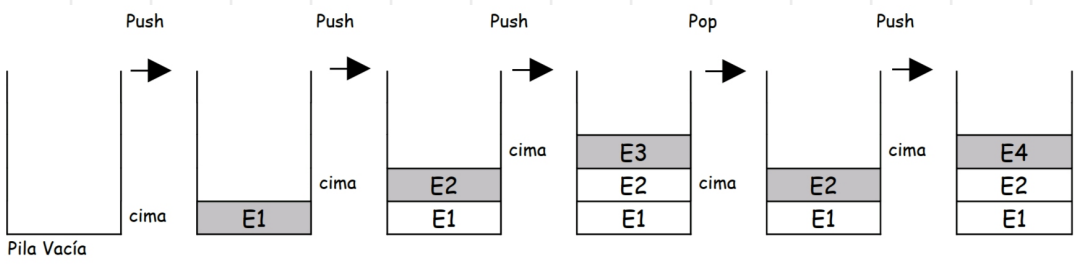
TEMA 2: TAD PILA

Definición : es una estructura ordenada y homogénea, en la que podemos apilar o desapilar elementos en una única posición llamada CIMA y siguiendo la política LIFO.

LIFO : (Last In First Out) el último elemento insertado es el 1° que puede salir de la pila.



↳ Cuando la pila es vacía, no tiene ningún elemento, entonces la CIMA tiene un valor indefinido

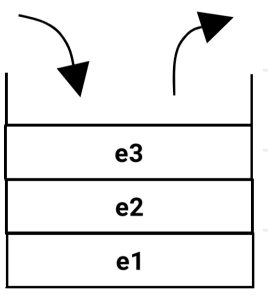
ESPECIFICACIÓN FORMAL

1. PilaVacia · Crea una pila vacía
2. Push · Inserta un elemento y devuelve la pila resultante.
3. Cima · Devuelve el elemento de la cima de la pila
4. Pop · Elimina el elemento de la cima y devuelve la pila resultante.
5. EsVacia : Determina si una pila tiene elementos o no.

Las operaciones constructoras de la estructura son PilaVacia y Push :

Push (Push (Push (PilaVacia, e1), e2), e3)

Gráficamente sería:



TAD TPILA ; Valores Colección de elementos homogéneos y que opera según el modelo LIFO.

SINTAXIS:

* PilaVacia .	→ TPILA
* Push (TPILA, ELEMENTO)	→ TPILA
Cima (TPILA) .	→ TELEMENTO
Pop (TPILA)	→ TPILA
EsVacia (TPILA) .	→ BOOLEAN

SEMÁNTICA

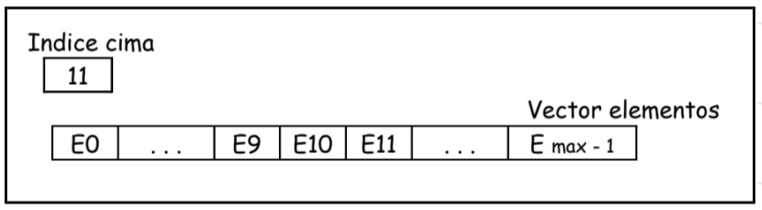
∀p ∈ TPILA, ∀e ∈ ELEMENTO

Cima (PilaVacia) .	→ ERROR
Cima (Push (p, e)) .	→ e
Pop (PilaVacia)	→ ERROR
Pop (Push (p, e)) .	→ p
EsVacia (PilaVacia)	→ TRUE
EsVacia (Push (p, e))	→ FALSE

REALIZACIÓN MEDIANTE MEMORIA ESTÁTICA EN PSEUDOCÓDIGO

- Uso de Arrays para implementar la estructura pila
- Además de un vector en el que se almacenen los elementos, es necesario contar con otra variable que indique cada momento cuál es el elemento cima de la pila
- Registro formado por dos campos, uno el array de elementos y otro campo para almacenar el índice de la posición que actúa de cima de la pila

- Gráficamente sería:



Estructura Tpila - Implementación mediante array

Declaraciones para hacer la implementación:

TIPO →

```
#define MAX ... /* número máximo de elementos que podrá almacenar la pila */
typedef .... TELEMENTO ; /* tipo de datos correspondiente a los elementos de la pila */
typedef struct {
    TELEMENTO arrayelementos[MAX];
    int cima;
} STPILA ;
typedef STPILA * TPILA;
```

→ Variables. `TPILA p;`

OPERACIONES:

- ① ***PilaVacia (/* E/S */ TPILA p)***

Precondición: Ninguna
Postcondición: Obtiene la pila p sin elementos. Por tanto, la pila p se modifica.

```
void PilaVacia(TPILA *p)
{
    *p = (TPILA) malloc(sizeof(STPILA));
    (*p)->cima = -1; /*Valor asignado por nosotros para indicar que la pila está vacía. */
}
```

- ***EsLlena (/* E */ TPILA p; /* S */ BOOLEAN respuesta)***

Precondición: Ninguna
Postcondición: Decide si la pila p no tiene capacidad para nuevos elementos. Por tanto, la pila p no se modifica.

```
int EsLlena (TPILA p)
{
    if (p->cima == MAX-1)
    return 1;
    else return 0;
}
```

- ② ***Push (/* E/S */ TPILA p; /* E */ TELEMENTO e)***

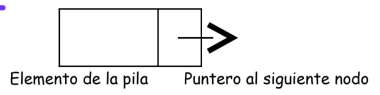
Precondición: La pila no debe estar llena
Postcondición: Almacena en la pila p el elemento e. Por tanto, la pila p se modifica.

```
void Push (TPILA *p, TELEMENTO e)
{
    int resp;
    resp = EsLlena( p: *p);
    if (resp == 1) printf("ERROR, la pila está llena\n");
    else
    {
        (*p)->cima++;
        (*p)->arrayelementos[(*p)->cima] = e ;
    }
}
```

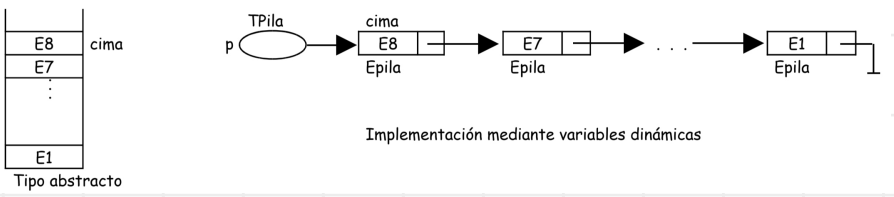
REALIZACIÓN MEDIANTE MEMORIA DINÁMICA en C.

- Hará uso de los Punteros para implementar la estructura pila
- El puntero solo accederá a la posición de memoria de la estructura dinámica que actúe de CIMA. Por tanto la Pila será un PUNTERO.
- La estructura dinámica se formará con elementos de tipo registro (los llamaremos NODOS) con los siguientes campos:
  - Valor del elemento de la pila.
  - Puntero con la dirección de memoria del siguiente nodo de la pila.

- Gráficamente cada nodo de la Pila será:



Así, la estructura Pila sería como:



Implementación mediante variables dinámicas

Declaraciones para hacer la implementación

TIPO →

```
typedef ... TELEMENTO ;
/* tipo de datos correspondiente a los elementos de la pila */
typedef struct nodo {
    TELEMENTO dato;
    struct nodo * sig;
} TNodo;
typedef TNodo * TPILA;
```

→ Variables. `TPILA p;`

OPERACIONES:

- ① ***PilaVacia (/* E/S */ TPILA p)***

Precondición: Ninguna
Postcondición: Obtiene la pila p sin elementos. Por tanto, la pila p se modifica.

```
void PilaVacia (TPILA *p)
{
    *p = NULL; /*Valor NULO en un puntero, para indicar que la pila está vacía. */
}
```

- ② ***Push (/* E/S */ TPILA p; /* E */ TELEMENTO e)***

Precondición: Ninguna
Postcondición: Almacena en la pila p el elemento e. Por tanto, la pila p se modifica.

```
void Push (TPILA * p , TELEMENTO e)
{
    TPILA q;
    q= (TPILA) malloc (sizeof(TNodo));
    q->dato = e ;
    q->sig = *p ;
    *p = q;
}
```

- ⑤ ***EsVacia (/* E */ TPILA p; /* S */ BOOLEAN respuesta)***

Precondición: Ninguna
Postcondición: Decide si la pila p tiene elementos o no. Por tanto, la pila p no se modifica.

```c
int EsVacia (TPILA p )
{
    if (p->cima == -1)
        return 1;
    else return 0;
}
```

- ③ ***Cima (/* E */ TPILA p; /* S */ TELEMENTO e)***

Precondición: La pila p no puede estar vacía
Postcondición: Obtiene el elemento que ocupa la cima de la pila p. Por tanto, la pila p no se modifica.

```c
void Cima (TPILA p, TELEMENTO * pe)
{
    int respuesta;
    respuesta = EsVacia(p);
    if (respuesta==1)
        printf("ERROR, la pila no tiene elementos\n");
    else
        *pe = p->arrayelementos [p->cima];
}
```

- ④ ***Pop (/* E/S */ TPILA p)***

Precondición: La pila p no puede estar vacía
Postcondición: Elimina de la pila p el elemento que ocupa la cima de la pila. Por tanto, la pila p se modifica.

```c
void Pop (TPILA *p)
{
    int respuesta;
    respuesta = EsVacia( p: *p);
    if (respuesta == 1)
        printf("ERROR, la pila no tiene elementos\n");
    else (*p)->cima = (*p)->cima - 1;
}
```

- ⑤ ***EsVacia (/* E */ TPILA p; /* S */ BOOLEAN respuesta)***

Precondición: Ninguna
Postcondición: Decide si la pila p tiene elementos o no. Por tanto, la pila p no se modifica.

```c
int EsVacia (TPILA p )
{
    if (p == NULL)
        return 1;
    else return 0;
}
```

- ③ ***Cima (/* E */ TPILA p; /* S */ TELEMENTO e)***

Precondición: La pila p no puede estar vacía
Postcondición: Obtiene el elemento que ocupa la cima de la pila p. Por tanto, la pila p no se modifica.

```c
void Cima (TPILA p, TELEMENTO * pe)
{
    int respuesta;
    respuesta = EsVacia(p);
    if (respuesta == 1)
        printf ("ERROR, la pila no tiene elementos\n");
    else *pe = p->dato;
}
```

- ④ ***Pop (/* E/S */ TPILA p)***

Precondición: La pila p no puede estar vacía
Postcondición: Elimina de la pila p el elemento que ocupa la cima de la pila. Por tanto, la pila p se modifica.

```c
void Pop (TPILA * p)
{
    TPILA q;
    int respuesta;
    respuesta = EsVacia( p: *p);
    if (respuesta==1)
        printf("ERROR, ....\n");
    else
    {
        q = *p;
        *p = (*p)->sig;
        free(q);
    }
}
```