UNIVERSIDAD AUTONOMA GABRIEL RENE MORENO

FACULTAD DE INGENIERIA Y CIENCIAS DE LA COMPUTACION Y TELECOMUNICACIONES

ESTRUCTURA DE DATOS 2

CONTENIDO: LAB-5. LISTAS DOBLEMENTE ENCADENADAS..

PORCENTAJE TERMINADO: 100%

INTEGRANTES	GRUPO	HG	HI	EVAL
Flores Veizaga Eudenia Gandira	14	1	1	100
Garcia Taboada Brayan Albaro	14	1	1	100

Fecha de presentación : Martes, 09 de Abril de 2024

Fecha Presentada : Martes, 09 de Abril de 2024

Días de Atraso : 0

```
/**
 *
 * @author braya
 */
public class Lista {
   public Nodo prim;
   public Nodo ult;
   public int cantElem;

   public Lista(){
      prim=ult=null;
      cantElem=0;
   }
   public String toString(){
      String s1="[";
      Nodo p=prim;
      while(p!=null){
         s1+=p.elem;
         if(p.prox!=null){
            s1+=", ";
         }
         p=p.prox;
      }
   return s1+"]";
   }
   //1.L1.insertarIesimo(x, i) : Método que inserta el elemento x, en la posición i, de la lista L1.
   public void insertarIesimo(int x,int i){
      int k=0;
      Nodo p=prim,ap=null;
      while(k<i && p!=null){
         ap=p;
```

```
        p=p.prox;
        k=k+1;
      }
      insertarNodo(ap,p,x);
   }
   //2.L1.insertarPrim(x) : Método que insertar el elemento x, al inicio de la lista L1.
   public void insertarPrim(int x){
      if (vacia()) prim=ult=new Nodo(null,x,null);
      else
         prim=prim.ant=new Nodo(null,x,prim);
      cantElem++;
   }
   //3.L1.insertarUlt(x) : Método que inserta el elemento x, al final de la lista L1.
   public void insertarUlt(int x){
      if(vacia()) prim=ult=new Nodo(null,x,null);
      else
         ult=ult.prox=new Nodo(ult,x,null);
      cantElem++;
   }
   //4. L1.insertarLugarAsc(x) : Método que inserta el elemento x, en su lugar correspondiente en la Lista ordenadas de menor a mayor.
   public void insertarLugarAsc(int x){
      Nodo p=prim,ap=null;
      while(p!=null && x>p.elem){
         ap=p;
         p=p.prox;
      }
      insertarNodo(ap,p,x);
   }

   //5. L1.insertarLugarDes(x) : Método que inserta el elemento x, en su lugar correspondiente en la Lista ordenadas de mayor a menor.
```

```
public void insertarLugarDes(int x){
    Nodo p=prim,ap=null;
    while(p!=null && x<p.elem){
        ap=p;
        p=p.prox;
    }
    insertarNodo(ap,p,x);
}
```

//6. L1.insertarIesimo(L2, i) : Método que insertar los elementos de la lista L2 en la lista L1, desde la posición i.

```
public void insertarIesimo(Lista L2,int i){
    for (int j = 0; j < L2.cantElem; j++) {
        insertarIesimo(L2.obtener(j),j+i);
    }

}
public int obtener(int i){
    Nodo p=prim;
    while (i>0){
        p=p.prox;
        i--;
    }
    return p.elem;
}
```

//7. L1.insertarPrim(L2) : Método que insertar los elementos de la lista L2 al principio de la lista L1.

```
public void insertarPrimero(Lista L2){
    for (int j = 0; j < L2.cantElem; j++) {
        insertarPrim(L2.obtener(j));
    }

}
```

```
   //8. L1.insertarUlt(L2) : Método que insertar los elementos de la lista L2 al final de la lista L1.
   public void insertarUlt(Lista L2){
      for (int j = 0; j < L2.cantElem; j++) {
         insertarUlt(L2.obtener(j));
      }

   }
   //9. L1.iguales() : Método Lógico que devuelve True, si todos los elementos de la lista L1 son iguales.
public boolean iguales(){
   Nodo p=prim,pp=prim.prox;
   while(pp!=null){
      if(p.elem!=pp.elem){
         return false;
      }
   }
   return true;
}
   //10. L1.diferentes() : Método Lógico que devuelve True, si todos los elementos de la lista L1 son diferentes.
public boolean diferentes(){ //123456
   Nodo p=prim,pp=prim.prox;
   for (int i = 0; i < cantElem; i++) {
      for (int j = 0; j < cantElem; j++) {
         if(p.elem==pp.elem){
            return false;
         }
         pp=pp.prox;
      }
      p=p.prox;
   }
   return true;
```

```
}
   //11. L1.mayorElem() : Método que devuelve el mayor elemento de la lista L1.
public int mayorElem(){
   Nodo p=prim.prox;
   int may=p.elem;
   while(p!=null){
      if(p.elem>=may){
         may=p.elem;
      }
      p=p.prox;
   }
   return may;
}
   //12. L1.ordenado()  : Método Lógico que devuelve True, si todos los elementos de la lista L1 están ordenados en forma ascendente o descendente.
public boolean ordenado(){//11233
   Nodo p=prim;
   while(p.elem==p.prox.elem){
      p=p.prox;
   }

   if(p.elem<p.prox.elem){
      p=p.prox;
      while (p!=null) {
         if(p.elem>p.prox.elem )
            return false;
         p=p.prox;
      }
   }else if(p.elem>p.prox.elem){
      p=p.prox;
      while (p!=null) {
```

```
        if(p.elem<p.prox.elem )
            return false;
        p=p.prox;
      }
   }
   return true;
}
```

//13. L1.indexOf(x) : Método que devuelve la posición (lugar) de la primera ocurrencia del elemento x. Si x no se encuentra en la lista L1, el método devuelve –

```
   public int indexOf(int x){
      Nodo p=prim;
      int pos=0;
      while(p!=null){
         if (p.elem==x)
            return pos;
         p=p.prox;
         pos++;
      }
      return -1;
   }
```

//14. L1.lastIndexOf(x) : Método que devuelve la posición (lugar) de la última ocurrencia del elemento x. Si x no se encuentra en la lista L1, el método devuelve –

```
   public int lastIndexOf(int x){
      Nodo p=ult;
      int pos=cantElem-1;
      while(p!=null){
         if (p.elem==x) return pos;
         pos--;
         p=p.ant;
      }
      return -1;
   }
```

```
   //15. L1.palindrome() : Método lógico que devuelve True, si la lista L1 contiene elementos
que forma un palíndrome.
      public boolean palindrome(){
      Nodo p=prim,u=ult;
      for (int i =0; i<=(cantElem/ 2);i++){
         if (p.elem!=u.elem) return false;
         p=p.prox;
         u=u.ant;
      }
      return true;
      }

   public void insertarNodo(Nodo ap,Nodo p,int x){
      if (ap==null) insertarPrim(x);
      else
      if(p==null) insertarUlt(x);
      else {
         ap.prox=p.ant =new Nodo(ap,x,p);
         cantElem++;
      }
   }

   public void insertarLugar(int x){
      Nodo p=prim,ap=null;
      while(p!=null && x>p.elem){
         ap=p;
         p=p.prox;
      }
      insertarNodo(ap,p,x);
   }
}
```

```
class Nodo {
   public Nodo ant;
   public int elem;
   public Nodo prox;
   public Nodo (Nodo ant,int elem, Nodo prox) {
      this.ant=ant;
      this.elem=elem;
      this.prox=prox;

   }
}
```

COMENTARIOS

En el trabajo del día de hoy aprendimos mas sobre el uso de listas, hay algunos ejercicios interesantes, pero la lógica con los ejercicios ya resueltos es buena y mejorando cada vez más.