ESTRUCTURAS DE DATOS

TIPOS ABSTRACTOS DE DATOS LINEALES

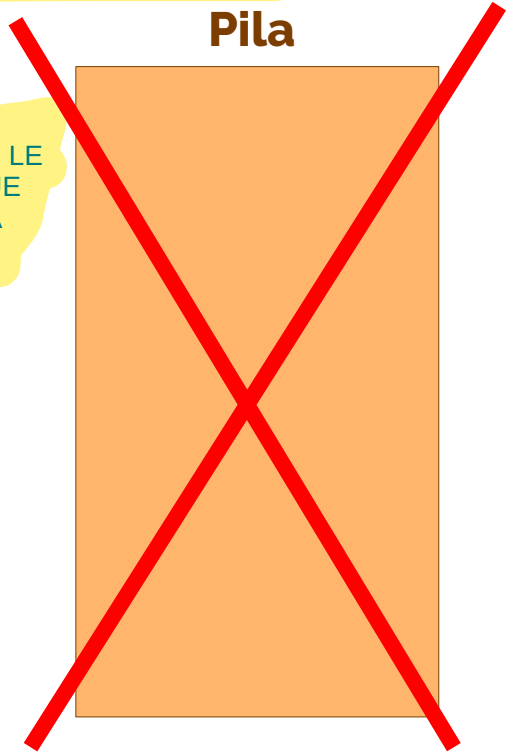
# El TAD Pila

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

# ¿Qué es una pila?

No vamos a hablar de esto

IMPORTANTE, NO NOS REFERIMOS A ESTA PILA!!!!!




Heap

Pero más o menos A ESTA PILA SE LE LLAMA DE ESTA MANERA PORQUE SE IMPLEMENTA COMO UNA PILA

Es un TAD que aparentemente es muy limitado, pero tiene mucha utilidad.

# ¿Qué es una pila?

- Es una colección de elementos que permite:
  - Insertar elementos.
  - Obtener o borrar el último elemento insertado que no haya sido borrado previamente.

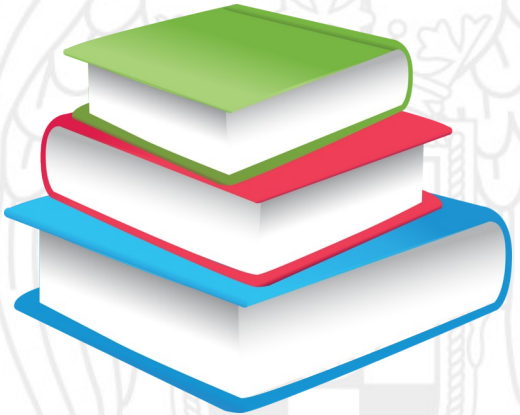
Se llama pila porque nos recuerda al resultado de apilar una seri de objetos.

top()

Vamos eliminando los de arriba.

Foto: John Leffmann (CC BY 3.0)

Apilamos desde abajo



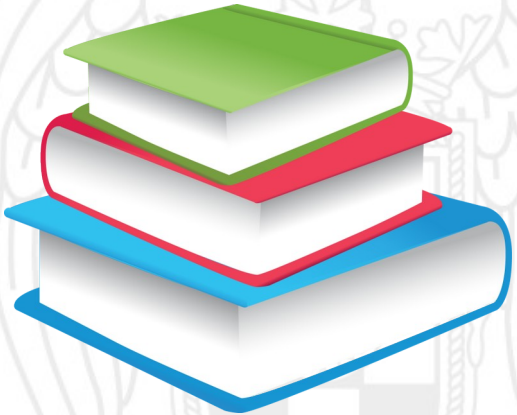
# ¿Qué es una pila?

- Las pilas reciben el nombre de estructuras de acceso **LIFO**

**L**ast **I**n, **F**irst **O**ut

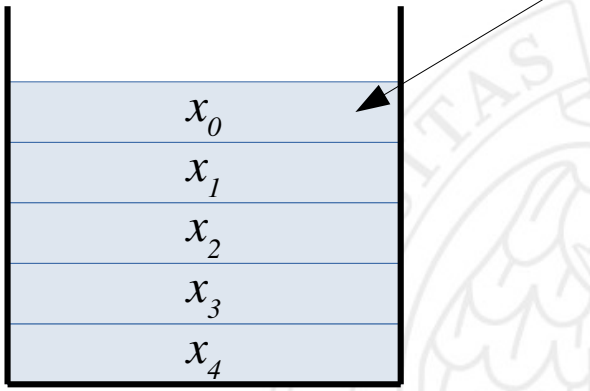
El último que entra es el primero que sale. A diferencia de los FIFOS que nosotros habamos en gestión empresarial o sistemas operativos.

Foto: John Leffmann (CC BY 3.0)



# Modelo de pilas

- Conceptualmente representamos las pilas de esta forma:




**Cima** de la pila: último elemento insertado.

Único al que podemos acceder.

Elementos metidos en un recipiente.

# Operaciones sobre pilas

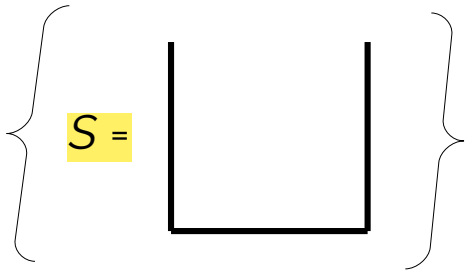
- **Constructoras**:
  - Crear una pila vacía (***create_empty***).
- **Mutadoras**:
  - Añadir elemento en la cima de la pila (***push***).
  - Eliminar elemento en la cima de la pila (***pop***).
- **Observadoras**:
  - Obtener el elemento en la cima de la pila (***top***).
  - Saber si una pila está vacía (***empty***).

Como en las listas enlazadas tanto simples como doblemente enlazadas circulares PERO SIEMPRE SOBRE EL ELEMENTO DE LA CIMA EN ESTE CASO.

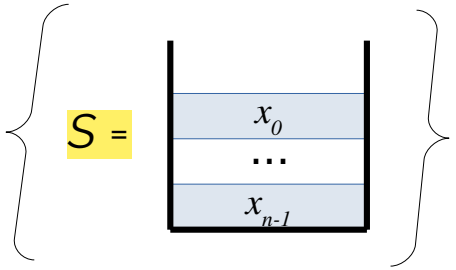
# Operación *create_empty*

*{ true }*

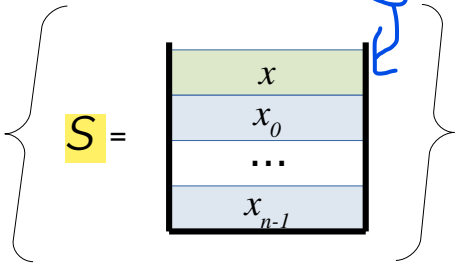
***create_empty***() → *(S: Stack)*



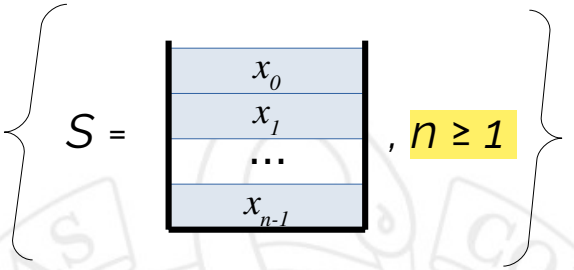
# Operaciones *push* y *pop*



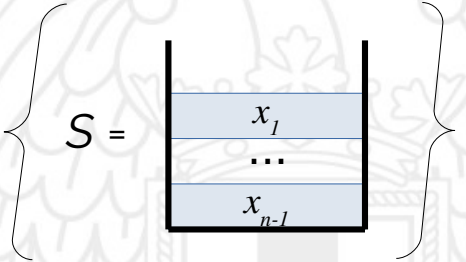

***push***(S: Stack, x: elem) APILAR



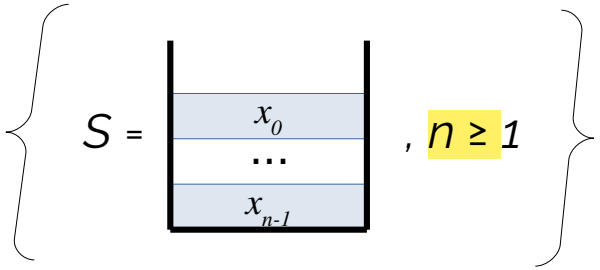

También mover top()




***pop***(S: Stack)

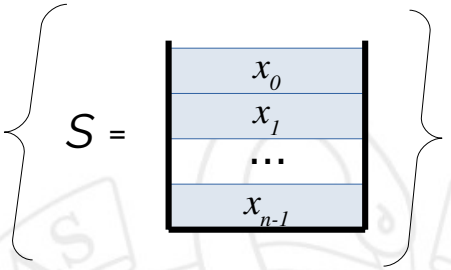


# Operaciones *top* y *empty*




***top***(S: Stack) → (x: elem)

{ $x = x_0$ }




***empty***(S: Stack) → (b: bool)

{ b ⇔ n = 0 }