


ESTRUCTURAS DE DATOS

TIPOS ABSTRACTOS DE DATOS LINEALES

El TAD Lista

Manuel Montenegro Montes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática – Universidad Complutense de Madrid



EL TAD Lista

Es uno de los Tipos Abstractos de Datos más utilizados.

Se usa para mantener todos los objetos dentro de un mismo contenedor en un determinado orden.

- Una **lista** es una colección de elementos, con las siguientes características:

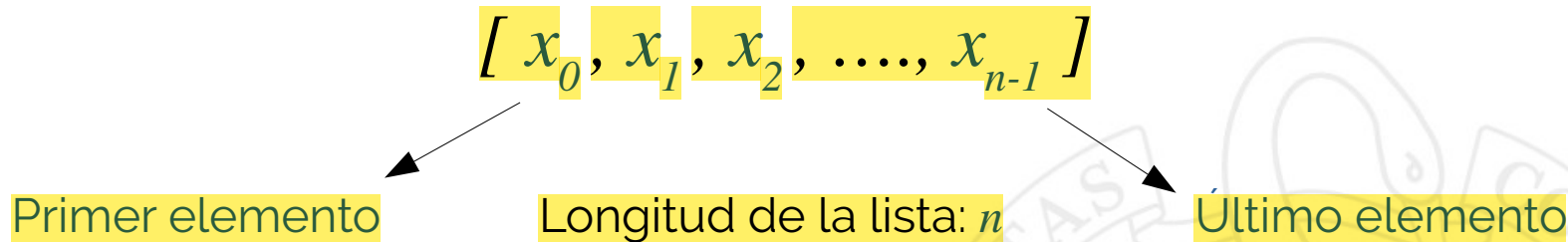
- Tiene longitud finita.
- Los elementos siguen un orden secuencial:
 - Existe un primer elemento y un último elemento.
 - Todos los elementos en la lista tienen un predecesor (excepto el primero) y un sucesor (excepto el último).
- Se permiten elementos duplicados en la lista.

Existe un primer y un último elemento.

Modelo del TAD Lista

Acordarnos que cada TAD tiene asociado un modelo, y para cada modelo podían haber varias representaciones.

- Las listas representan **secuencias** de elementos de la siguiente forma:



- Ejemplos: ejemplos de listas

$[3, 1, 6, 14, 5]$



$[1, 3, 5, 6, 14]$

mismos elementos pero de distinto orden, por lo tanto son listas distintas.

$[25, 25, 7, 5, 7, 7]$

$[Marta, David, Gerardo]$

$[true]$

$[[1, 0, 0], [0, 1, 0], [0]]$

$[]$

Lista vacía (longitud 0)

Operaciones sobre el TAD Lista

- **Constructoras:**

- Crear una lista vacía (**create_empty**).

- **Mutadoras:**

- Añadir un elemento al principio de la lista (**push_front**).
- Añadir un elemento al final de la lista (**push_back**).
- Eliminar el elemento del principio de la lista (**pop_front**).
- Eliminar el elemento del final de la lista (**pop_back**).

- **Observadoras:**

- Obtener el tamaño de la lista (**size**).
- Comprobar si la lista es vacía (**empty**).
- Acceder al primer elemento de la lista (**front**).
- Acceder al último elemento de la lista (**back**).
- Acceder a un elemento que ocupa una posición determinada (**at**).

Esto es muy parecido a las listas de Java

Operaciones constructoras

$\{ true \}$ precondición

create_empty() $\rightarrow (L: List)$

$\{ L = [] \}$ postcondición



Operaciones mutadoras

push= añadir
pop = eliminar.

$\{ L = [x_0, \dots, x_{n-1}] \}$ recibe una lista

push_front(x: elem, L: List)

$\{ L = [x, x_0, \dots, x_{n-1}] \}$

$\{ L = [x_0, x_1, \dots, x_{n-1}], n \geq 1 \}$

pop_front(L: List)

$\{ L = [x_1, \dots, x_{n-1}] \}$

$\{ L = [x_0, \dots, x_{n-1}] \}$

push_back(x: elem, L: List)

$\{ L = [x_0, \dots, x_{n-1}, x] \}$

$\{ L = [x_0, \dots, x_{n-2}, x_{n-1}], n \geq 1 \}$

pop_back(L: List)

$\{ L = [x_0, \dots, x_{n-2}] \}$

Operaciones observadoras

$\{ L = [x_0, \dots, x_{n-1}] \}$

size($L: \text{List}$) \rightarrow (*tamaño*: *int*)

$\{ \text{tamaño} = n \}$

Lista de al menos un elemento.

$\{ L = [x_0, \dots, x_{n-1}] \wedge n \geq 1 \}$

front($L: \text{List}$) $\rightarrow e: \text{elem}$

$\{ e = x_0 \}$ devuelve el primer elemento.

$\{ L = [x_0, \dots, x_{n-1}] \wedge 0 \leq i < n \}$

at($i: \text{int}, L: \text{List}$) $\rightarrow (e: \text{elem})$

$\{ e = x_i \}$ devuelve el elemento de la posición demandada

$\{ L = [x_0, \dots, x_{n-1}] \}$

empty($L: \text{List}$) $\rightarrow b: \text{bool}$

$\{ b \Leftrightarrow n = 0 \}$ booleano que devuelve true sii los elementos de la lista

$\{ L = [x_0, \dots, x_{n-1}] \wedge n \geq 1 \}$

back($L: \text{List}$) $\rightarrow e: \text{elem}$

$\{ e = x_{n-1} \}$ devuelve el último elemento.

Operaciones adicionales

- Algunas implementaciones permiten las siguientes operaciones:
 - Mostrar una lista por pantalla. Esta las vamos a implementar en los videos siguientes
 - Insertar/eliminar en una posición determinada.
 - Concatenar dos listas.
 - Invertir el orden de los elementos de una lista.
 - Recorrer una lista.

Otras de ellas utilizaremos iteradores (que también lo estamos viendo en Java de Tp2) para realizar algunas de las funciones.