Ejercicio 1. Implementamos el TAD secuencia sobre una lista simplemente enlazada usando:

```
modulo ListaEnlazada<T> implementa Secuencia<T>{
      var primero: Nodo
      var ultimo: Nodo
      var longitud: ℤ
         ...
} Escriba los algoritmos para los siguientes procs y calcule su complejidad:
```

- nuevaListaVacia(): ListaEnlazada<T>
- agregarAdelante(inout l: ListaEnlazada<T>, in t: T)
- agregarAtras(inout l: ListaEnlazada<T>, in t: T)
- eliminar(inout l: ListaEnlazada<T>, in i: int)
- pertenece(in l: ListaEnlazada<T>, in t: T): Bool
- obtener(in l: ListaEnlazada<T>, in i: int): T
- concatenar(inout l1: ListaEnlazada<T>, in l2: ListaEnlazada<T>)
- sublista(in l:ListaEnlazada<T>, in inicio: int, in final: int): ListaEnlazada<T>

```
modulo ListaEnlazada<T> implementa Secuencia<T>{
      var primero: Nodo
      var ultimo: Nodo
      var longitud: ℤ
      impl nuevaListaVacia():ListaEnlazada<T>{
            primero:=null;
            ultimo:=null;
            longitud:=0;
         }
```

Complejidad O(1)

```
      impl agregarAdelante(inout l: ListaEnlazada<T>, in t: T):{
            nodo := nuevo Nodo
            nodo.siguientes=primero
            primero=nodo;
         }
```

Complejidad O(1)

```
      impl eliminar(inout l: ListaEnlazada<T>, i: int):{
1:          Nodo actual ← l.primero
2:          if i = 0 then
3:          |  l.primero=l.primero.siguiente
4:          else if i = l.length − 1 then
5:          |  l.ultimo=l.ultimo.anterior
6:          else
7:          |  int j ← 0
8:          |
            |  while j < i − 1 do        ←O(i − 1)
9:          |  |  actual ← actual.siguiente
10:         |  |  j + +
11:         |  actual.siguiente = actual.siguiente.siguiente
```

Complejidad O(i)

```
         }
      impl pertenece(in l: ListaEnlazada<T>, in elem: T):Bool{
            j=0
            actual=l.primero
            res=false

            while (j < l.longitud and res = false) do
               if actual.valor==elem then
```

```
            res=true
        else
            j++
        end if
    end while
    return res
}
```

```
    impl obtener(in l: ListaEnlazada<T>, in i: int):T{
          j=0
          actual=l.primero

          while j! = i do
             j++
          end while
          return actual.valor
       }
    impl concatenar(inout l1: ListaEnlazada<T>,in l2: ListaEnlazada<T>):{
          l1.ultimo.siguiente=l2.primero
       }
    impl sublista(in l: ListaEnlazada<T>, in inicio: int, in final: int):ListaEnlazada<T>{
          res= nueva ListaEnlazada<T>
          j=0
          actual=l.primero
          anterior=null
          copiar=false
          while j < final do
             copiar= j >= inicio
             if copiar then
                nodo = nuevo Nodo
                nodo.valor=actual.valor
                if j==inicio then
                   res.primero=nodo
                   res.final=nodo
                   break
                else
                   res.final.siguiente=nodo
                   res.final=nodo
                end if
             else
                nothing
             end if
             j++
          end while
       }
}
```