

# UNIDAD I. ESTRUCTURA DE DATOS BÁSICAS (PILAS)

Francisco J. Hernández López

fcoj23@cimat.mx

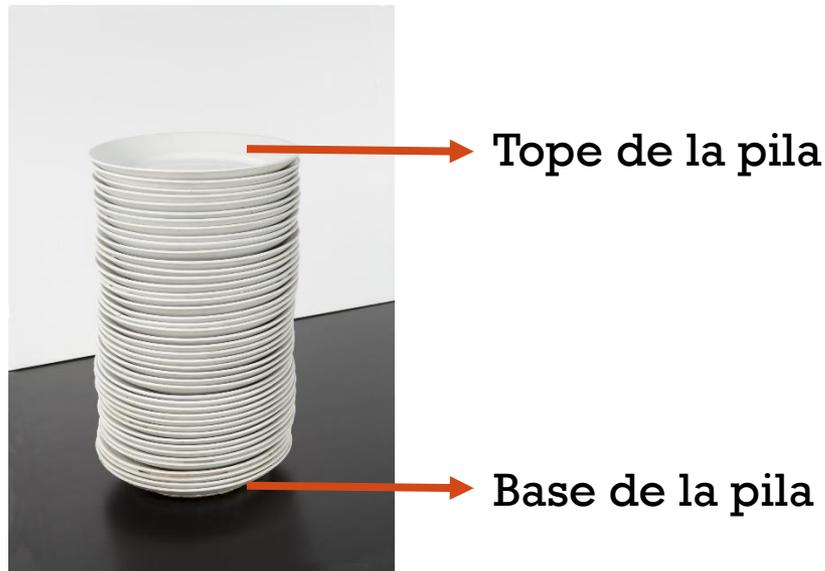


# ESTRUCTURAS DE DATOS

- **Lineales**
  - Arreglos → Colección ordenada de datos del mismo tipo y que están almacenados en memoria contigua o posiciones adyacentes
  - Pilas
  - Colas
  - Listas
  
- **No lineales**
  - Árboles
  - Grafos

# PILAS

- Estructura lineal a cuyos datos solo se puede acceder por un extremo, llamado tope o cima (top)
- Es una estructura tipo LIFO (Last In First Out)



## Operaciones:

- Meter (push) → Añadir un elemento al final de la pila
- Sacar (pop) → Leer y eliminar un elemento del final de la pila

# IMPLEMENTACIÓN DE PILAS

- Memoria Estática
  - Fijar el tamaño máximo de la pila
  - Solo es necesario una variable que controle las operaciones (tope)
  - Overflow → Si la pila está llena y se intenta insertar un nuevo elemento
  - Uso ineficiente de la memoria → asignar más memoria de la que realmente necesitamos
- Memoria Dinámica
  - Uso eficiente de la memoria sin overflow
  - Cada elemento necesita un espacio más en memoria para guardar el apuntador al siguiente elemento

En los dos casos se puede presentar:

Underflow → Si la pila está vacía y se intenta eliminar un elemento

# PSEUDOCÓDIGO PARA METER UN ELEMENTO A UNA PILA

- Meter dato  $x$  en la pila (push)
  1. Inicio
  2. Si  $(\text{tope} == \text{MaxTam} - 1)$  entonces
  3.     Escribir “La Pila está Llena...”
  4. Si no
  5.      $\text{tope} \leftarrow \text{tope} + 1$
  6.      $\text{pila}(\text{tope}) \leftarrow x$
  7.     Fin\_si
  8. Fin

# PSEUDOCÓDIGO PARA SACAR UN ELEMENTO DE UNA PILA

- Sacar un dato de la pila (pop)
  1. Inicio
  2. Si  $(\text{tope} == -1)$  entonces
  3.     Escribir “La Pila está Vacía...”
  4. Si no
  5.     Escribir  $\text{pila}(\text{tope})$
  6.      $\text{tope} \leftarrow \text{tope} - 1$
  7.     Fin\_si
  8. Fin

# PROGRAMAR UNA PILA USANDO MEMORIA ESTÁTICA...

# PROGRAMAR UNA PILA USANDO MEMORIA DINÁMICA...

# APLICACIONES DE LAS PILAS

- Compiladores y SO (comprobar sintaxis)
- Llamadas a subprogramas
- Tratamiento de expresiones aritméticas (prefija, infija, postfija)
- Navegadores de internet (sitios recientemente visitados)
- Editor de textos (borrado de caracteres)
- Etc...

# COMPROBAR SINTAXIS DE PARÉNTESIS, CORCHETES Y LLAVES EN C

```
#include <iostream>
#include <stdlib.h>
#include <stdio.h>

/* run this program using the console pauser or add you
```

```
#define MaxTam 5

void push(int pila[], int *tope, int x);
int pop(int pila[], int *tope);
```

```
int main(int argc, char** argv {
    int pila[MaxTam];
    int tope=-1; // -1 --> Indica que la pila está vacía
```

```
    push(pila, &tope, 10);
    push(pila, &tope, 20);
```

```
    printf("\nElemento %d: ", tope);
    printf("%d ", pop(pila, &tope));
    printf("\nElemento %d: ", tope);
    printf("%d ", pop(pila, &tope));
    printf("\nElemento %d: ", tope);
    printf("%d ", pop(pila, &tope));
```

```
    system("pause");
    return 0;
```

}

1. Iniciamos la lectura del archivo .c o .cpp carácter por carácter y creamos una pila

2. Si el carácter es: (, [ o { entonces

2.1 Se realiza un push(pila, carácter)

3. Si el carácter es: ), ] o } entonces

3.1. Si se cancela con el carácter del tope de la pila entonces

3.1.1. Se realiza un pop(pila)

3.2. Si no

3.2.1. Se realiza push(pila, carácter)

# TRATAMIENTO DE EXPRESIONES ARITMÉTICAS

- Prefija:  $+ * a b * c d$  (operador operando operando)
- Infija:  $a * b + c * d$  (operando operador operando)
- Postfija:  $a b * c d * +$  (operando operando operador)

Operador	Símbolo	Lugar de Prioridad
Paréntesis	()	0
Potencia	^	1
Multiplicación y División	* /	2
Suma y Resta	+ -	3

# CONVERTIR INFIJA → POSTFIJA USANDO UNA PILA

1. Leer cada elemento de la entrada (del inicio al final de la entrada)
2. Si es un operando entonces
  3. Se manda a la salida
4. Si es un "(" entonces
  5. push(pila,"(")
6. Si es un ")" entonces
  7. pop(pila) → va a la salida hasta encontrar un "("
  8. pop(pila) → para eliminar el "("
9. Si es un operador entonces
  10. pop(pila) todos los operadores de la pila con prioridad mayor al operador leído
  11. push(pila,operador)
12. Al final se sacan todos los elementos de la pila



# SUMA DE 2 NÚMEROS GRANDES USANDO PILAS

1. Leer el primer número y almacenar cada una de sus cifras en una pila (pila\_1)
2. Leer el segundo número y almacenar cada una de sus cifras en otra pila (pila\_2)
3. Crear una tercera pila (pila\_3) para guardar el resultado de la suma
4.  $\text{result} \leftarrow 0$
5. Mientras pila\_1 o pila\_2 no estén vacías
  6.  $\text{result} \leftarrow \text{result} + \text{pop}(\text{pila}_1) + \text{pop}(\text{pila}_2)$
  7.  $\text{push}(\text{pila}_3, \text{la parte de las unidades de result})$
  8. Quitarle la parte de las unidades a result
9.  $\text{push}(\text{pila}_3, \text{result})$
10. Desplegar todos los elementos de la pila\_3