



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Guia 6

2do cuatrimestre 2024

Algoritmos y Estructuras de Datos

Integrante	LU	Correo electrónico
Prieto, Nahuel	646/20	enprieto@dc.uba.ar



Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (++54 +11) 4576-3300

<http://www.exactas.uba.ar>

Ejercicio 1

Ejercicio 1. Implementamos el TAD Secuencia sobre una **lista simplemente enlazada** usando

```
NodoLista<T> es struct<
    valor: T,
    siguiente: NodoLista<T>,
>

Módulo ListaEnlazada<T> implementa Secuencia<T> {
    var primero: NodoLista<T> // "puntero" al primer elemento
    var último: NodoLista<T> // "puntero" al primer elemento
    var longitud: int // cantidad total de elementos

    proc NuevaListaVacía ( ): ListaEnlazada<T>
        res := new ListaEnlazada<T>
        res.primero := null
        res.último := null
        res.longitud := 0
        return res

    proc AgregarAdelante (inout l: ListaEnlazada<T>, in e: T):
        nodo := new NodoLista<T>
        nodo.value := e
        nodo.siguiente := null

        if l.longitud == 0 then
            l.primero := nodo
            l.último := nodo
        else
            nodo.siguiente := l.primero
            l.primero := nodo
        end if
        l.longitud := l.longitud + 1

    proc Pertenece (in l: ListaEnlazada<T>, in e: T): bool
        res := false
        actual := l.primero
        while actual ≠ null do
            if actual.valor == e then
                res := true
            end if
            actual := actual.siguiente
        end while
        return res
}
```

- Escriba los algoritmos para los siguientes procs y calcule su complejidad
 - agregarAtras
 - obtener
 - eliminar
 - concatenar
- Escriba el invariante de representación para este módulo en castellano

- Dado el siguiente invrep, indique si es correcto. En caso de no serlo, corrijalo:

```

pred InvRep (l: NodoLista<T>) {
    accesible(l.primerO,l.ultimo)  $\wedge$  largoOK(l.primerO,l.longitud)
}
pred largoOK (n: NodoLista<T>, largo:  $\mathbb{Z}$ ) {
    (n = null  $\wedge$  largo = 0)  $\vee$  (largoOK(n.siguiente,largo - 1))
}
pred accesible (n0: NodoLista<T>, n1: NodoLista<T>) {
    n1 = n0  $\vee$  (n0.siguiente  $\neq$  null  $\wedge_L$  accesible(n0.siguiente,n1))
}

```

Solucion Ejercicio 1:

proc agregarAtras (inout l: ListaEnlazada<T>, in e: T)

```

1      nodo := new Nodo < T >
2      nodo.valor := e
3      if l.longitud == 0 then
4          l.primerO := nodo
5          l.ultimo := nodo
6      else
7          l.ultimo.siguiente := nodo
8          l.ultimo := nodo
9      endif
10     l.longitud ++

```

proc obtener (in l: ListaEnlazada<T>, in i \mathbb{Z}) : T

```

1      var actual := l.primerO
2      var j := 0
3      while j < i do
4          actual := actual.siguiente
5          j++
6      endwhile
7      return actual.valor

```

proc eliminar (inout l: ListaEnlazada<T>, in i \mathbb{Z})

```

1      var eliminar := l.primerO
2      var j := 0
3      while j < i do
4          eliminar := eliminar.siguiente
5          j ++
6      endwhile
7      if l.longitud == 1 then
8          l.primerO := null
9          l.ultimo = null
10     else if eliminar == l.primerO
11         l.primerO := eliminar.siguiente
12     else
13         anterior := l.primerO

```

```

14     var k := 0
15     while k < i - 1 do
16         anterior := anterior.siguiete
17         k ++
18     endwhile
19     if eliminar == l.ultimo then
20         l.ultimo := anterior
21         l.ultimo.siguiete := null
22     else
23         anterior.siguiete := eliminar.siguiete
24     endif
25     anterior := null
26     anterior.siguiete = null
27 endif
28 eliminar.siguiete := null
29 eliminar := null

```