

LISTAS

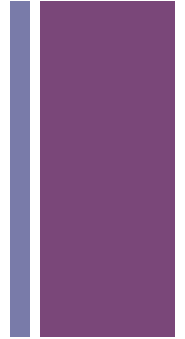
Julio, 2013

Guadalajara, Jalisco. México

Prof. Sabrina Lizbeth Vega Maldonado



Tipo de dato primitivo “C”



- int, float, char, double ...
- Cuando se define una variable se establecen varios hechos:
 1. Nombre de la variable
 2. Cantidad de bytes que ocupa (bytes en memoria y rango)
 3. Operaciones que se pueden realizar sobre esta variable

Ejemplo

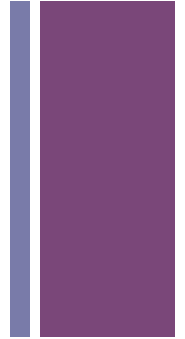
X `char c; c = sqrt (3, 5);`

X `int x; strcmp (“hola”, x);`



TDA

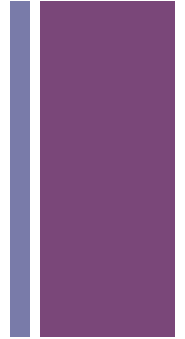
Tipo de Dato Abstracto



- La **abstracción** es la capacidad de definir las características que forman la esencia de las cosas.
- Un TDA es un modelo matemático o lógico organizado donde se definen los datos y operaciones más importantes que corresponden a un objeto.





TDA LISTA



- Una lista es una colección de elementos ordenada por posiciones. Donde todos los elementos de la lista a_i son del mismo tipo.

$$L = [a_1 , a_2 , a_3 , a_4 , \dots , a_n]$$

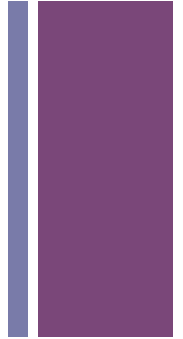
 **primero**  **último**

- n – Longitud de la lista

Nota: observe que la longitud de la lista corresponde al último elemento



ESTRUCTURA DE LA LISTA



```
typedef struct {  
    tipoDato  elem[TAM];  
    int       ultimo;  
} Lista;
```

tipoDato: Es el tipo de los elementos a_i de la lista

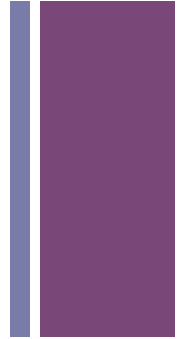
elem: Es la lista de datos (un arreglo).

TAM: Es el tamaño del arreglo que contiene los elementos a_i

ultimo: Es la longitud de la lista



Operaciones de la Lista



CONSTRUCCIÓN

- Inicializar

POSICIONES

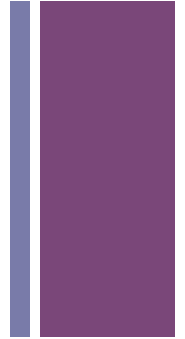
- Primero
- Fin
- Siguiende
- Anterior

CONSULTA

- Vacía
- Llena
- Recupera

MODIFICACION

- Inserta
- Suprime
- Modifica



CONSTRUCCIÓN

- **Inicializar (Lista L)** : Devuelve una lista nueva
 $L.ultimo = -1$

POSICIONES

- **Primero (Lista L)** : Devuelve la posición del primer elemento de la lista, si es que no esta vacía
 -1 : Si esta vacía 0 : Si tiene elementos
- **Fin (Lista L)** : Devuelve la posición fin de lista, que es una posición después del último
 $L.ultimo + 1$
- **Siguiente (Lista L, pos p)** : Devuelve la posición $p+1$
 $p + 1$
- **Anterior (Lista L, pos p)** : Devuelve la posición $p-1$
 $p - 1$



CONSTRUCCIÓN

- Inicializar (Lista L)

$L.ultimo = -1$

POSICIONES

- Primero (Lista L)

0 : Si no esta vacía

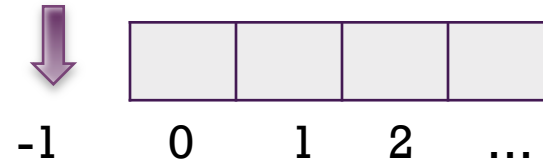
- Fin

$L.ultimo + 1$

- Siguiente (Lista L, pos p)

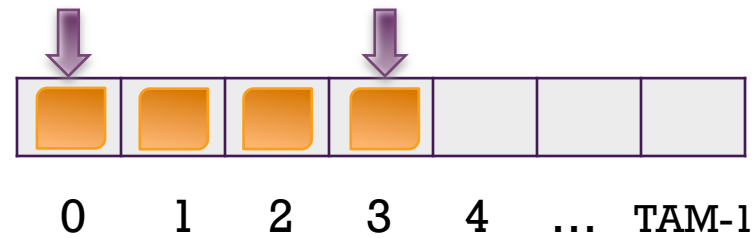
- Anterior (Lista L, pos p)

ultimo



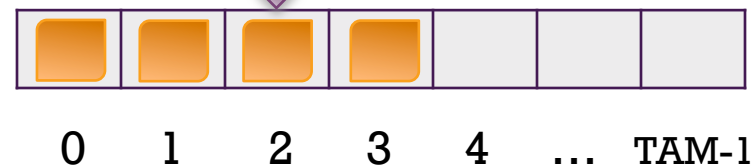
primero

ultimo



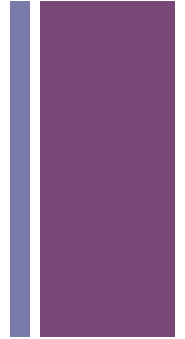
fin

p



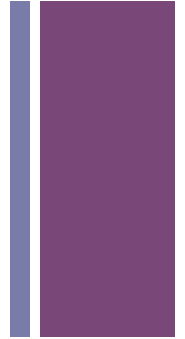
anterior

siguiente



CONSULTA

- **Vacia (Lista L) :** Regresa un valor lógico
VERDADERO (1) cuando la lista No tiene elementos
FALSO (0) cuando la lista tiene elementos
- **Llena(Pila P) :** Regresa un valor lógico
verdadero (1) si el fin (Lista) = TAM
falso (0) si la Pila tiene espacios libres donde insertar mas
elementos
- **Recupera (Lista L, pos p) :** Regresa el elemento que se
encuentra en la posición p
L.elem [p]



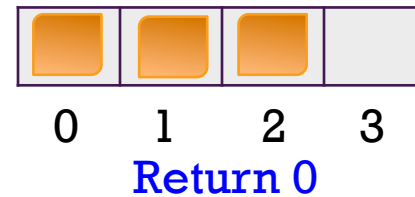
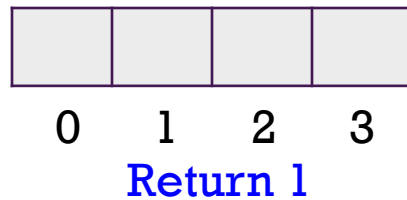
MODIFICACION

- **Inserta (Lista L, tipoDato x, pos p) :** Desplaza una posición hacia la derecha todos los elementos a partir de la posición p y finalmente inserta el elemento x en la posición p
- **Suprime (Lista L, pos p) :** Elimina de la lista el elemento que esta en la posición p desplazando todos los elementos una posición hacia la izquierda hasta la posición p
- **Modifica (Lista L, tipoDato x, pos p) :** Reemplaza el elemento de la posición p con el elemento x

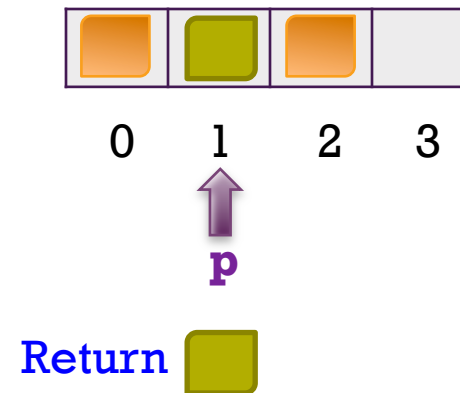
+

CONSULTA

■ Vacia (Lista L)




■ Recupera (Lista L, pos p)
return L.elem[p]

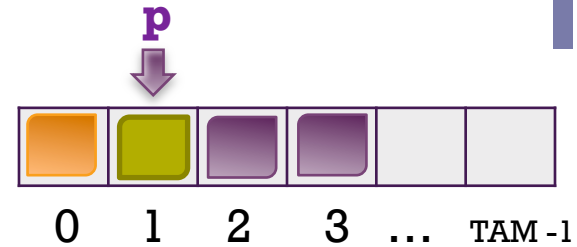
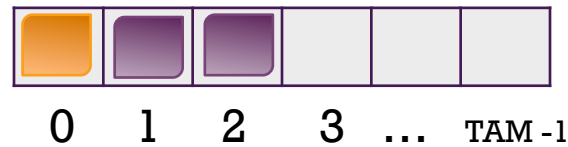




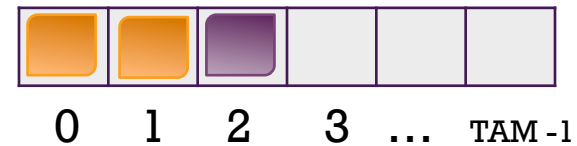
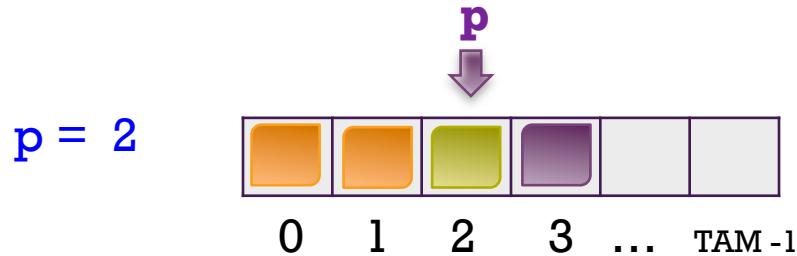
MODIFICACION

■ Inserta (Lista L, tipoDato x, pos p)

x = 
p = 1



■ Suprime (Lista L, pos p)



■ Modifica (Lista L, tipoDato x, pos p)

