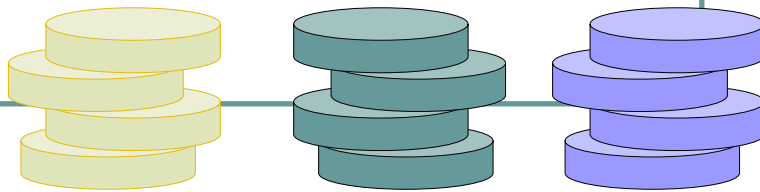


# Pilas (Stacks)

## Estructuras de Datos



1

## Pilas

- Apilar: Poner una cosa sobre otra haciendo **pila** (ll montón)
- **Pila**: Montón, rimerio o cúmulo que se hace poniendo una sobre otra las piezas o porciones de que consta algo.  
*Ejemplo: Pila de lana, de ladrillos.*

[www.rae.es](http://www.rae.es)

2


# Pilas




Libros "apilados"



Pila de Platos



Pila de sillas



Pila de gatos....

3

## Informática: Pila

- Es una lista ordinal en la que el modo de acceso a sus elementos es de tipo **LIFO**
- LIFO (*Last In First Out*, es decir, "último en entrar, primero en salir") que permite almacenar y recuperar datos.

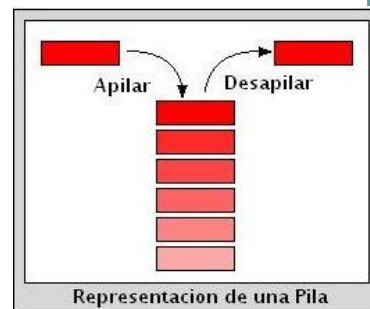


Imagen: [www.wikipedia.org](http://www.wikipedia.org)

4

## Pilas

- Para el manejo de los datos se cuenta con dos operaciones básicas:
  1. "apilar" (*push*), que coloca un objeto en la pila, y
  2. su operación inversa, "desapilar" (*pop*), que retira el último elemento apilado.

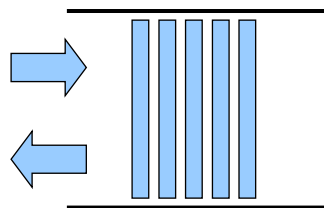


Imagen: [www.wikipedia.org](http://www.wikipedia.org)

5

## La pila

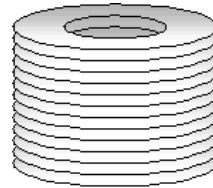
- Es un contenedor que utiliza el protocolo **LIFO** (Last In, First Out).



6

## Pilas

Son un subconjunto de las listas, en donde las eliminaciones e inserciones se dan en un solo extremo, de manera tal que, el último elemento es el único accesible.



7



- En cada momento sólo se tiene acceso al **tope** de la pila, es decir, al último objeto apilado (**TOS**, *Top of Stack*).
- La operación **pop** permite la obtención de este elemento, que es retirado de la pila permitiendo el acceso al siguiente (apilado con anterioridad), que pasa a ser el nuevo tope.



8

## Utilización de pilas

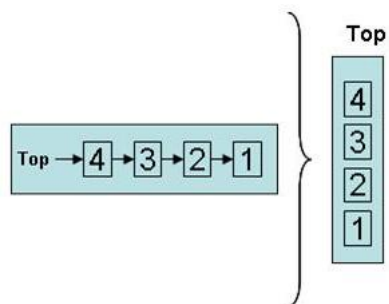
- Las pilas suelen emplearse en los siguientes contextos:
  1. Evaluación de expresiones en notación postfija (notación polaca inversa).
  2. Reconocedores sintácticos de lenguajes independientes del contexto.
  3. Implementación de recursividad.
  4. Pilas hardware, un uso muy común de las pilas a nivel de arquitectura hardware es la asignación de memoria.

9

## Representación

Estática a través de arrays, o enlazada a través de punteros.

En los dos casos, se debe tener claro el tope de la pila, por donde se ingresan y retiran los elementos.



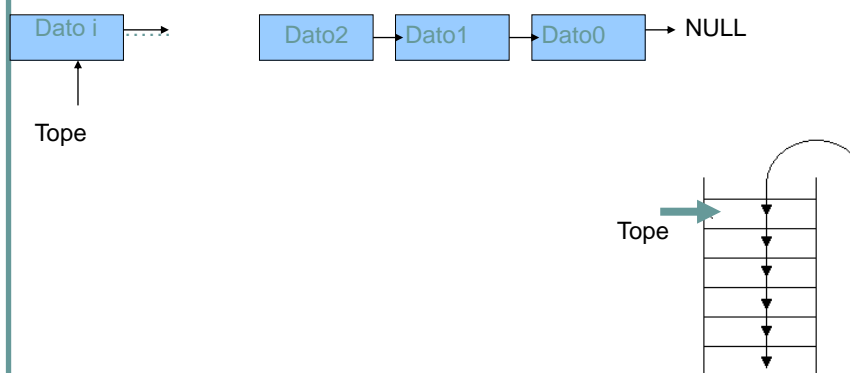
10

## Operaciones comunes...

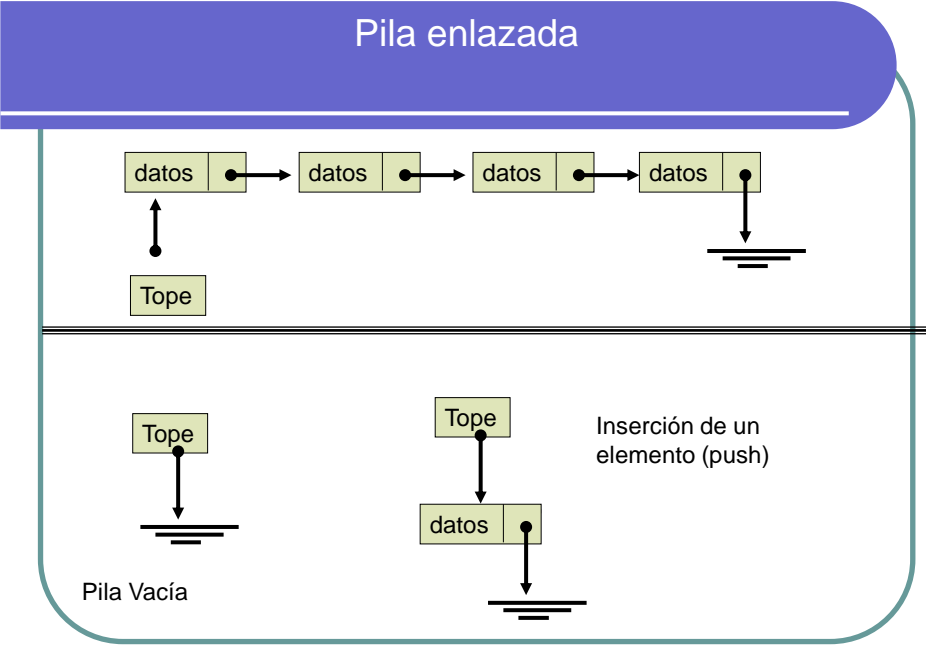
- **Push**
- **Pop**
- **top()**: informa del último elemento ingresado en la pila sin removerlo de ella.
- **size()**: informa el número de elemento
- **Empty()**: Boolean, indica V si no hay elementos almacenados en la pila.

11

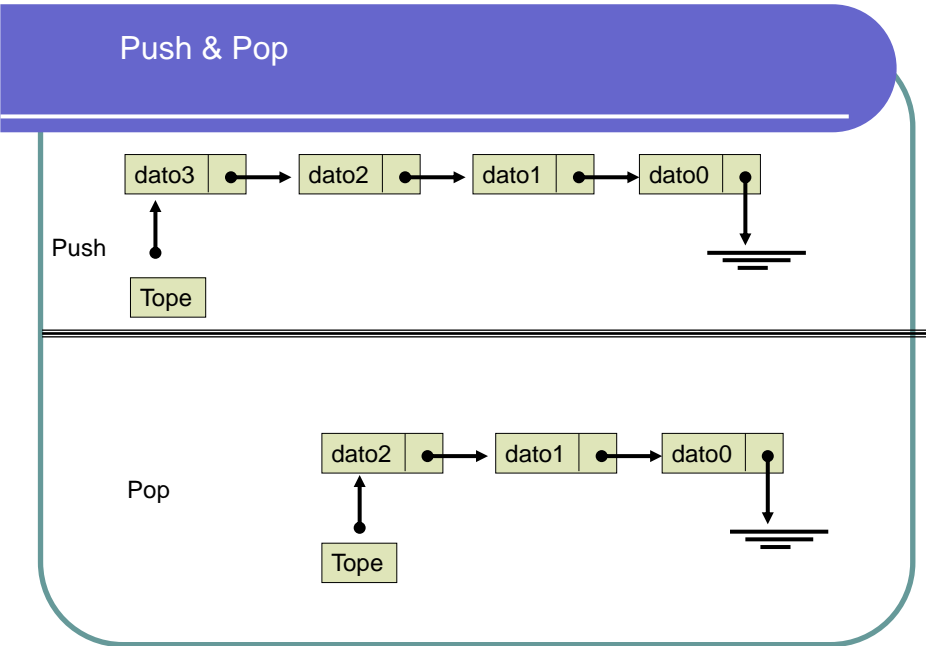
## Implementación dinámica de Pilas



12



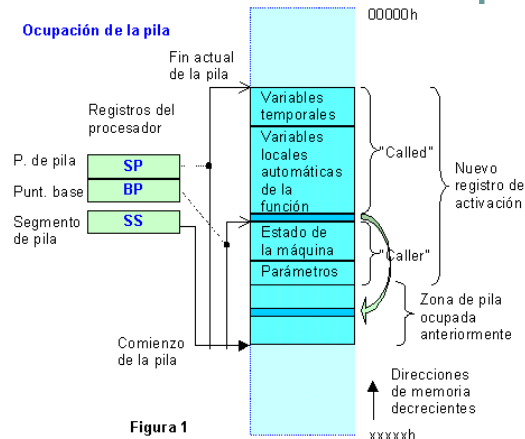
13



14

## Aplicación en Compiladores

La mayoría de los lenguajes de programación son de contexto libre de los idiomas que les permite ser analizados con máquinas basadas en la pila.



15

## Recursividad y Pilas

- Suponga la función recursiva:

```
int factorialRecursivo(int N)
{
    if (N==0) return(1)
    else return(N*factorialRecursivo(N-1);
}
```

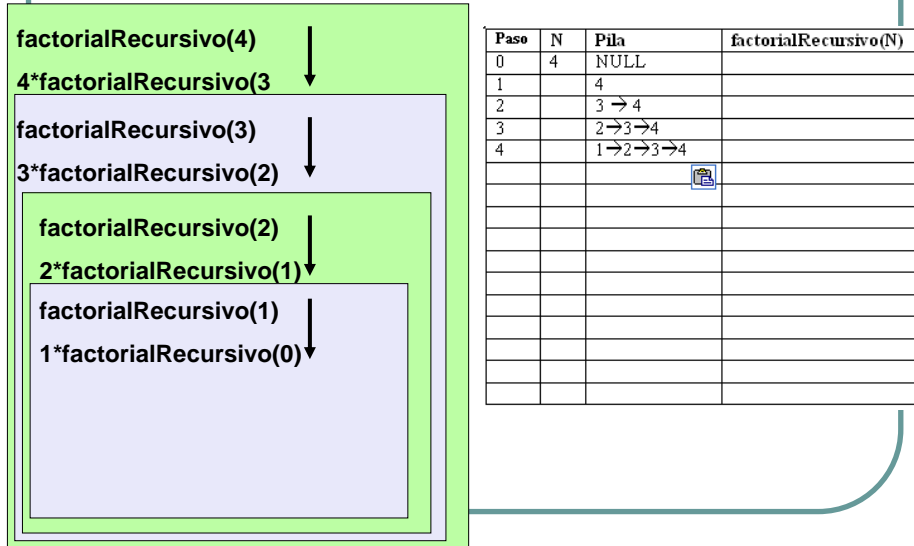
16





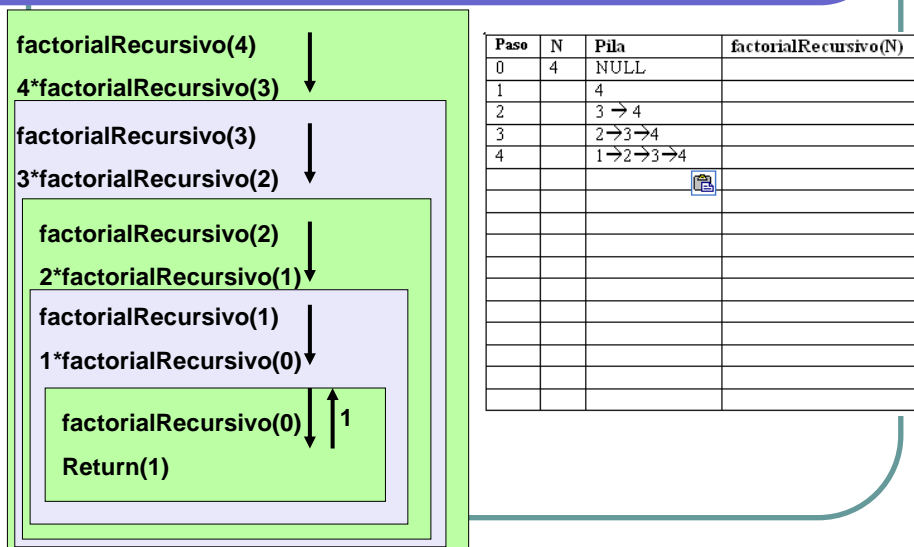


# Recursividad y Pilas



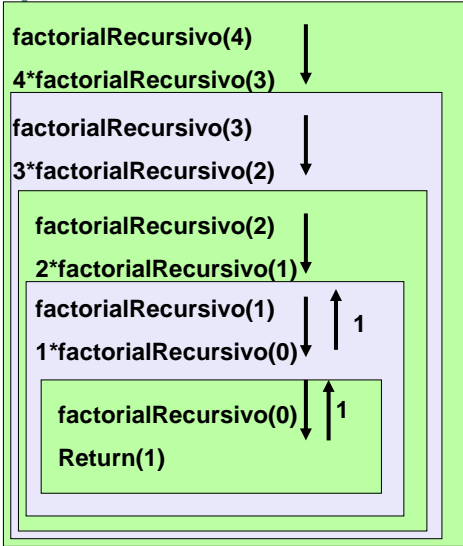
21

# Recursividad y Pilas



22

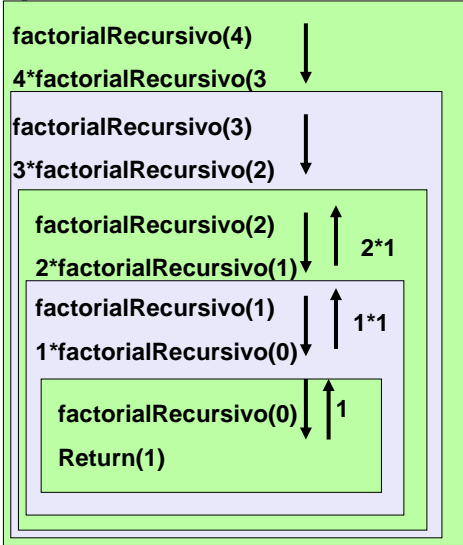
# Recursividad y Pilas



Paso	N	Pila	factorialRecursivo(N)
0	4	NULL	
1		4	
2		4→3	
3		4→3→2	
4		4→3→2→1	
5			1
6		4→3→2	1*1

23

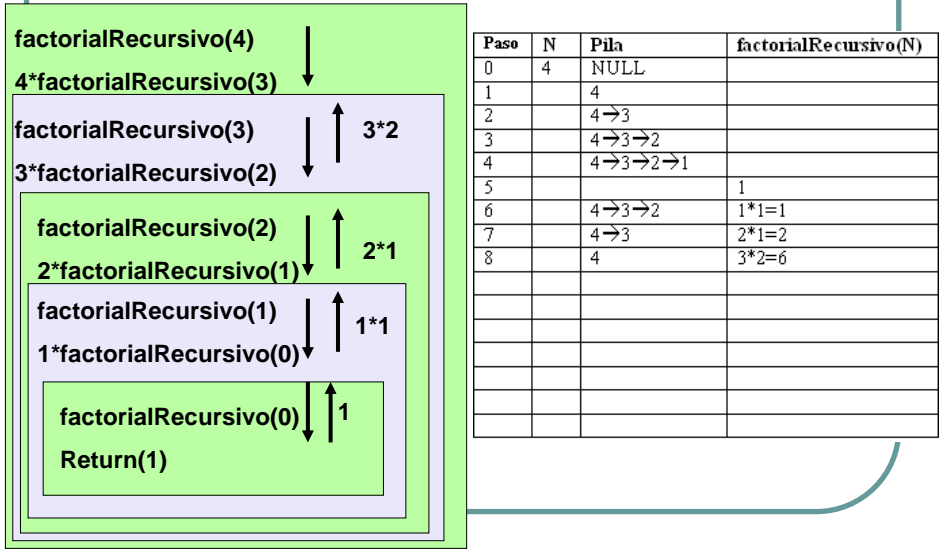
# Recursividad y Pilas



Paso	N	Pila	factorialRecursivo(N)
0	4	NULL	
1		4	
2		4→3	
3		4→3→2	
4		4→3→2→1	
5			1
6		4→3→2	1*1=1
7		4→3	2*1=2

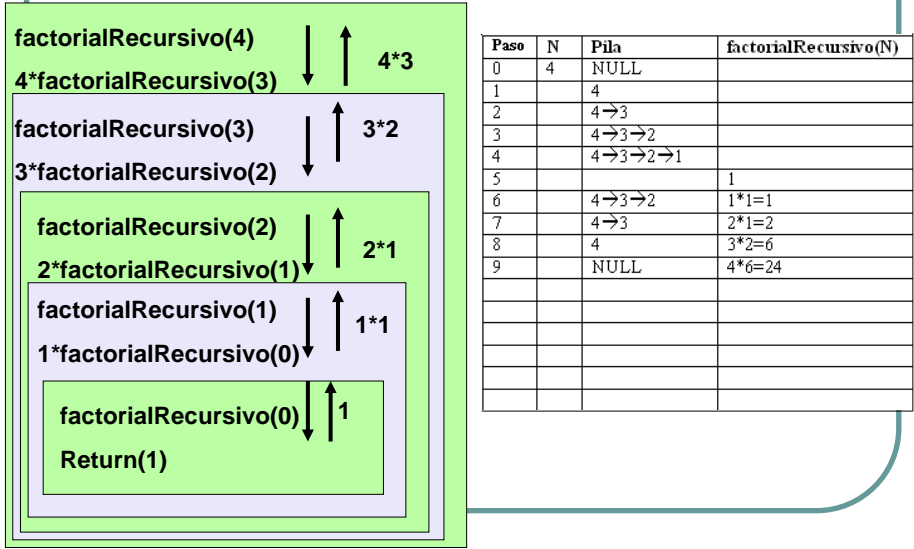
24

# Recursividad y Pilas



25

# Recursividad y Pilas



26



27

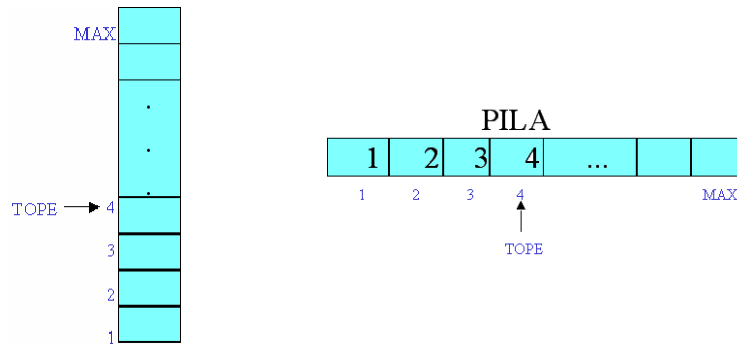
# Recursividad

Entonces, un algoritmo para hallar el factorial de un número utilizando pilas será:

```
Lpila = NULL;  
leer n;  
mientras n > 1  
    push (Lpila, n);  
    n = n-1;  
fin mientras  
factorial = 1;  
mientras pila no está vacía  
    factorial = factorial * pop (Lpila);  
fin mientras
```

28

## Implementación de Pilas con arrays

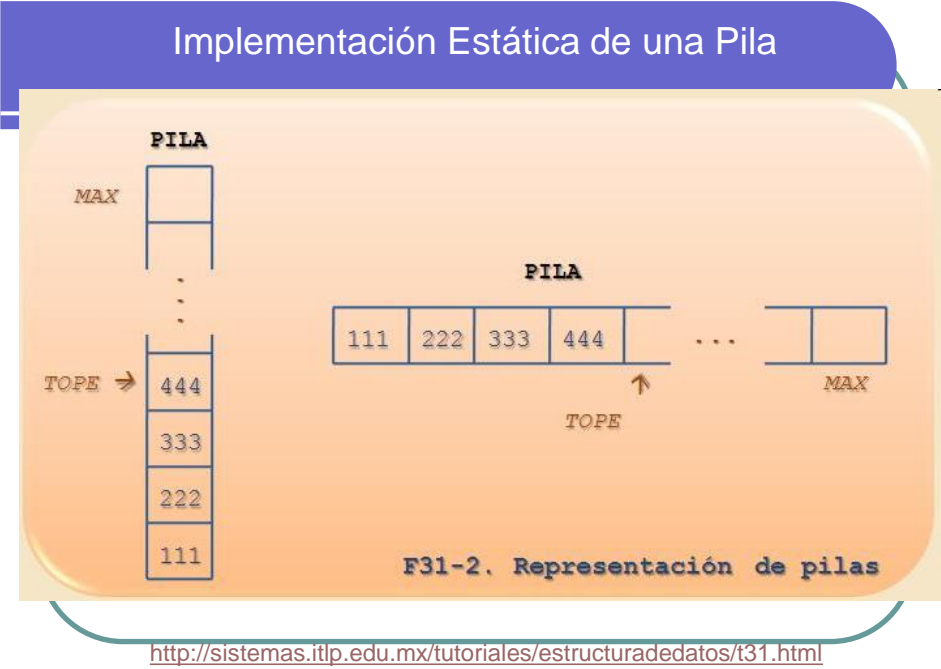


29

## Implementación mediante array

- Al ser **estática**, da un tamaño máximo fijo a la pila. Si se sobrepasa dicho límite se produce un error.
- Es necesario comprobar las operaciones de apilado en una pila llena o desapilado en una pila vacía.

30



31

### Pila con estructura estática

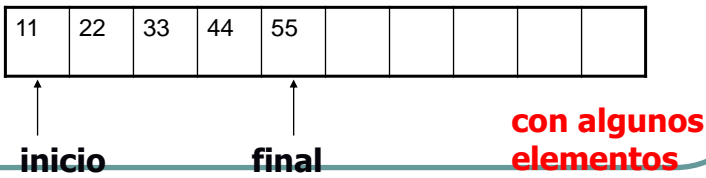
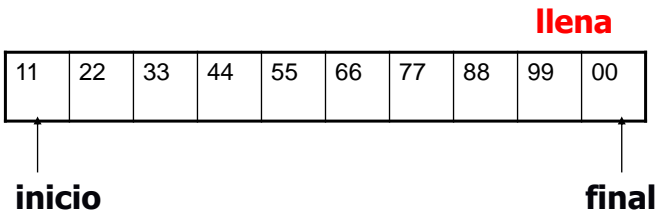
Es una pila implementada como un arreglo

The diagram shows a static stack implemented as an array. The array has indices 0, 1, 2, 3, 4, ..., **max**. The first four cells contain the values 11, 22, 33, and 44. An arrow labeled **inicio** points to the first cell (index 0), and an arrow labeled **final** points to the fourth cell (index 3).

32

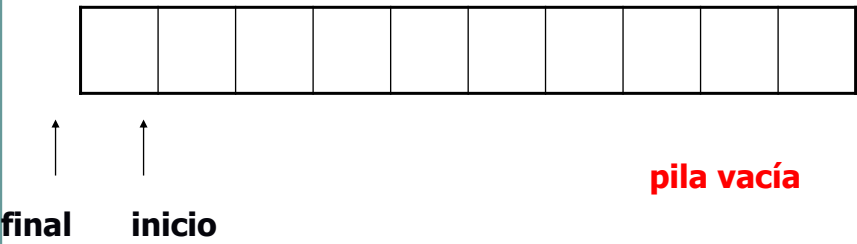


pilas: Casos posibles:

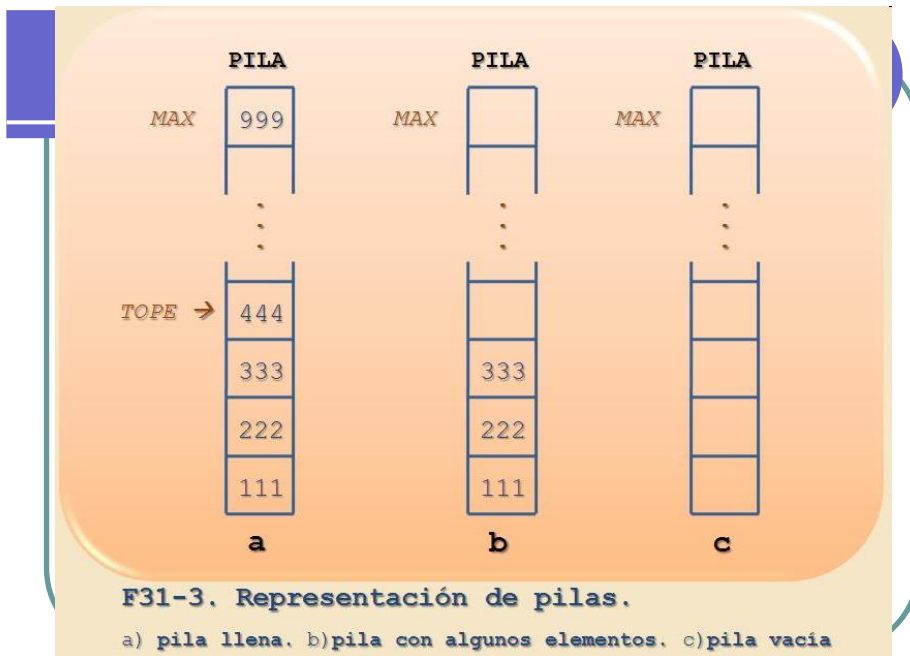


33

pilas: Casos posibles:



34



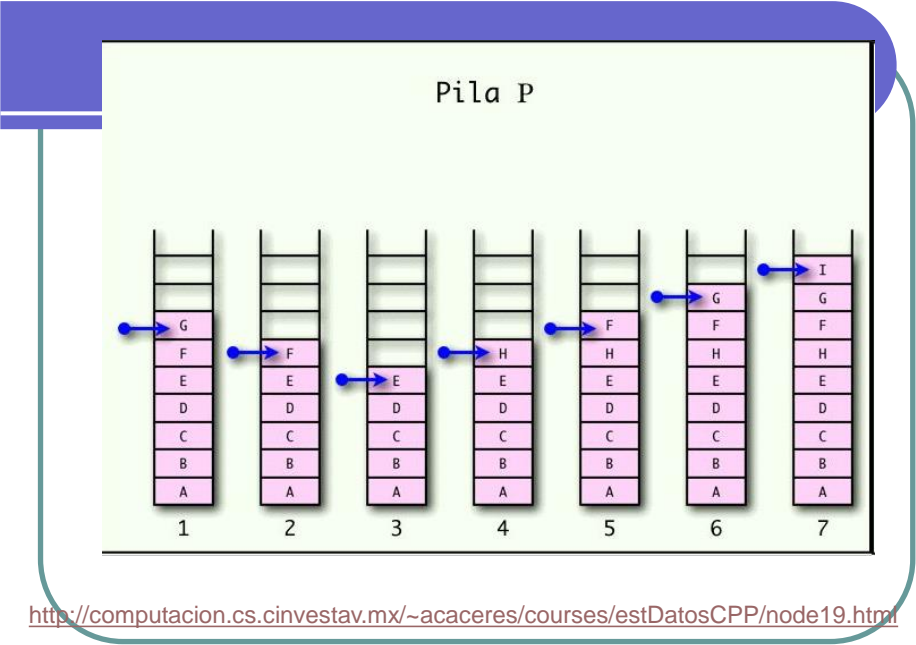
<http://sistemas.itlp.edu.mx/tutoriales/estructuradedatos/t31.html>

35

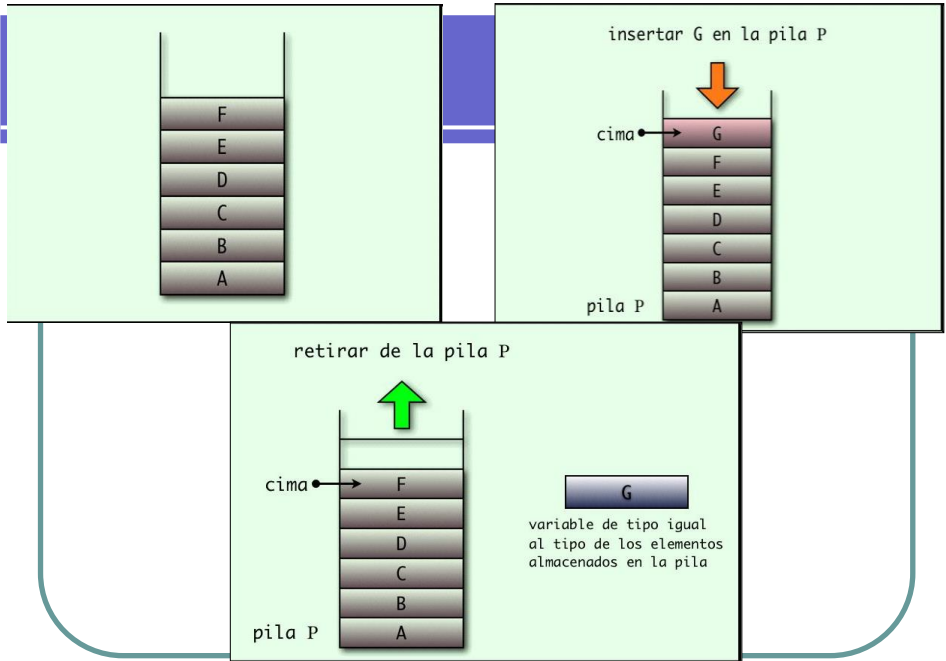
## Desbordamiento: *overflow*

- Si la pila está llena y se intenta insertar un nuevo elemento, se produce un error conocido como **desbordamiento** —*overflow*—.
- Otro error es tratar de eliminar un elemento de una pila vacía. Este tipo de error se conoce como **subdesbordamiento** —*underflow*—.

36



37



38

## Ejercicio:

Para cada problema escriba un **algoritmo** para resolver el problema utilizando sólo pilas:

1. La famosa sucesión de Fibonacci puede definirse en términos de recurrencia de la siguiente manera:

$$\begin{aligned} (1) & \text{ Fib}(1) = 1 ; \text{ Fib}(0) = 0 \\ (2) & \text{ Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \text{ si } n \geq 2 \end{aligned}$$

Considere  $N=6$  (determine el sexto término de la serie).

39

## Ejercicios:

1. Escriba una función C que permita verificar que en una cadena de caracteres existen pares de paréntesis: {}, [], () sintácticamente correctos.

Cadenas como:  $((\{[x]+y\}^*z)+2)-5$ ;  $(\{\}\{\}\{\})$  son correctas, en tanto que otras como:  $\{ \{, [ \{, \{\}, (\{ \}$  son incorrectas

Implemente una versión estática y otra versión dinámica. En ambos casos, debe permitir leer varias cadenas y determinar para cada una de ellas su correctitud.

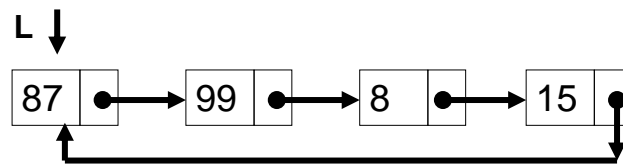
40

## Lista enlazada circular

- En una lista enlazada circular, el primer y el último nodo están unidos juntos.
- Para recorrer una lista enlazada circular se puede comenzar por cualquier nodo y seguir la lista hasta que se regrese hasta el nodo original.
- Este tipo de listas es el más usado para visitar todos los nodos de una lista a partir de uno dado.

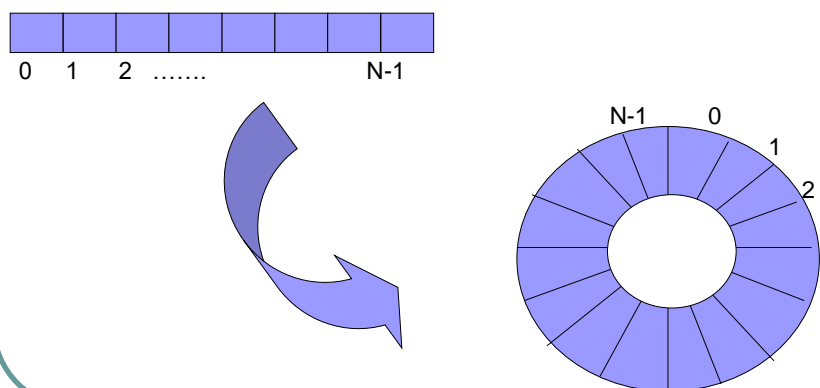
41

## Listas Circulares



42

## Listas circulares: implementación estática



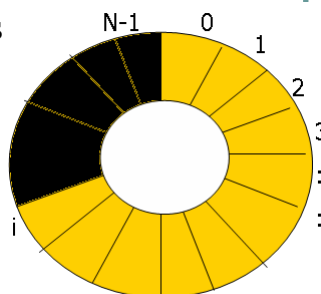
43

## Listas circulares, implementación estática

- **Recorrido circular:**

desde la posición 0 hasta la N-1, desde la N-1 continua con la posición 0.... Cuando está llena la capacidad del vector, si tiene menos elementos, entonces....

El recorrido es desde la posición 0 hasta la posición i, desde la posición i continúa con la posición 0....



44

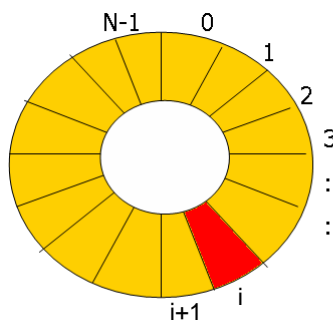
## Listas circulares, implementación estática

- Overflow: hay más elementos que capacidad de almacenamiento.

45

## Listas circulares, implementación estática

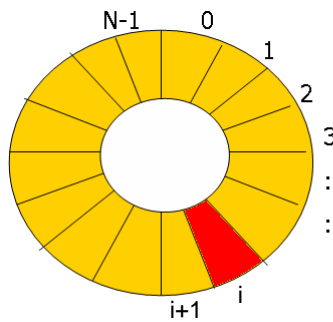
- Eliminación: obliga a desplazar hacia a la izquierda a todos los elementos del array.



46

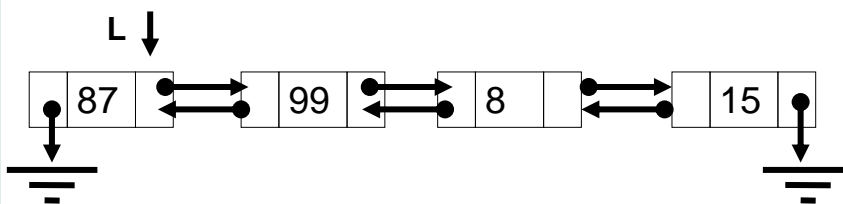
## Listas circulares, implementación estática

- Inserción: obliga a desplazar hacia la derecha a los elementos del array.



47

## Listas Doblemente Encadenadas



48