

## 1. Módulo Lista Enlazada( $\alpha$ )

### Interfaz

**parámetros formales**

**géneros**  $\alpha$   
**función**  $\text{COPIAR}(\text{in } a : \alpha) \rightarrow \text{res} : \alpha$   
**Pre**  $\equiv \{\text{true}\}$   
**Post**  $\equiv \{\text{res} =_{\text{obs}} a\}$   
**Complejidad:**  $\Theta(\text{copy}(a))$   
**Descripción:** función de copia de  $\alpha$ 's

**se explica con:**  $\text{SECUENCIA}(\alpha)$ ,  $\text{ITERADOR BIDIRECCIONAL}(\alpha)$ .

**géneros:**  $\text{lista}(\alpha)$ ,  $\text{itLista}(\alpha)$ .

### Operaciones básicas de lista

$\text{VACÍA}() \rightarrow \text{res} : \text{lista}(\alpha)$   
**Pre**  $\equiv \{\text{true}\}$   
**Post**  $\equiv \{\text{res} =_{\text{obs}} <>\}$   
**Complejidad:**  $\Theta(1)$   
**Descripción:** genera una lista vacía.

$\text{AGREGARADELANTE}(\text{in/out } l : \text{lista}(\alpha), \text{in } a : \alpha) \rightarrow \text{res} : \text{itLista}(\alpha)$   
**Pre**  $\equiv \{l =_{\text{obs}} l_0\}$   
**Post**  $\equiv \{l =_{\text{obs}} a \bullet l_0 \wedge \text{res} = \text{CrearItBi}(<>, l) \wedge \text{alias}(\text{SecuSuby}(\text{res}) = l)\}$   
**Complejidad:**  $\Theta(\text{copy}(a))$   
**Descripción:** agrega el elemento  $a$  como primer elemento de la lista. Retorna un iterador a  $l$ , de forma tal que Siguiente devuelva  $a$ .  
**Aliasing:** el elemento  $a$  agrega por copia. El iterador se invalida si y sólo si se elimina el elemento siguiente del iterador sin utilizar la función  $\text{ELIMINARSIGUIENTE}$ .

### Operaciones del iterador

$\text{CREARIT}(\text{in } l : \text{lista}(\alpha)) \rightarrow \text{res} : \text{itLista}(\alpha)$   
**Pre**  $\equiv \{\text{true}\}$   
**Post**  $\equiv \{\text{res} =_{\text{obs}} \text{crearItBi}(<>, l) \wedge \text{alias}(\text{SecuSuby}(it) = l)\}$   
**Complejidad:**  $\Theta(1)$   
**Descripción:** crea un iterador bidireccional de la lista, de forma tal que al pedir SIGUIENTE se obtenga el primer elemento de  $l$ .  
**Aliasing:** el iterador se invalida si y sólo si se elimina el elemento siguiente del iterador sin utilizar la función  $\text{ELIMINARSIGUIENTE}$ .

$\text{CREARITULT}(\text{in } l : \text{lista}(\alpha)) \rightarrow \text{res} : \text{itLista}(\alpha)$   
**Pre**  $\equiv \{\text{true}\}$   
**Post**  $\equiv \{\text{res} =_{\text{obs}} \text{crearItBi}(l, <>) \wedge \text{alias}(\text{SecuSuby}(it) = l)\}$   
**Complejidad:**  $\Theta(1)$   
**Descripción:** crea un iterador bidireccional de la lista, de forma tal que al pedir ANTERIOR se obtenga el último elemento de  $l$ .  
**Aliasing:** el iterador se invalida si y sólo si se elimina el elemento siguiente del iterador sin utilizar la función  $\text{ELIMINARSIGUIENTE}$ .

## Representación

### Representación de la lista

$\text{lista}(\alpha)$  se representa con  $\text{lst}$   
donde  $\text{lst}$  es  $\text{tupla}(\text{primero} : \text{puntero}(\text{nodo}), \text{longitud} : \text{nat})$

donde **nodo** es  $\text{tupla}(\text{dato}: \alpha, \text{anterior}: \text{puntero}(\text{nodo}), \text{siguiente}: \text{puntero}(\text{nodo}))$

$\text{Rep} : \text{lst} \rightarrow \text{bool}$

$\text{Rep}(l) \equiv \text{true} \iff (l.\text{primero} = \text{NULL}) = (l.\text{longitud} = 0) \wedge_L (l.\text{longitud} \neq 0 \Rightarrow_L$   
 $\text{Nodo}(l, l.\text{longitud}) = l.\text{primero} \wedge$   
 $(\forall i: \text{nat})(\text{Nodo}(l, i) \rightarrow \text{siguiente} = \text{Nodo}(l, i + 1) \rightarrow \text{anterior}) \wedge$   
 $(\forall i: \text{nat})(1 \leq i < l.\text{longitud} \Rightarrow \text{Nodo}(l, i) \neq l.\text{primero})$

$\text{Nodo} : \text{lst } l \times \text{nat} \rightarrow \text{puntero}(\text{nodo})$

$\{l.\text{primero} \neq \text{NULL}\}$

$\text{Nodo}(l, i) \equiv \text{if } i = 0 \text{ then } l.\text{primero} \text{ else } \text{Nodo}(\text{FinLst}(l), i - 1) \text{ fi}$

$\text{FinLst} : \text{lst} \rightarrow \text{lst}$

$\text{FinLst}(l) \equiv \text{Lst}(l.\text{primero} \rightarrow \text{siguiente}, l.\text{longitud} - \min\{l.\text{longitud}, 1\})$

$\text{Lst} : \text{puntero}(\text{nodo}) \times \text{nat} \rightarrow \text{lst}$

$\text{Lst}(p, n) \equiv \langle p, n \rangle$

$\text{Abs} : \text{lst } l \rightarrow \text{secu}(\alpha)$

$\{\text{Rep}(l)\}$

$\text{Abs}(l) \equiv \text{if } l.\text{longitud} = 0 \text{ then } <> \text{ else } l.\text{primero} \rightarrow \text{dato} \bullet \text{Abs}(\text{FinLst}(l)) \text{ fi}$

## Representación del iterador

**itLista( $\alpha$ ) se representa con iter**

donde **iter** es  $\text{tupla}(\text{siguiente}: \text{puntero}(\text{nodo}), \text{lista}: \text{puntero}(\text{lst}))$

$\text{Rep} : \text{iter} \rightarrow \text{bool}$

$\text{Rep}(it) \equiv \text{true} \iff \text{Rep}(*(\text{it}.\text{lista})) \wedge_L (\text{it}.\text{siguiente} = \text{NULL} \vee_L (\exists i: \text{nat})(\text{Nodo}(*\text{it}.\text{lista}, i) = \text{it}.\text{siguiente}))$

$\text{Abs} : \text{iter } it \rightarrow \text{itBi}(\alpha)$

$\{\text{Rep}(it)\}$

$\text{Abs}(it) =_{\text{obs}} b: \text{itBi}(\alpha) \mid \text{Siguietes}(b) = \text{Abs}(\text{Sig}(\text{it}.\text{lista}, \text{it}.\text{siguiente})) \wedge$   
 $\text{Anteriores}(b) = \text{Abs}(\text{Ant}(\text{it}.\text{lista}, \text{it}.\text{siguiente}))$

$\text{Sig} : \text{puntero}(\text{lst}) \times \text{puntero}(\text{nodo}) \rightarrow \text{lst}$

$\{\text{Rep}(\langle l, p \rangle)\}$

$\text{Sig}(i, p) \equiv \text{Lst}(p, l \rightarrow \text{longitud} - \text{Pos}(*l, p))$

$\text{Ant} : \text{puntero}(\text{lst}) \times \text{puntero}(\text{nodo}) \rightarrow \text{lst}$

$\{\text{Rep}(\langle l, p \rangle)\}$

$\text{Ant}(i, p) \equiv \text{Lst}(\text{if } p = l \rightarrow \text{primero} \text{ then } \text{NULL} \text{ else } l \rightarrow \text{primero} \text{ fi}, \text{Pos}(*l, p))$

Nota: cuando  $p = \text{NULL}$ ,  $\text{Pos}$  devuelve la longitud de la lista, lo cual está bien, porque significa que el iterador no tiene siguiente.

$\text{Pos} : \text{lst } l \times \text{puntero}(\text{nodo}) \rightarrow \text{puntero}(\text{nodo})$

$\{\text{Rep}(\langle l, p \rangle)\}$

$\text{Pos}(l, p) \equiv \text{if } l.\text{primero} = p \vee l.\text{longitud} = 0 \text{ then } 0 \text{ else } 1 + \text{Pos}(\text{FinLst}(l), p) \text{ fi}$