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;
         }
      //nodo tiene las variables nodo.valor (T) y nodo.siguiente (Nodo)
      impl agregarAdelante(inout l: ListaEnlazada<T>, in t: T):{
            nodo ← nuevo Nodo
            nodo.siguiente=primero
            nodo.valor=t
            primero=nodo;
         }
      impl agregarAtras(inout l: ListaEnlazada<T>, in t: T):{
            nodo ← nuevo Nodo
            nodo.siguiente=null
            nodo.valor=t
            ultimo.siguiente=nodo
            ultimo=nodo
         }
      impl eliminar(inout l: ListaEnlazada<T>, in i:Int):{
            j=0 actual= primero
            while j < i − 1 do
               actual=actual.siguiente
               j++
            end while  actual.siguiente=actual.siguiente.siguiente;
         }
}
```