# TEMA 7. EL TDA PILA

**7.1. Definición**

**7.2. Operaciones del TDA PILA**

**7.3. Implementaciones más usuales: vectores y listas enlazadas**

**7.4. Utilidades de las Pilas**

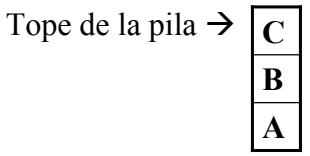
**Bibliografía**

- **Weiss**
- **Cairó y Guardati**
- **Aho, Hopcroft y Ullman**

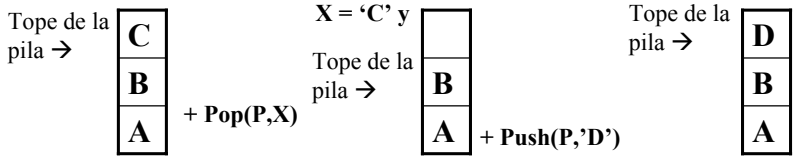
## 7.1. Definición

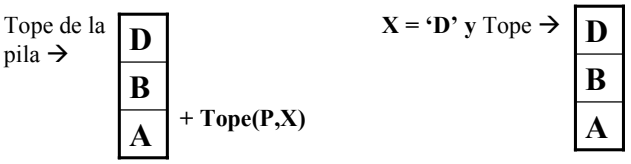
- Las pilas son estructuras que permiten almacenar elementos, con la particularidad de que todas las inserciones y borrados se hacen por el punto denominado tope de la pila.

- Los tipos de dato PILA son equivalentes a las pilas de platos, discos, etc. que vemos a diario.
- La memoria de los programas, dentro de un ordenador, se almacenan siguiendo la forma de una PILA.



- Las tres operaciones elementales son: **push, pop y tope**, son muy sencillas. No obstante, son muy útiles en programación:
  - Permiten simplificar algoritmos complejos.
  - Permiten accesos muy rápidos (los algoritmos que los usan no pierden tiempo en los accesos).




# 6.2. Operaciones del TDA Pila

- **Crear_pila** (P: Pila, ok: lógico)
  {P: parámetro resultado; genera una pila vacía;
  ok: parámetro resultado, cierto/falso si todo fue o no bien}
- **Borrar_pila** (P: Pila, ok: lógico)
  {P: parámetro resultado; borra todos los elementos de la pila;
  ok: parámetro resultado, cierto/falso si todo fue o no bien}
- **Vacía?** (P: Pila, resp: lógico)
  {P: parámetro dato;
  resp: parámetro resultado, cierto/falso si la pila está o no vacía}
- **Llena?** (P: Pila, resp: lógico)
  {P: parámetro dato;
  resp: parámetro resultado, cierto/falso si la pila está o no llena}

- **Push** (P: Pila, x: ELEMENTO, resp: lógico)
  {P: parámetro dato-resultado; se modifica el tope de la pila.
  x: parámetro dato, dato a introducir en el tope de la pila,
  resp: parámetro resultado, cierto/falso si la operación pudo o no realizarse}
- **Pop** (P: Pila, x: ELEMENTO, resp: lógico)
  {P: parámetro dato-resultado; se modifica el tope de la pila.
  x: parámetro resultado, dato que ocupa el tope de la pila,
  resp: parámetro resultado, cierto/falso si la operación pudo o no realizarse}
- **Tope** (P: Pila, x: ELEMENTO, resp: lógico)
  {P: parámetro dato; no se modifica el tope de la pila.
  x: parámetro resultado, dato que ocupa el tope de la pila,
  resp: parámetro resultado, cierto/falso si la operación pudo o no realizarse}

## 7.3. Implementaciones más usuales: vectores y listas enlazadas

## 7.3.1. Implementación con vectores

```
#define MÁXIMO 256

typedef int LOGICO;
typedef char ELEMENTO;

struct t_pila {
int tope;
ELEMENTO datos [MAXIMO];
}
typedef struct t_pila PILA;
```

**Cabeceras en C de las operaciones del TDA Pila**

- LOGICO **crear_pila (**PILA *P**)**
- LOGICO **borrar_pila** (PILA *P)
- LOGICO **vacía?** (PILA P)
- LOGICO **llena?** (PILA P)
- LOGICO **push** (PILA *P, T x)
- LOGICO **tope** (PILA P, T *x)
- LOGICO **pop** (PILA *P, T *x)

## 7.3.2. Implementación con listas enlazadas

**A) TDA Pila como TDA Lista**

- push(), top() y pop() se implementan en función de primero(), insertar(), eliminar(), anterior(), sgte(),...

**B) TDA Pila directamente con listas enlazadas**

- Es posible simplificar la representación anterior, ya que sólo interesa saber dónde está el primer elemento.

{Declaración de tipos}
**ELEMENTO** = T;
**POSICIÓN** = puntero a NODO;
**NODO** = registro de
    info: ELEMENTO;
    sgte: puntero a NODO;
finregistro;
PILA = registro de
    longitud: numérico;
    prim: puntero a NODO;
finregistro;

```
{Nec: Una pila y un elemento.
Prod.: Pone el elemento en el tope de la pila.}
Algoritmo Push (P, x, resp) es
P: Pila; {par. dato-resultado}
x: ELEMENTO; {p. dato}
resp: lógico; {p. resultado}
tmp: posición; {variable local}
Inicio
    llena? (P, resp);
    si resp = cierto entonces {La pila está llena. No se puede insertar}
            resp:= falso; Escribir "Pila llena";
    sino    obtener (tmp);
            tmp→.info := x;
            tmp→.sgte := P.prim; {puede ser nil}
            P.prim := tmp;
            P.longitud := P.longitud + 1;
            resp := cierto;
    finsi
Fin
```

```
{Nec: Una pila con, al menos, un elemento.
Prod.: Saca el elemento del tope de la pila.}
Algoritmo Pop (P, x, resp) es
P: Pila; {p. dato-resultado}
x: ELEMENTO; {p. resultado}
resp: lógico; {p. resultado}
tmp: posición;
Inicio
    vacía? (P, resp);
    si resp = cierto entonces {La pila está vacía. No se puede sacar}
            resp := falso; Escribir "Pila vacía";
    sino
            tmp := P.prim;
            P.prim := tmp→.sgte; {que puede ser nil}
            P.longitud := P.longitud – 1;
            x := tmp→.info;
            liberar(tmp);
            resp := cierto;
    finsi
Fin
```

```
{Nec: Una pila.
Prod.: El elemento que ocupa el tope de la pila}
Algoritmo Tope (P, x, resp) es
P: Pila; {p. dato}
x: ELEMENTO; {parámetro resultado}
resp: lógico; {p. resultado}
Inicio
    vacía? (P, resp);
    si resp = cierto entonces {La pila está vacía}
            resp := falso;
            Escribir "Pila vacía";
    sino {Hay al menos un elemento}
            x := P.prim→.info;
            resp:=cierto;
    finsi
Fin
```