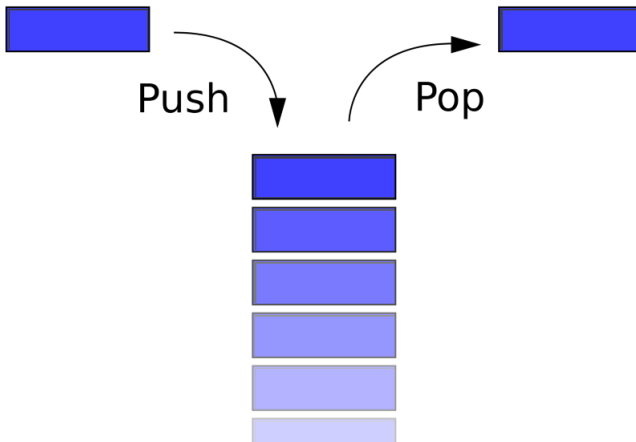


Estructura de datos en la que sólo se introducen y eliminan elementos por un extremo, llamado cima o tope.

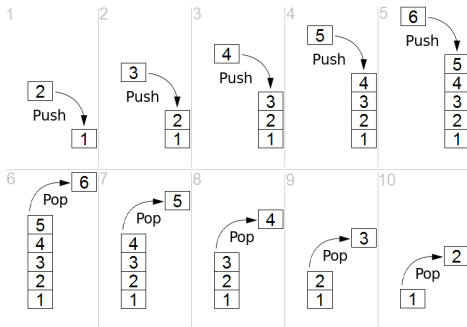


Estructura de datos en la que sólo se introducen y eliminan elementos por un extremo, llamado cima o tope.



Tiene dos operaciones básicas:

- **push()**: inserta un elemento en el tope de la pila.
- **pop()**: regresa el último elemento insertado en la pila.



# Implementación de operaciones básicas para pilas

**push()**

**pop()**

## Implementación de operaciones básicas para pilas: programa completo (1/2)

```
#include <stdio.h>
void main()
{
    int pila[5]={0,0,0,0,0}, tope=-1, opcion=0;
    while(opcion != 3)
    {
        printf("1) Push \n");
        printf("2) Pop \n");
        printf("3) Salir \n");
        printf("Elige la operacion a ejecutar: ");
        scanf("%d", &opcion);
        switch(opcion)
        {
            case 1: % push
                if(tope<4)
                {
                    tope ++;
                    printf("Introduce un numero a la pila: ");
                    scanf("%d",&pila[tope]);
                }
                else
                {
                    printf("La pila esta llena.\n");
                }
                break;
        }
    }
}
```

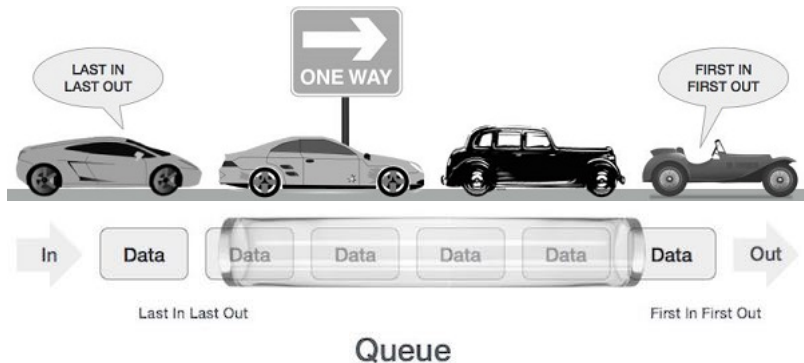
## Implementación de operaciones básicas para pilas: programa completo (2/2)

```
case 2: % pop
if(tope>=0)
{
printf("El elemento que sale de la pila es: %d.\n", pila[tope]);
pila[tope]=0;
tope --;
}
else
{
printf("La pila esta vacia.\n");
}
break;
}
}
```

Operaciones complementarias:

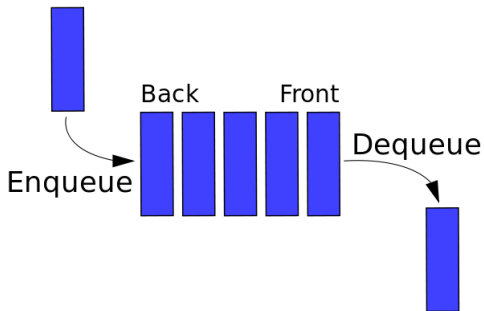
- **llena()**: regresa verdadero si la pila está llena.
- **vacía()**: regresa verdadero si la pila está vacía.
- **vaciar()**: elimina todos los elementos de la pila.
- **size()**: regresa la cantidad de elementos que contiene la pila.

Estructura de datos en la que la inserción de elementos se realiza por un extremo y la extracción por el otro.



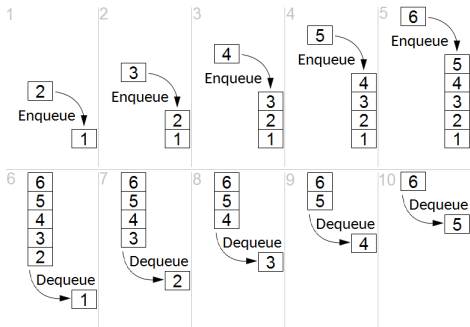


Estructura de datos en la que la inserción de elementos se realiza por un extremo y la extracción por el otro.



## Operaciones básicas:

- **insertar():** agrega un elemento al final de la cola.
- **remover():** remueve el primer elemento de la cola.



### Operaciones complementarias:

- **llena()**: regresa verdadero cuando la cola está llena.
- **vacía()**: regresa verdadero cuando la cola está vacía.

# Estructuras en c

```
struct <nombreDeEstructura>
{
    <tipoDeDato> <nombreDeVariable>;
    <tipoDeDato> <nombreDeVariable>;
    <tipoDeDato> <nombreDeVariable>;
    ...
    <tipoDeDato> <nombreDeVariable>;
};
```

## Ejemplo 1:

```
struct Pila
{
    int tope;
    int dato[5];
};

void main()
{
    struct Pila libros={0};
    ...
}
```

# Implementación de pilas con estructuras en c

## Ejemplo 2:

```
#include <stdio.h>
#define max 5

struct Pila
{
    int tope;
    int dato[max];
};

int main()
{
    struct Pila numeros={0};
    int num=0;
    printf("Que dato deseas ingresar?");
    scanf("%d", &num);
    numeros.dato[numeros.tope++]=num;
    printf("El dato introducido es %d \n", numeros.dato[numeros.tope-1]);
    printf("Sacar un dato de la pila\n");
    printf("El dato que se saca es %d \n", numeros.dato[--numeros.tope]);
    return 0;
}
```

## Ejercicio

- Implementar pilas y colas con estructuras estáticas.