



# Algoritmos Estructuras de Datos I

Facultad de Ciencias Exactas y Tecnología  
Universidad Nacional de Tucumán  
2023

# LISTA CIRCULAR - LISTA



# LISTA CIRCULAR

Una lista circular es una estructura de datos de tipo lineal donde el ultimo nodo esta ligado al primero y permite una operación de rotación entre sus elementos.

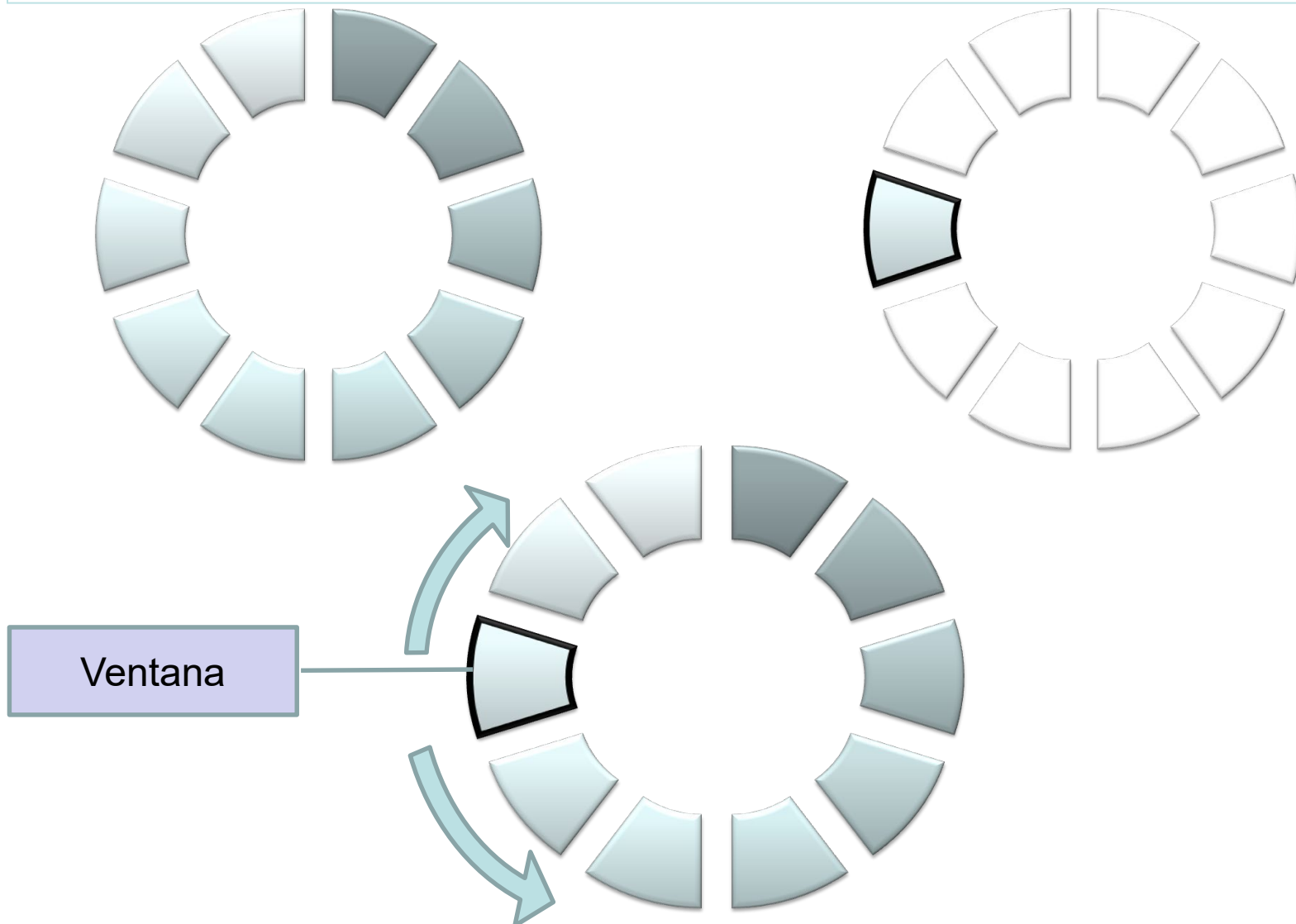
El acceso a sus componentes se hace con como en una LIFO, pero la operación de rotación permite pasar un elemento del tope de la pila al fondo de la pila.

## Ventaja:

Con la operación de rotación se puede acceder a todos sus elementos sin necesidad de borrarlos de la lista.



# Esquema LC



# LISTA CIRCULAR (ITEM)

## Especificación Algebraica

### OPERACIONES

#### A) Sintaxis:

LCVACIA :  $\rightarrow$  LISTACIRCULAR

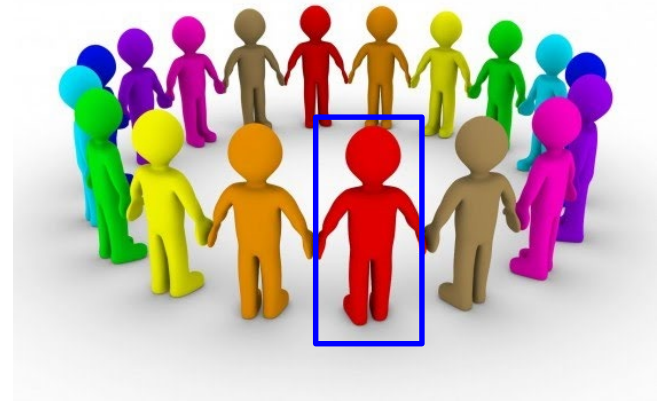
LCINSERTAR : LISTACIRCULAR  $\times$  ITEM  $\rightarrow$  LISTACIRCULAR

LCBORRAR : LISTACIRCULAR  $\rightarrow$  LISTACIRCULAR

LCVALOR : LISTACIRCULAR  $\rightarrow$  ITEM  $\cup$  {indefinido}

ESLCVACIA : LISTACIRCULAR  $\rightarrow$  BOOLEAN

LCROTAR : LISTACIRCULAR  $\rightarrow$  LISTACIRCULAR



# LISTA CIRCULAR (ITEM)

## Especificación Algebraica

**B) Semántica:** Para todo  $L \in \text{LISTACIRCULAR}$ ,  $\forall i, j \in \text{ITEM}$

$\text{ESLCVACIA}(\text{LCVACIA}) \equiv \text{TRUE}$

$\text{ESLCVACIA}(\text{LCINSERTAR}(L, i)) \equiv \text{FALSE}$

$\text{LCBORRAR}(\text{LCVACIA}) \equiv \text{LCVACIA}$

$\text{LCBORRAR}(\text{LCINSERTAR}(L, i)) \equiv L$

$\text{LCVALOR}(\text{LCVACIA}) \equiv \text{indefinido}$

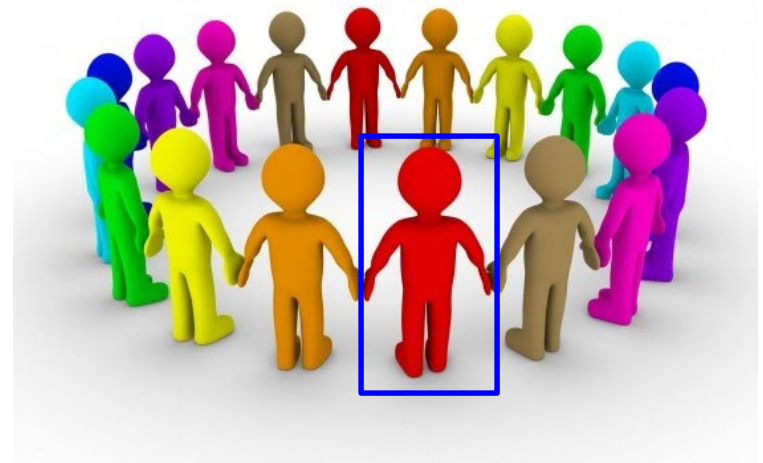
$\text{LCVALOR}(\text{LCINSERTAR}(L, i)) \equiv i$

$\text{LCROTAR}(\text{LCVACIA}) \equiv \text{LCVACIA}$

$\text{LCROTAR}(\text{LCINSERTAR}(\text{LCVACIA}, i)) \equiv \text{LCINSERTAR}(\text{LCVACIA}, i)$

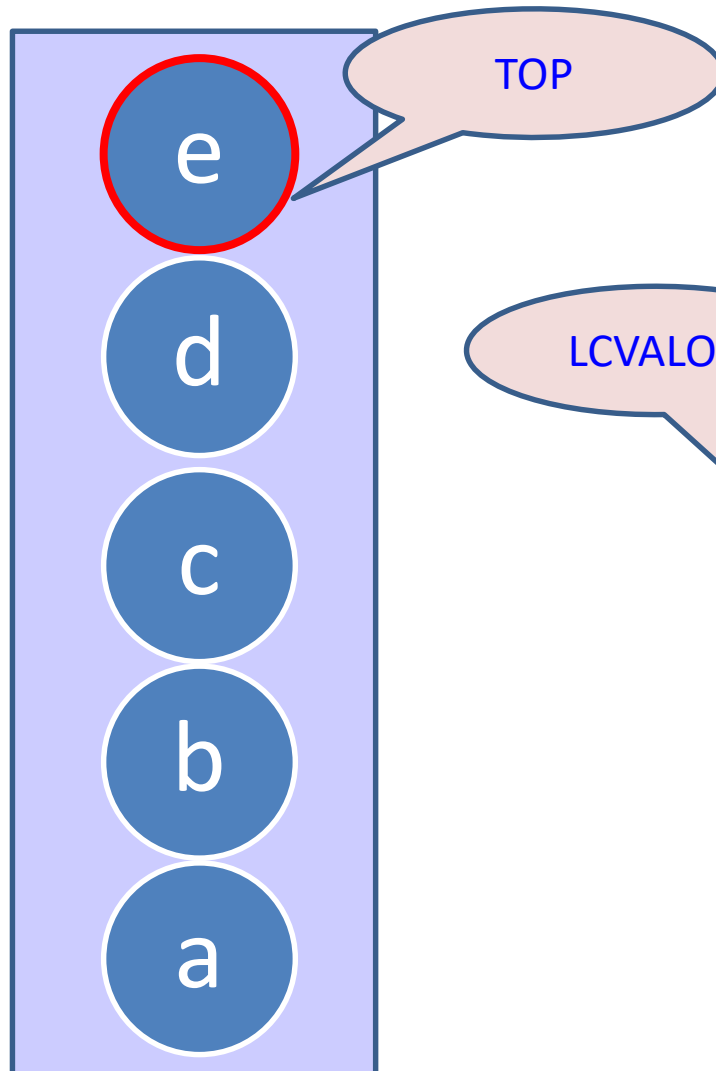
$\text{LCROTAR}(\text{LCINSERTAR}(\text{LCINSERTAR}(L, i), j)) \equiv$

$\text{LCINSERTAR}(\text{LCROTAR}(\text{LCINSERTAR}(L, j)), i)$

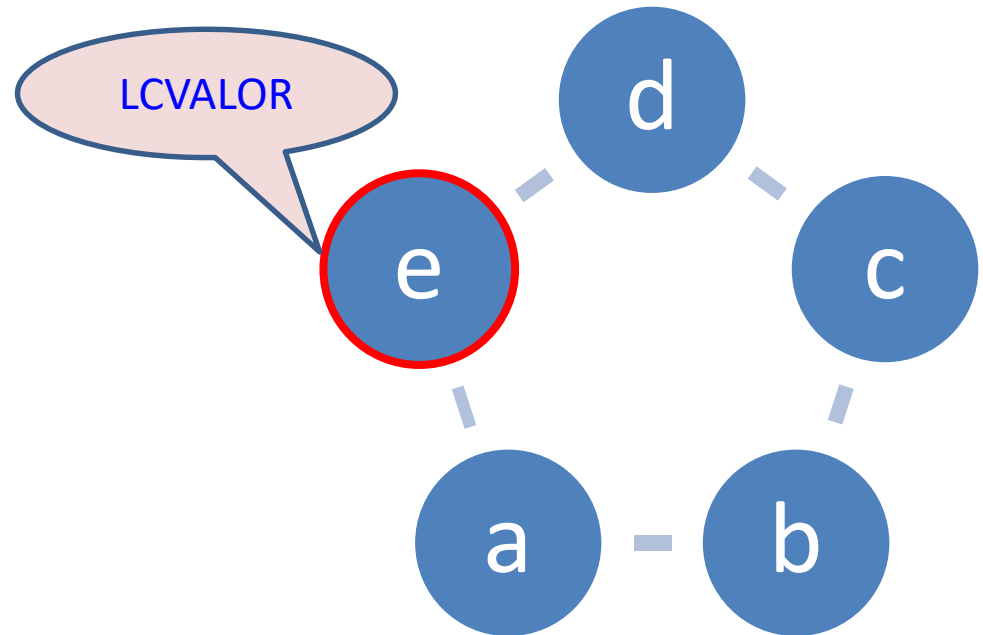


# Comparación PILA con LISTACIRCULAR

PILA:



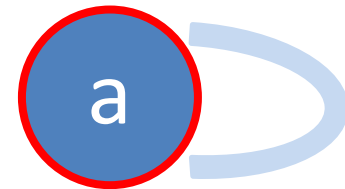
LISTACIRCULAR:



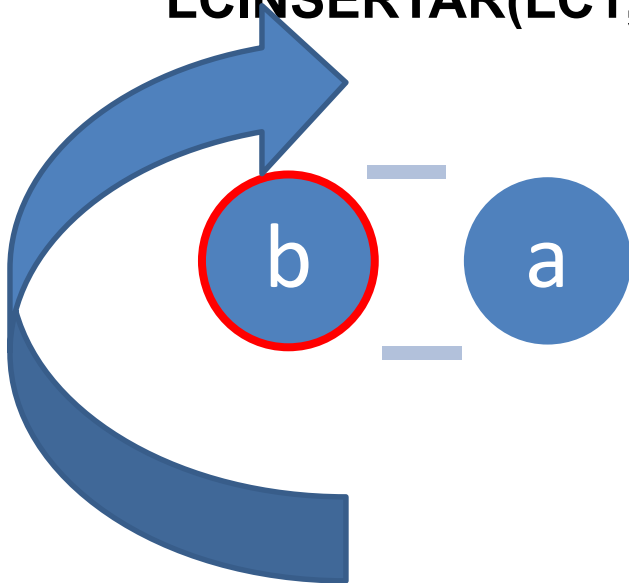
# Esquema LCINSERTAR

LCVACIA(LC1):

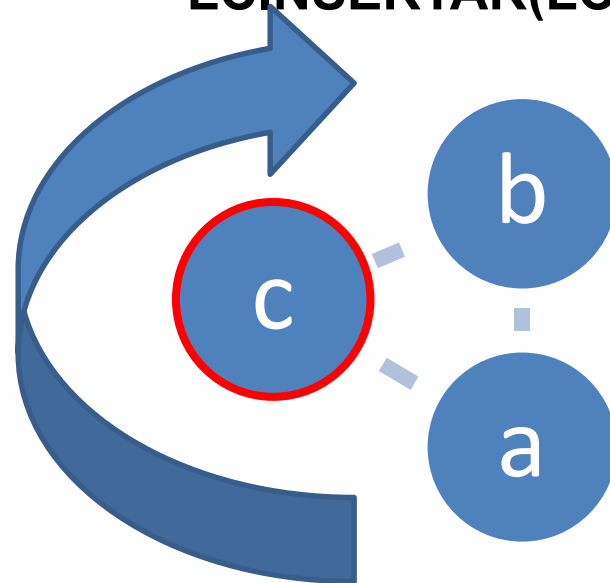
LCINSERTAR(LC1,a):



LCINSERTAR(LC1,b):



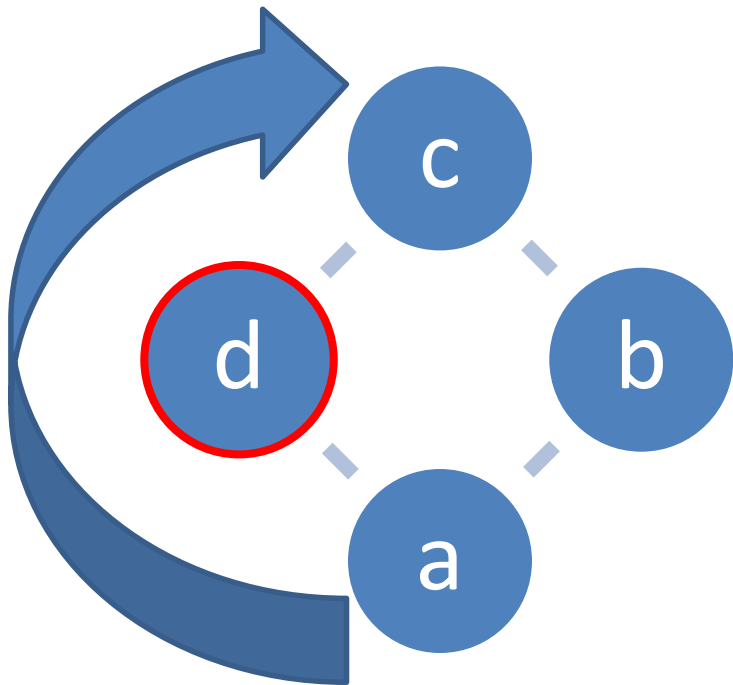
LCINSERTAR(LC1,c):



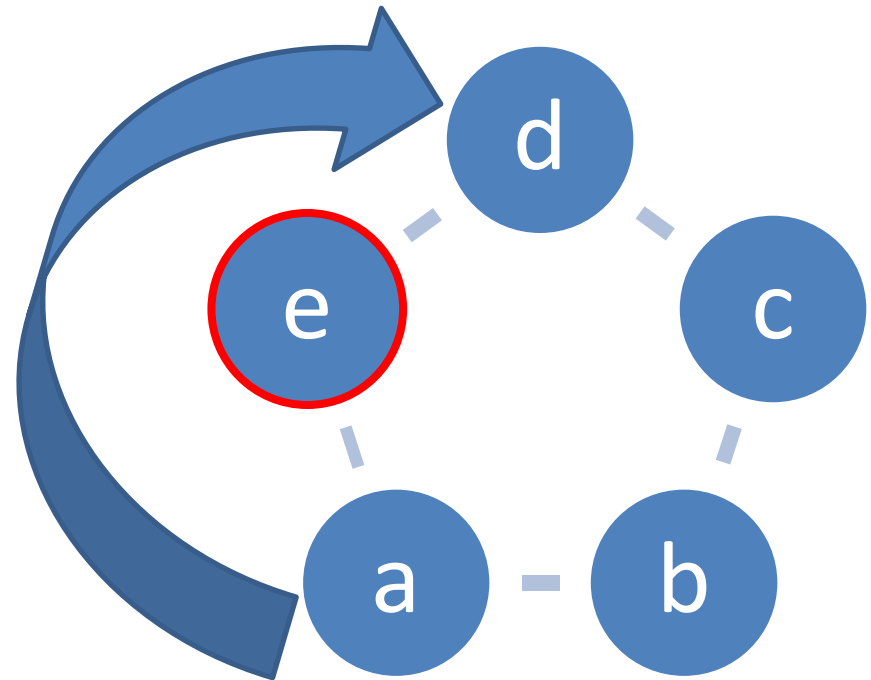


# Esquema LCINSERTAR

LCINSERTAR(LC1,d):

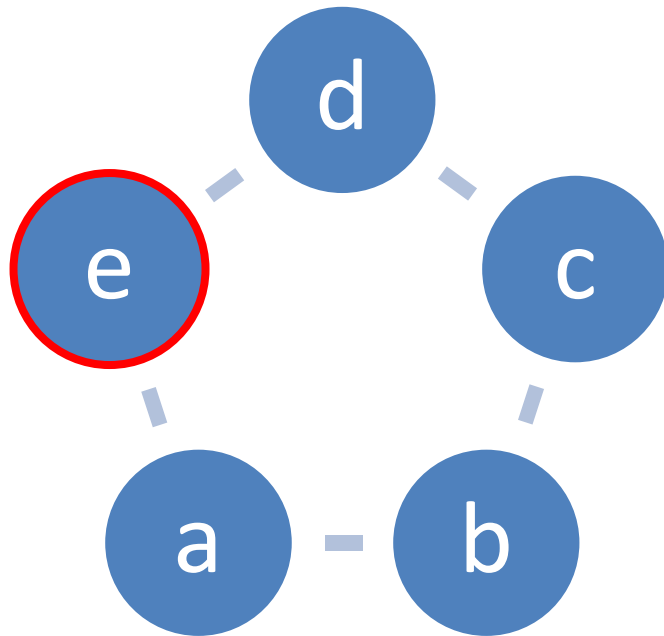


LCINSERTAR(LC1,e):

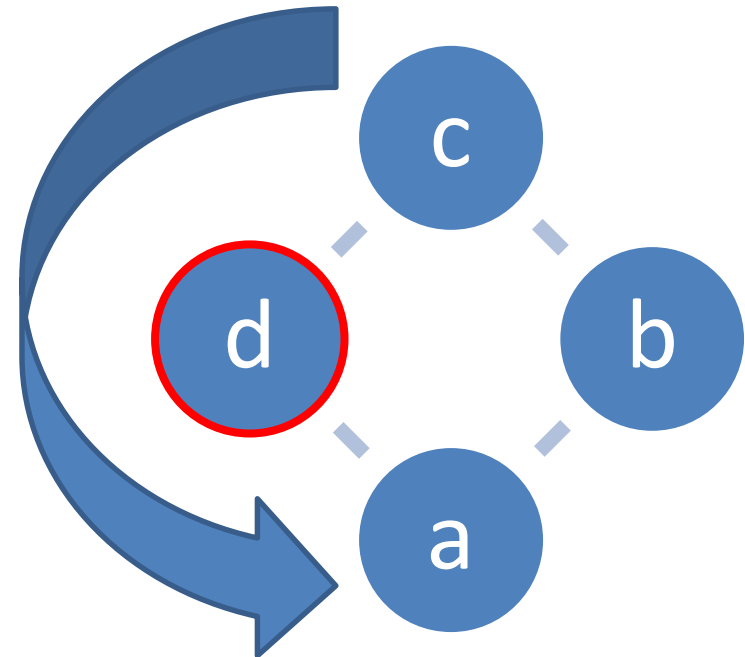


# Esquema LCBORRAR

LC1:



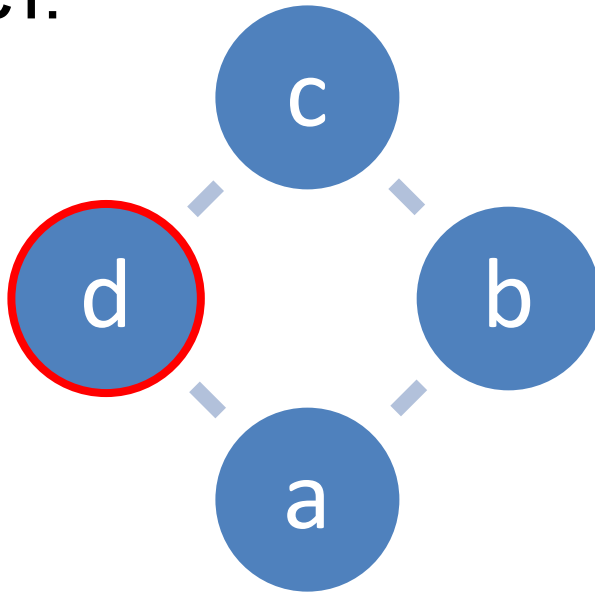
LCBORRAR(LC1):



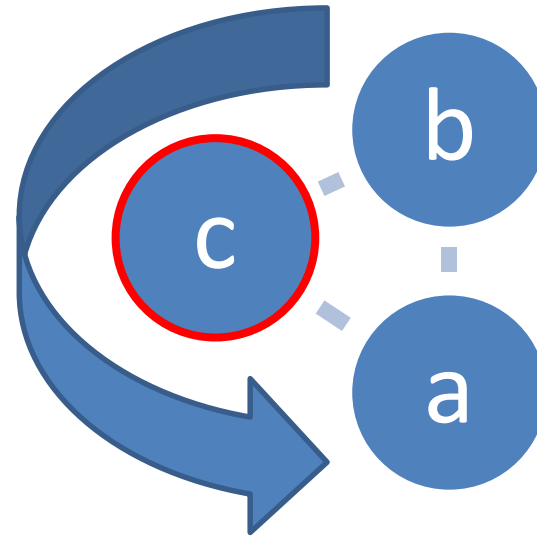
$$\text{LCBORRAR}(\text{LCINSERTAR}(L,i)) \equiv L$$

# Esquema LCBORRAR

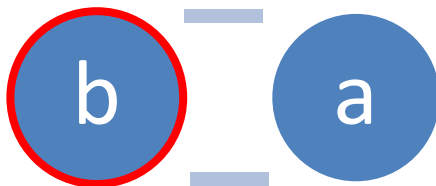
LC1:



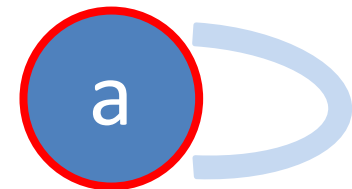
LCBORRAR(LC1):



LCBORRAR(LC1):

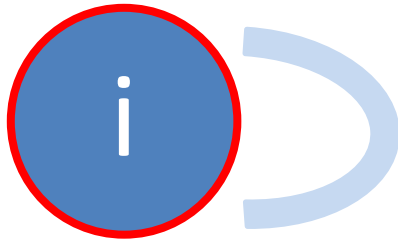


LCBORRAR(LC1):

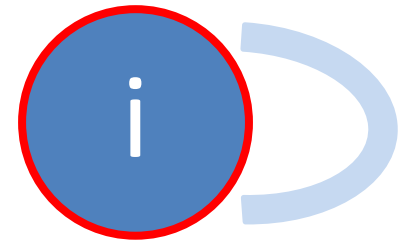


# Esquema LCROTAR

**LC2:**

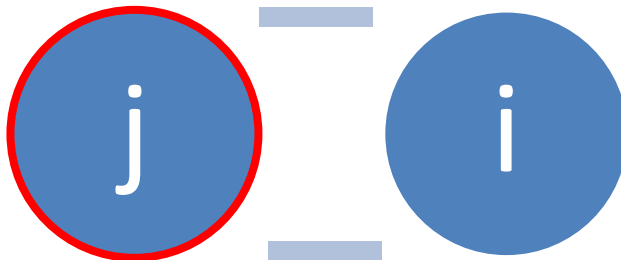


**LCROTAR(LC2):**

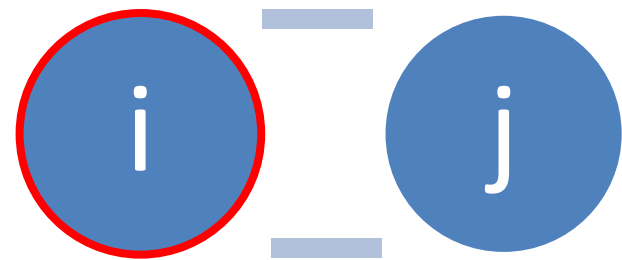


$$\text{LCROTAR}(\text{LCINSERTAR}(\text{LCVACIA}, i)) \equiv \text{LCINSERTAR}(\text{LCVACIA}, i)$$

**LC3:**



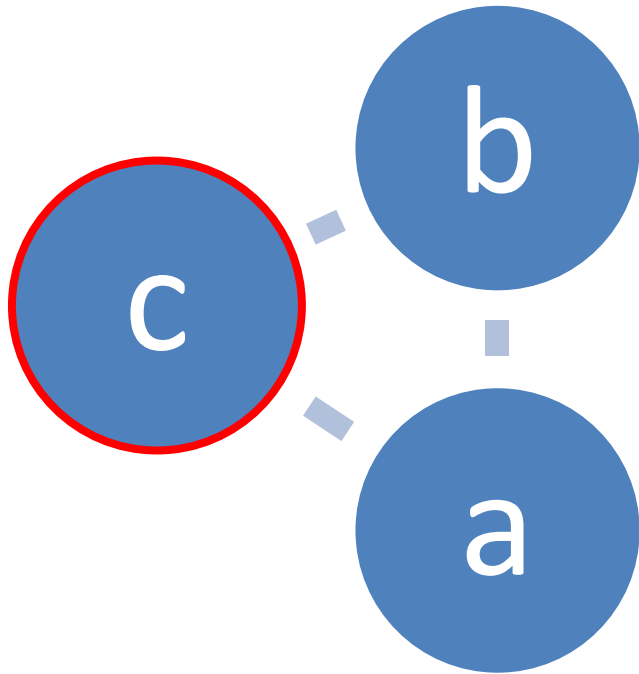
**LCROTAR(LC3):**



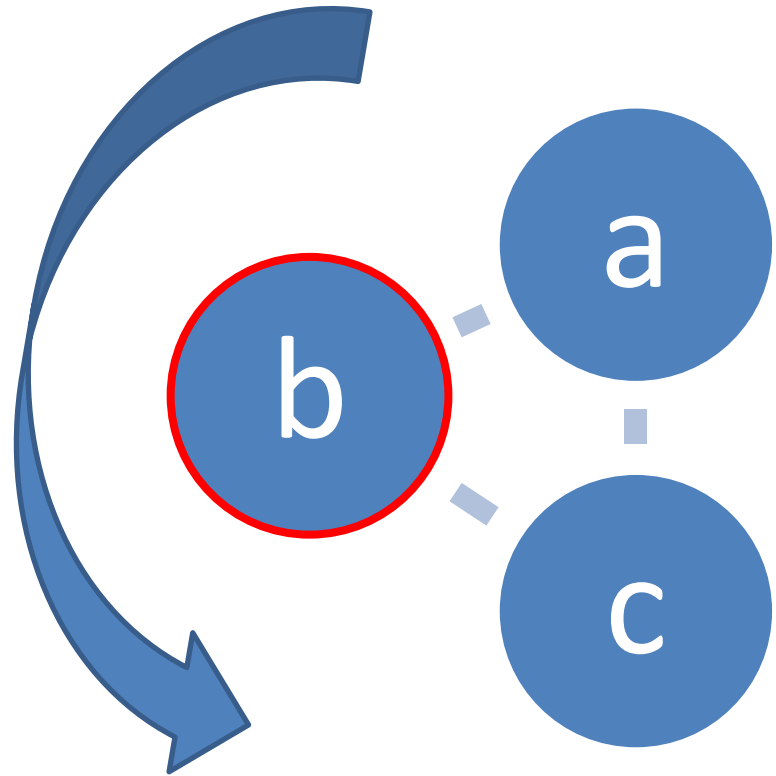
$$\text{LCROTAR}(\text{LCINSERTAR}(\text{LCINSERTAR}(L, i), j)) \equiv \text{LCINSERTAR}(\text{LCROTAR}(\text{LCINSERTAR}(L, j)), i)$$

# Esquema LCROTAR

LC4:



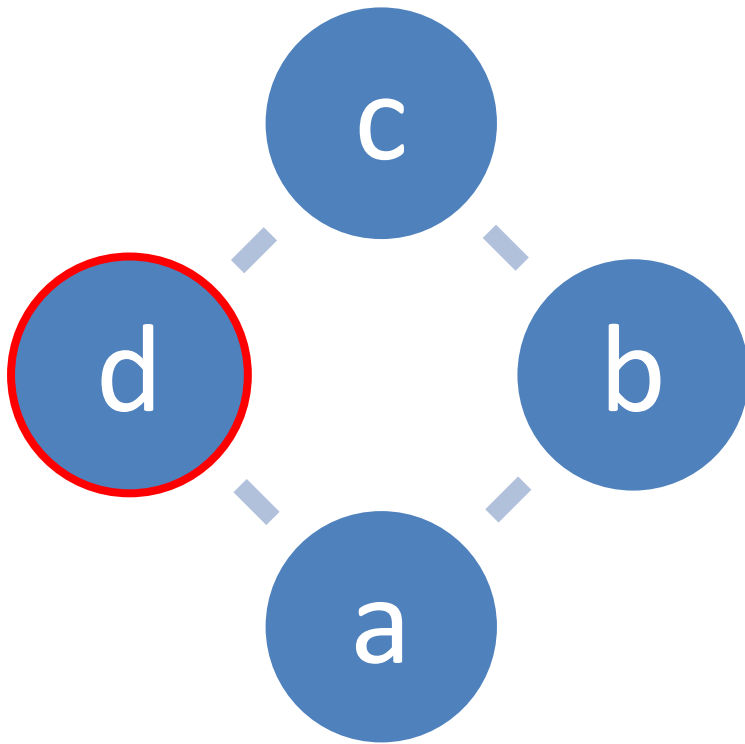
LCROTAR(LC4):



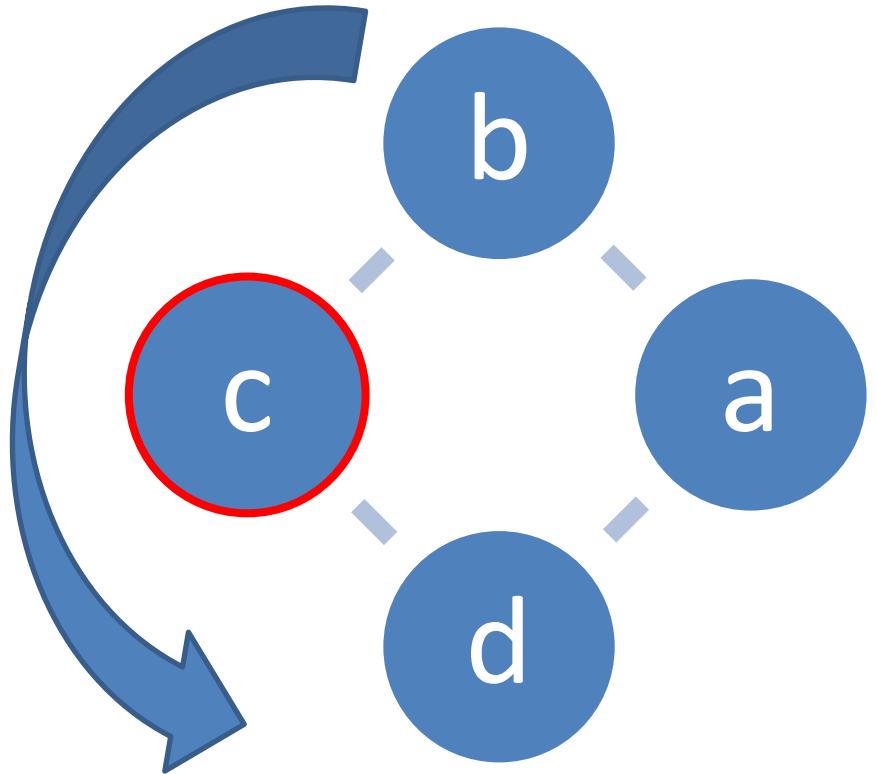
$$\text{LCROTAR}(\text{LCINSERTAR}(\text{LCINSERTAR}(L,i),j)) \equiv \text{LCINSERTAR}(\text{LCROTAR}(\text{LCINSERTAR}(L,j)),i)$$

# Esquema LCROTAR

LC5:



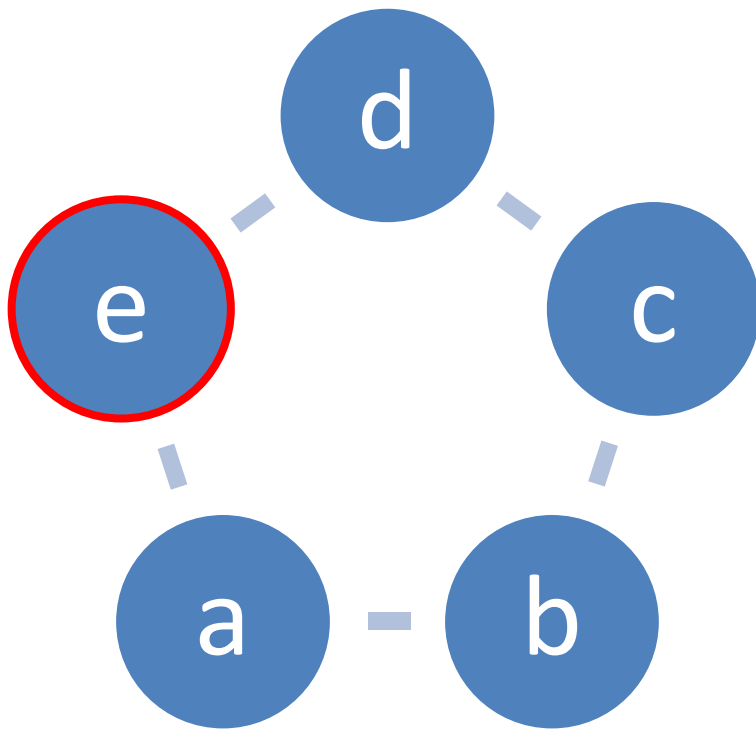
LCROTAR(LC5):



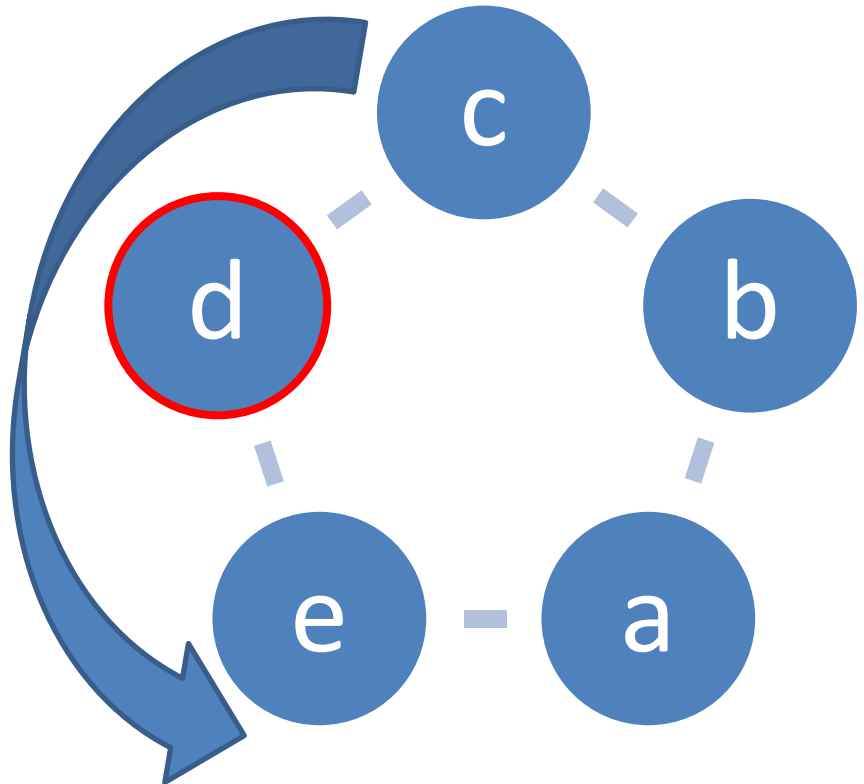
$$\text{LCROTAR}(\text{LCINSERTAR}(\text{LCINSERTAR}(L,i),j)) \equiv \text{LCINSERTAR}(\text{LCROTAR}(\text{LCINSERTAR}(L,j)),i)$$

# Esquema LCROTAR

LC6:



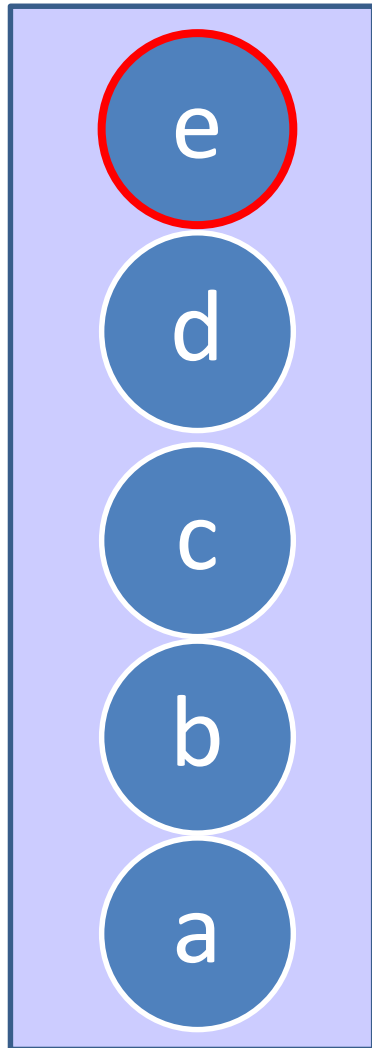
LCROTAR(LC6):



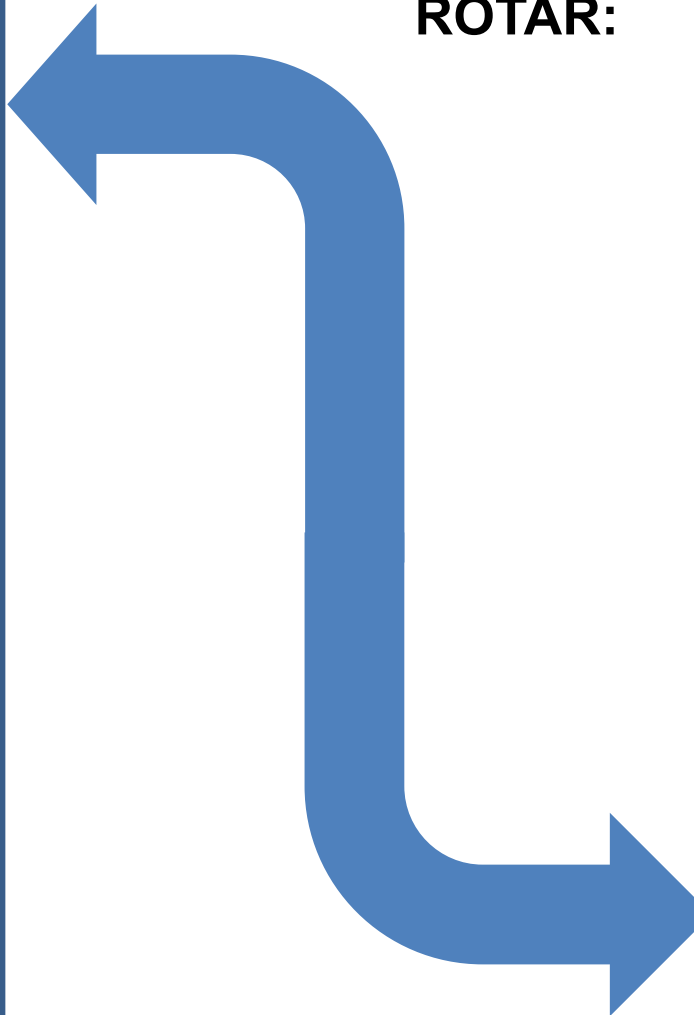
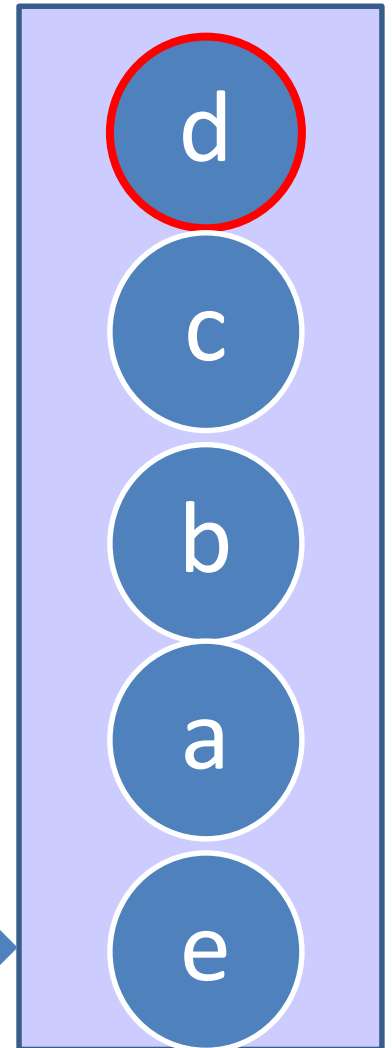
$$\text{LCROTAR}(\text{LCINSERTAR}(\text{LCINSERTAR}(\text{L}, i), j)) \equiv \text{LCINSERTAR}(\text{LCROTAR}(\text{LCINSERTAR}(\text{L}, j)), i)$$

# Esquema LCROTAR sobre una PILA

PILA:



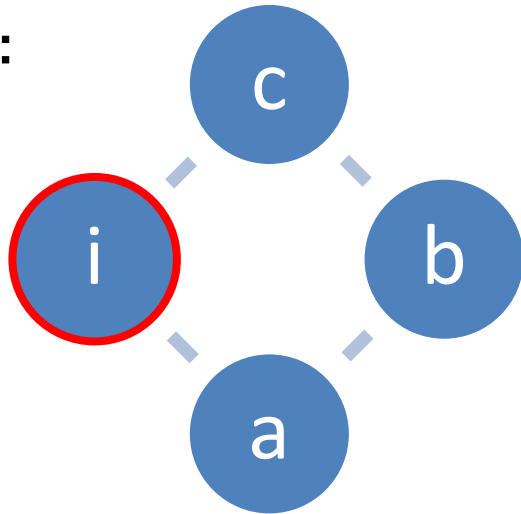
ROTAR:



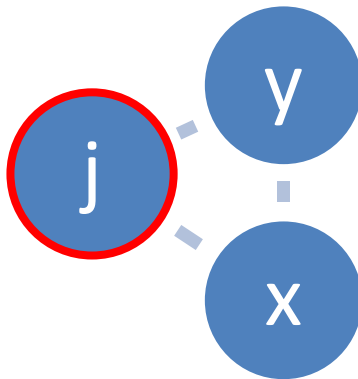


# Esquema operación LCUNIR

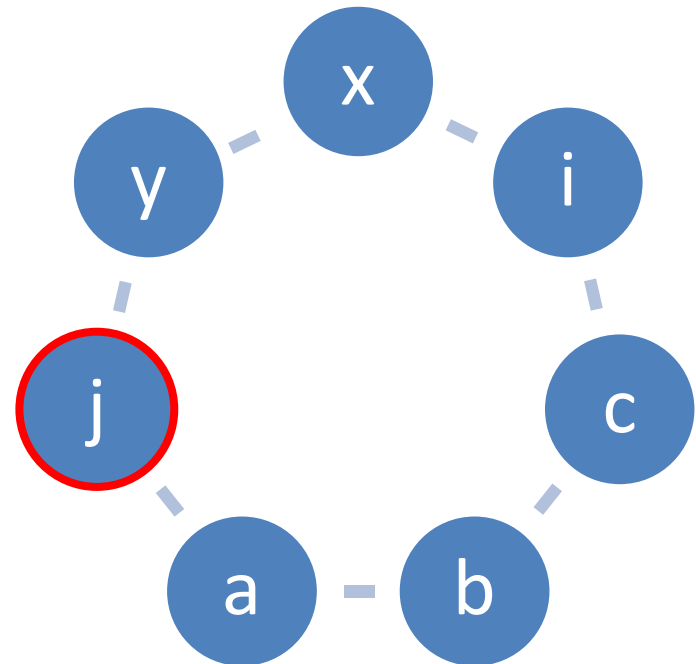
LC1:



LC2:



LCUNIR(LC1,LC2):



# LISTA CIRCULAR

## Aplicación: unir 2 listas circulares

### 1) Dentro de la Especificación Algebraica, 4 axiomas

Sintaxis:

$LCUNIR : LISTACIRCULAR \times LISTACIRCULAR \rightarrow LISTACIRCULAR$

Semántica:  $\forall L1, L2 \in LISTACIRCULAR, \forall i, j \in ITEM$

$LCUNIR (LCVACIA, LCVACIA) \equiv LCVACIA$

$LCUNIR (LCINSERTAR(L1, i), LCVACIA) \equiv LCINSERTAR(L1, i)$

$LCUNIR (LCVACIA, LCINSERTAR(L2, j)) \equiv LCINSERTAR(L2, j)$

$LCUNIR (LCINSERTAR(L1, i), LCINSERTAR(L2, j)) \equiv$

$LCINSERTAR (LCUNIR (LCINSERTAR(L1, i), L2), j)$

# LISTA CIRCULAR

Aplicación: unir 2 listas circulares

## 2) Dentro de la Especificación Algebraica, 2 axiomas

Sintaxis:

$LCUNIR : LISTACIRCULAR \times LISTACIRCULAR \rightarrow LISTACIRCULAR$

Semántica:  $\forall L1, L2 \in LISTACIRCULAR, \forall j \in ITEM$

$LCUNIR (L1, LCVACIA) \equiv L1$

$LCUNIR (L1, LCINSERTAR(L2, j)) \equiv$

$LCINSERTAR (LCUNIR (L1, L2) , j)$

# **LISTA CIRCULAR**

Aplicación: unir 2 listas circulares

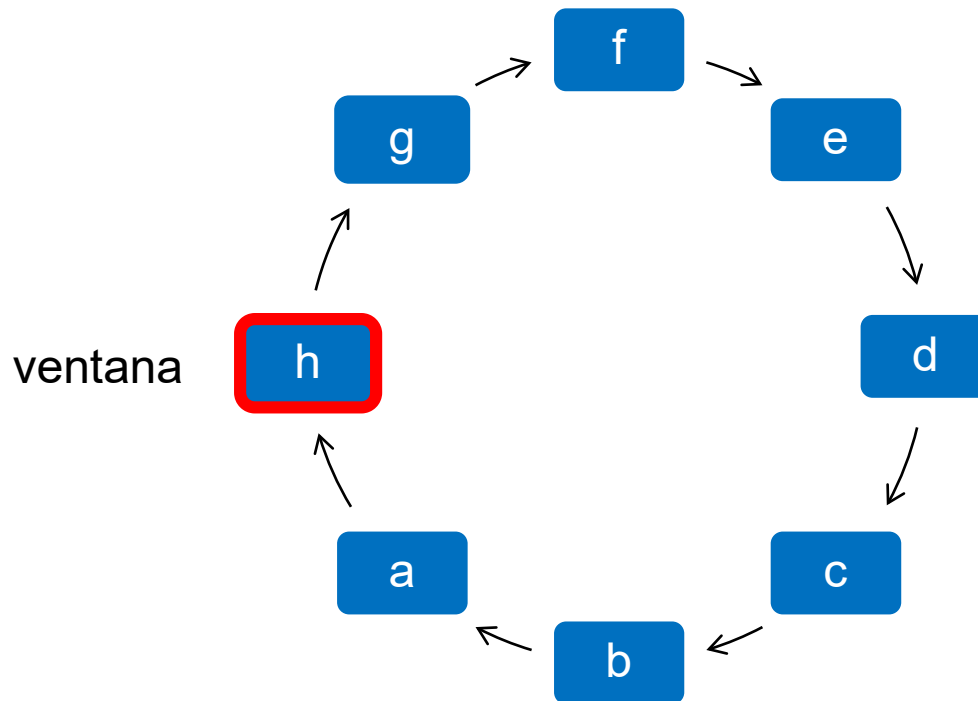
**Ejercitacion:**

**3) Como usuario del ADT LC, función recursiva**

**4) Como usuario del ADT LC, función iterativa**

# Implementación LC

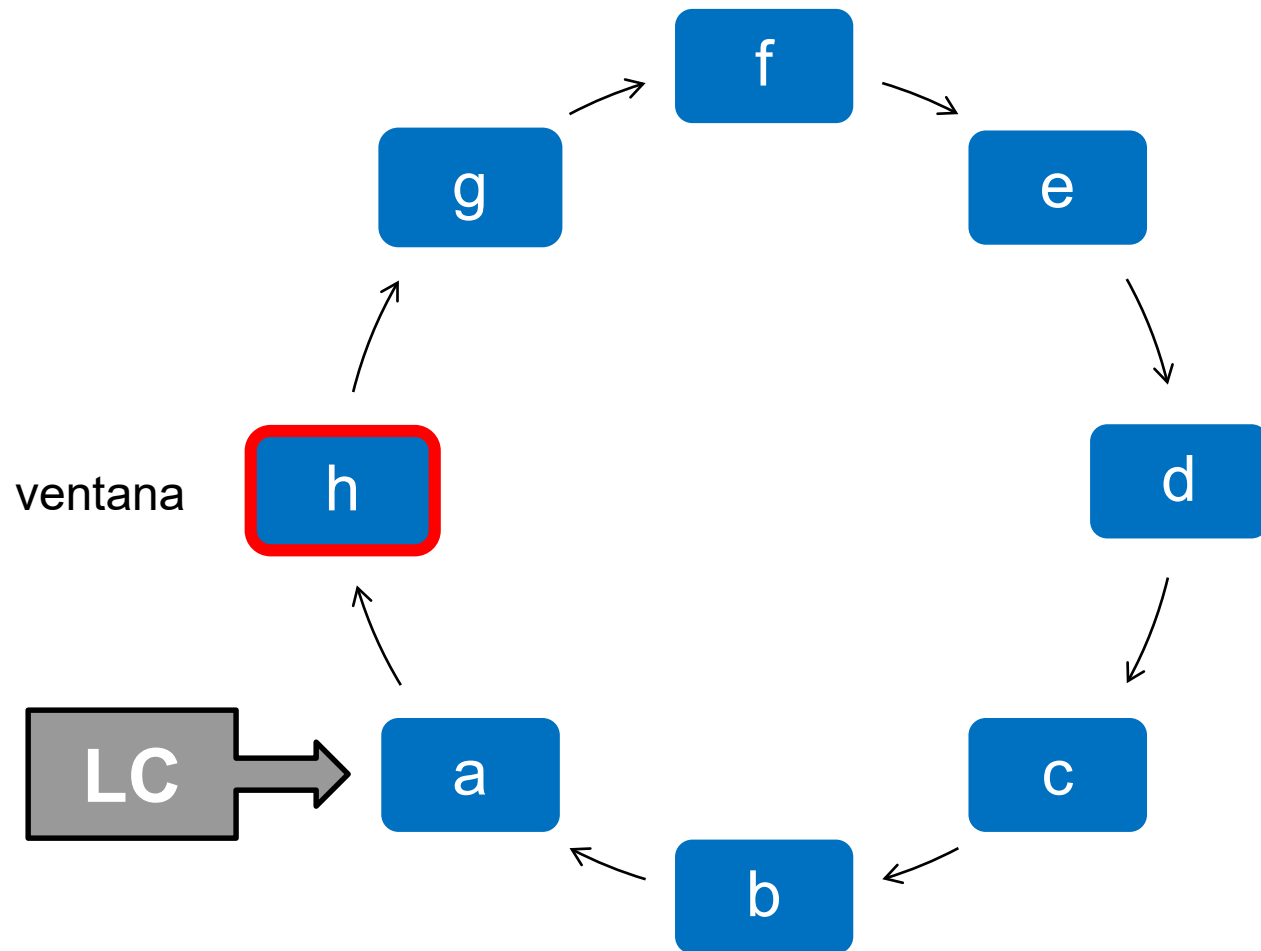
Con lista enlazada circular:



Donde se pone el puntero a la lista enlazada circular?:

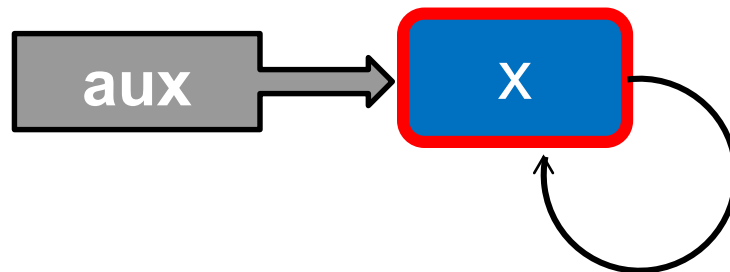
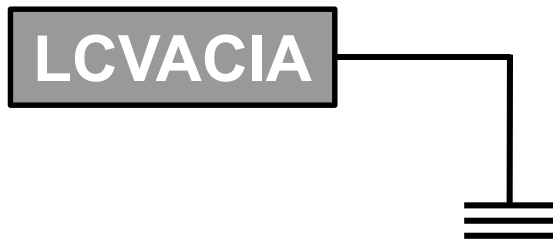


# Implementación LC

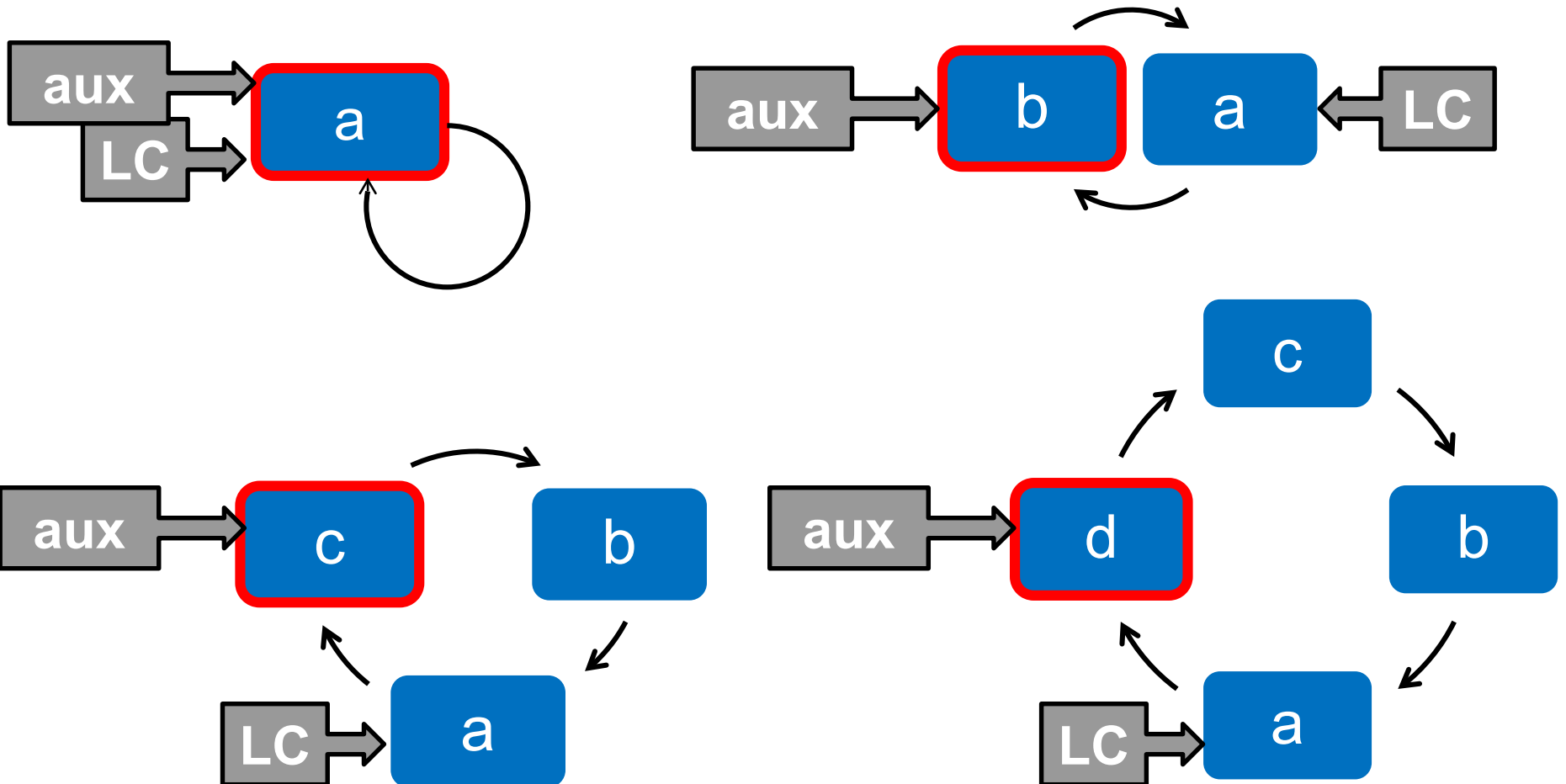


***// NOTA: todas las operaciones son  $O(1)$  en esta implementación*** <sup>22</sup>

# Implementación LCINSERTAR

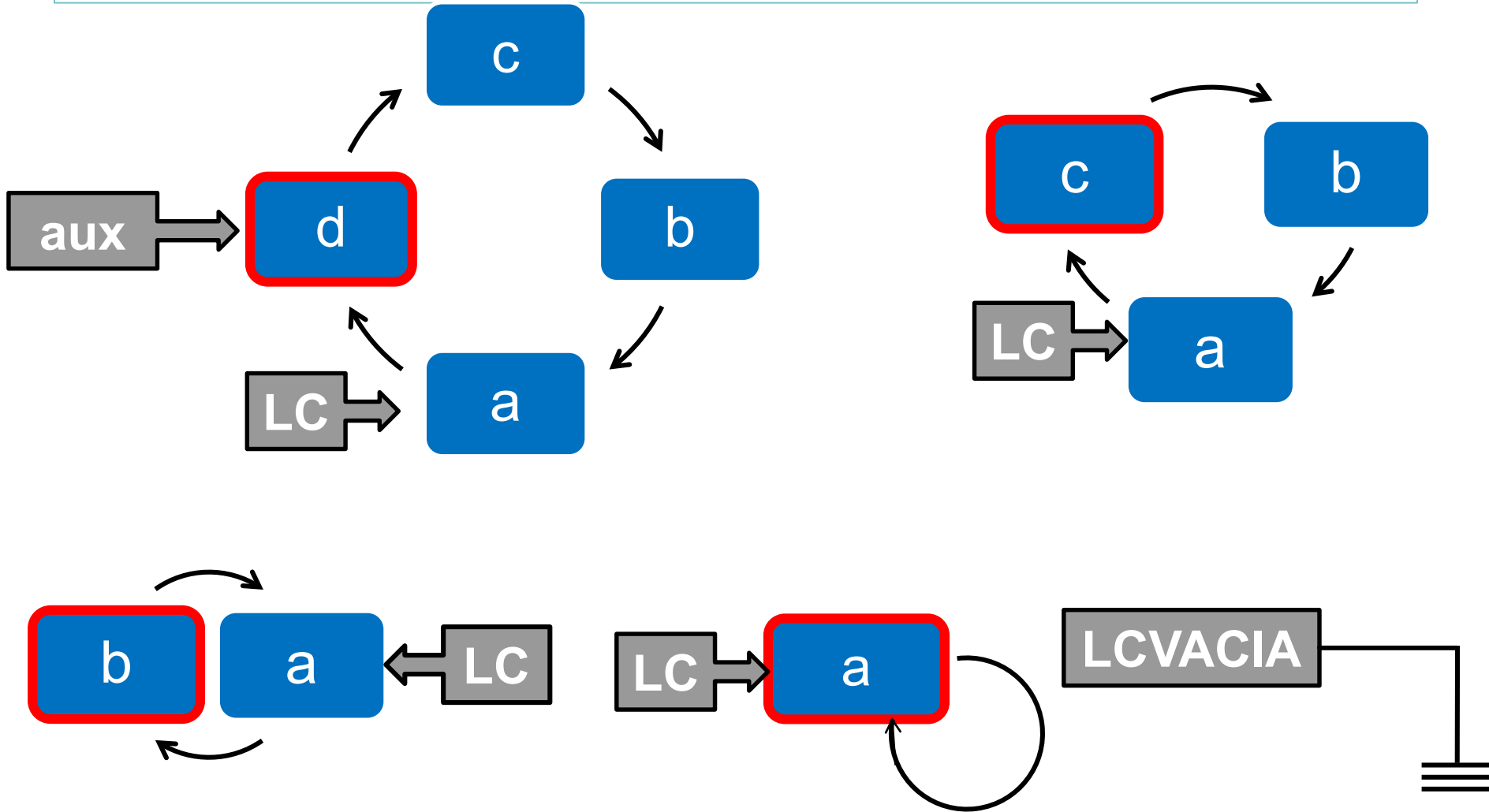


# Implementación LCINSERTAR

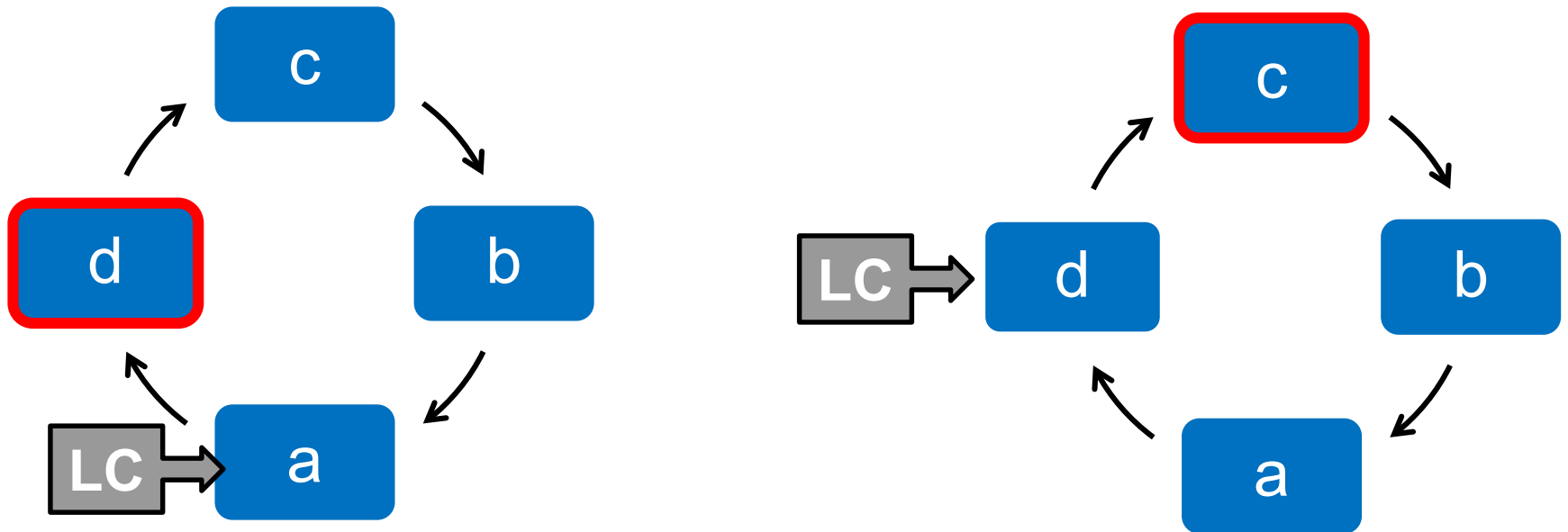




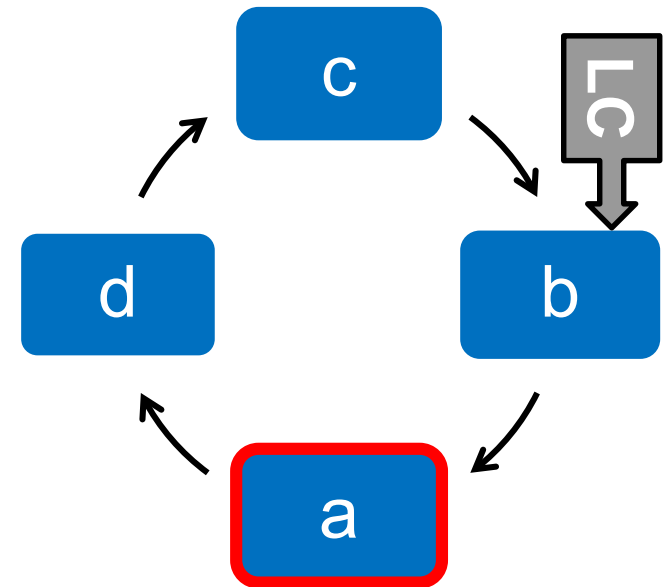
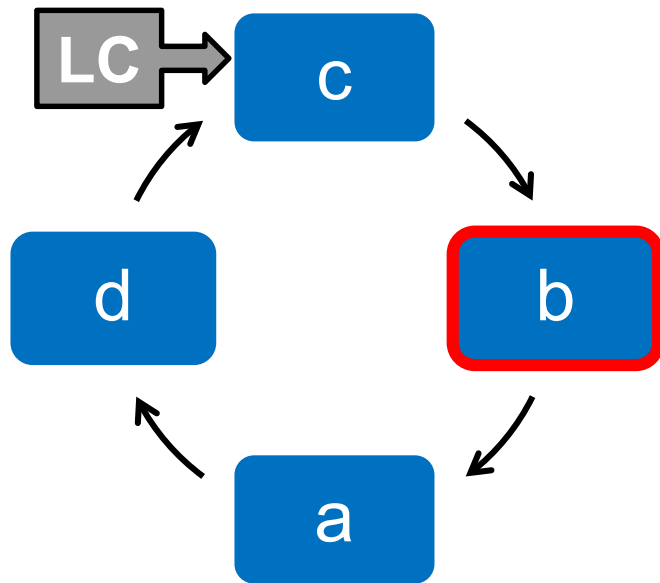
# Implementación LCBORRAR



# Implementación LCROTAR



# Implementación LCROTAR



# ADT LISTA



La lista es una secuencia de cero o mas elementos de un tipo dado. Una lista es una estructura para almacenar información con la característica de que puede contener cualquier número de elementos, y que estos están representados en un cierto orden.

Una lista es una estructura particularmente flexible que es de mucha utilidad en la vida diaria como por ejemplo: lista de precios, lista de estudiantes, lista de candidatos, etc.

# ADT LISTA



Los elementos de una lista se distinguen por su posición en la misma, se habla del primer elemento, del segundo, del k-ésimo, del ultimo.

El ADT LISTA esta basado en el concepto de *posición* que es un valor entero. Se define en función de un tipo base *item* que representa el tipo de elementos que contendrá la lista. De esta manera se define el ADT independiente del tipo de elementos que contenga.

# Ejemplo: ADT LISTA

Posicion	Contenido
1	UNIDAD I) Algoritmos.
2	UNIDAD II) Tipos abstractos de datos básicos.
3	UNIDAD III) Tipos de datos no lineales.
4	UNIDAD IV) Ordenamiento.
5	UNIDAD V) Búsqueda.

# LISTA (ITEM)

## Especificación Algebraica

### OPERACIONES      A) Sintaxis:

LISTAVACIA :  $\rightarrow$  LISTA

\* GUARDAR : LISTA x ENTERO x ITEM  $\rightarrow$  LISTA

INSERTAR : LISTA x ENTERO x ITEM  $\rightarrow$  LISTA

OBTENER : LISTA x ENTERO  $\rightarrow$  ITEM U {indefinido}

BORRAR : LISTA x ENTERO  $\rightarrow$  LISTA

LONG : LISTA  $\rightarrow$  ENTERO  $\geq 0$

OBSERVACION: la constructora **GUARDAR** es una **constructora escondida** del tipo, se usa solo para la especificación algebraica. El usuario tiene disponible solamente las constructoras LISTAVACIA e INSERTAR.

# LISTA (ITEM)

## Especificación Algebraica

**B) Semántica:** Para todo  $L \in \text{LISTA}$ ,  $\forall i, j \in \text{ENTERO}$ ,  $\forall a, b \in \text{ITEM}$

$\text{LONG}(\text{LISTAVACIA}) \equiv 0$

$\text{LONG}(\text{GUARDAR}(L, i, a)) \equiv 1 + \text{LONG}(L)$

$\text{OBTENER}(\text{LISTAVACIA}, j) \equiv \text{indefinido}$

$\text{OBTENER}(\text{GUARDAR}(L, i, a), j) \equiv \text{si } i=j \text{ entonces } a$

$\text{sino } \text{OBTENER}(L, j)$

$\text{INSERTAR}(\text{LISTAVACIA}, j, b) \equiv \text{GUARDAR}(\text{LISTAVACIA}, j, b)$

$\text{INSERTAR}(\text{GUARDAR}(L, i, a), j, b) \equiv \text{si } i \geq j \text{ entonces}$

$\text{GUARDAR}(\text{INSERTAR}(L, j, b), i+1, a)$

$\text{sino } \text{GUARDAR}(\text{INSERTAR}(L, j, b), i, a)$

$\text{BORRAR}(\text{LISTAVACIA}, j) \equiv \text{LISTAVACIA}$

$\text{BORRAR}(\text{GUARDAR}(L, i, a), j) \equiv \text{si } i=j \text{ entonces } \text{BORRAR}(L, j)$

$\text{sino si } i > j \text{ entonces } \text{GUARDAR}(\text{BORRAR}(L, j), i-1, a)$

$\text{sino } \text{GUARDAR}(\text{BORRAR}(L, j), i, a)$



# LISTA

## Aplicación: igualdad de listas

### 1) Dentro de la Especificación Algebraica

Sintaxis:

IGUALI : LISTA x LISTA  $\rightarrow$  BOOL

Semántica:  $\forall L1, L2 \in \text{LISTA}, \forall i, j \in \text{ENTERO}, \forall a, b \in \text{ITEM}$

IGUALI(LISTAVACIA, LISTAVACIA)  $\equiv$  TRUE

IGUALI(LISTAVACIA, GUARDAR(L2, i, a))  $\equiv$  FALSE

IGUALI(GUARDAR(L1, i, a), LISTAVACIA)  $\equiv$  FALSE

IGUALI(GUARDAR(L1, i, a), GUARDAR(L2, j, b))  $\equiv$   
 $i=j \text{ AND } a=b \text{ AND IGUALI}(L1, L2)$

*donde = es la operación que permite comparar 2 objetos del tipo item*

# LISTA

## Aplicación: igualdad de listas

### 2) Como usuario del ADT LISTA, función recursiva

Funcion IGUALISTA (L1,L2) : LISTA x LISTA  $\rightarrow$  BOOL

Si LONG(L1)  $\neq$  LONG(L2) entonces

Retorna FALSE

Sino

Retorna IGUAL (L1,L2,1)

Fin

# LISTA

## Aplicación: igualdad de listas

### 2) Como usuario del ADT LISTA, función recursiva

Funcion IGUAL(L1,L2,posicion): LISTAxLISTAxENTERO → BOOL

SI posicion > LONG(L1) ENTONCES

RETORNA TRUE

SINO

RETORNA

OBTENER(L1, posicion) = OBTENER(L2, posicion)

AND IGUAL (L1,L2, posicion + 1)

Fin

*donde = es la operación que permite comparar 2 objetos del tipo item*

# LISTA

## Aplicación: igualdad de listas

**3) Como usuario del ADT LISTA**, función iterativa

Funcion IGUALISTA (L1,L2) : LISTA x LISTA  $\rightarrow$  BOOL

Si LONG(L1)  $\neq$  LONG(L2) entonces

Retorna FALSE

Sino

$L \leftarrow \text{LONG}(L1)$

$i \leftarrow 1$

Mientras  $i \leq L$  AND  $\text{OBTENER}(L1,i) = \text{OBTENER}(L2,i)$

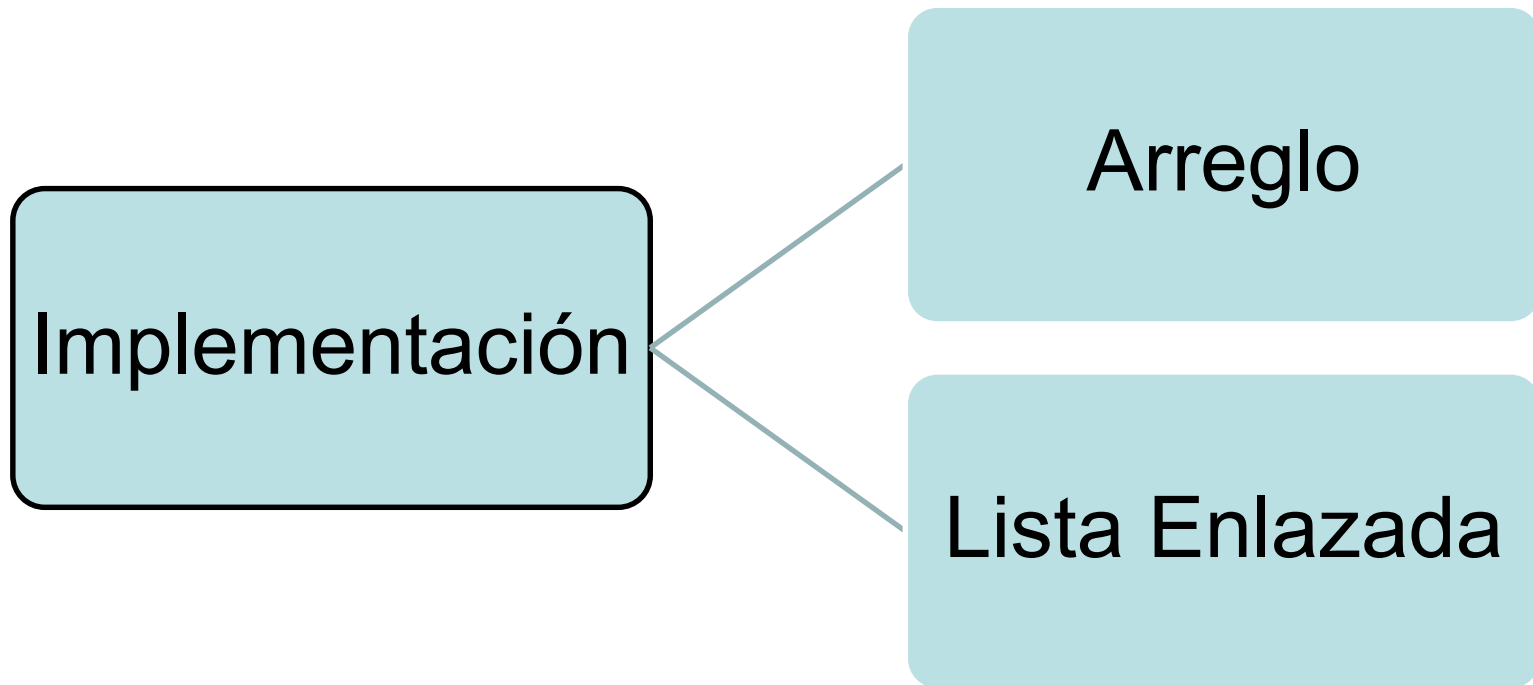
Hacer

$i \leftarrow i+1$

Retorna  $i > L$

Fin

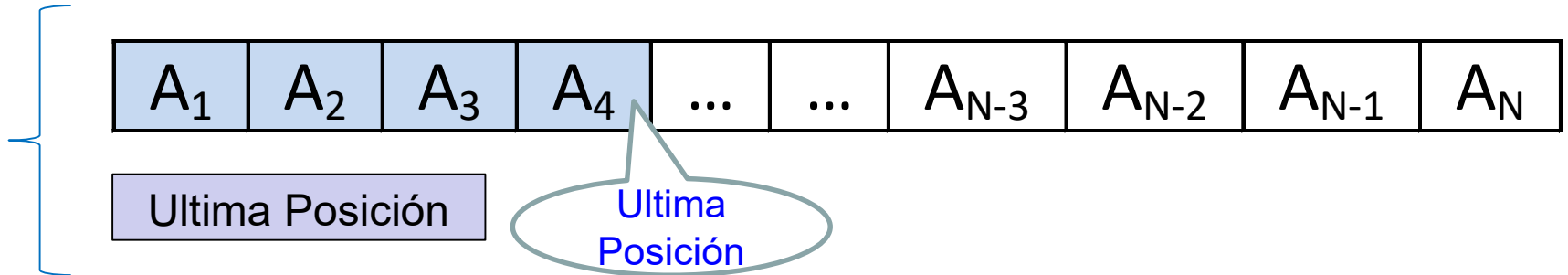
# LISTA



Ventajas y Desventajas

# Implementación Lista con arreglo

## LISTA:

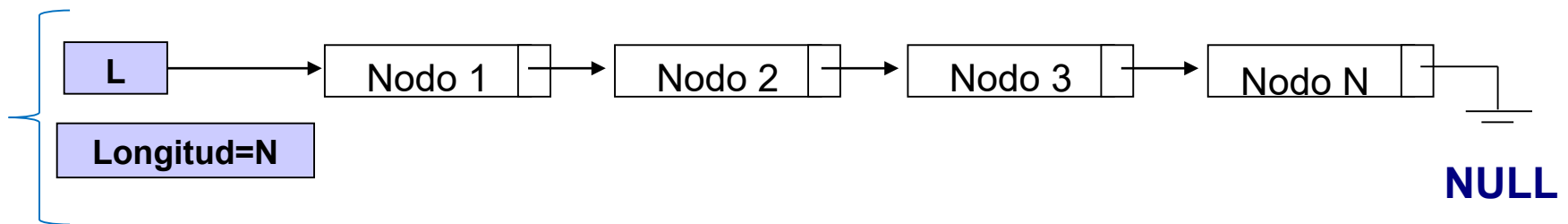


## Costo de las Operaciones:

LISTAVACIA	$\in O(1)$
GUARDAR	$\in O(1)$
INSERTAR	$\in O(N)$
OBTENER	$\in O(1)$
BORRAR	$\in O(N)$
LONG	$\in O(1)$

# Implementación Lista con lista enlazada

## LISTA:



## Costo de las Operaciones:

LISTAVACIA	$\in O(1)$
GUARDAR	$\in O(N)$
INSERTAR	$\in O(N)$
OBTENER	$\in O(N)$
BORRAR	$\in O(N)$
LONG	$\in O(1)$