición.

2. L1.eliminarUlt() : Método que elimina el último elemento de la lista L1.

3. L1.eliminarNodo(ap, p) : Método que elimina el nodo p, y devuelve el nodo el nodos siguiente a ap. El nodo p, puede estar al principio, final o al centro de la lista.

4. L1.eliminarTodo(x) : Método que elimina todos los elementos x de la lista L1.

5. L1.eliminarPrim(n) : Método que eliminar los primeros n-elementos de la lista L1.

6. L1.eliminarUlt(n) : Método que elimina los n-últimos elementos de la lista L1.

7. L1.eliminarIesimo(i) : Método que elimina el i-ésimo elemento de la lista L1.

En realidad esto de las listas encadenadas dobles es muy interesante en especial su utilidad y practicidad por lo que veo, tuve alugnas dificultades para entender unas partes pero al final termine el trabajo sabiendo mas cada vez

```
//1. L1.eliminarPrim() : Método que elimina el elemento de la primer posición.
public void eliminarPrim() {
    if (vacia()) {
        return;
    }
    if (prim == ult) {
        prim = ult = null;
    } else {
        prim.prox.ant = null;
        prim = prim.prox;
    }
    cantElem--;
}
//2. L1.eliminarUlt() : Método que elimina el último elemento de la lista L1.
```

```
public void eliminarUlt() {
    if (vacia()) {
        return;
    }
    if (prim == ult) {
        prim = ult = null;
    } else {
        ult.ant.prox = null;
        ult = ult.ant;
    }
    cantElem--;
}
```

// 3. L1.eliminarNodo(ap, p) : Método que elimina el nodo p, y devuelve el nodo el nodos siguiente a ap. El nodo p, puede estar al principio, final o al centro de la lista.

```
public Nodo eliminarNodo(Nodo ap, Nodo p) {
    if (p == null) {
        return null;
    }
    if (ap == null) {
        eliminarPrim();
        return prim;
    }
    if (p.prox == null) {
        eliminarUlt();
        return null;
    }
    ap.prox = p.prox;
    p.prox.ant = ap;
    cantElem--;
    return ap.prox;
```

```
    }

    //4. L1.eliminarTodo( x ) : Método que elimina todos los elementos x de la lista L1.
    public void eliminarTodo(int x) {
        Nodo p = prim, ap = null;
        while (p != null) {
            if (p.elem == x) {
                ap.prox = eliminarNodo(ap, p);
                p = p.prox;
            } else {
                ap = p;
                p = p.prox;
            }
        }

    }
//5. L1.eliminarPrim( n ) : Método que eliminar los primeros n-elementos de la lista L1.

    public void eliminarPrim(int n) {
        for (int i = 0; i < n; i++) {
            eliminarPrim();
        }
    }
//6. L1.eliminarUlt( n ) : Método que elimina los n-últimos elementos de la lista L1.

    public void eliminarUlt(int n) {
        for (int i = 0; i < n; i++) {
            eliminarUlt();
        }
    }
```

//7. L1.eliminarIesimo(i) : Método que elimina el i-ésimo elemento de la lista L1.

```
public void eliminarIesimo(int i) {
    int k = 0;
    Nodo p = prim, ap = null;
    while (k < i && p != null) {
        ap = p;
        p = p.prox;
        k += 1;
    }
    eliminarNodo(ap, p);
}
```