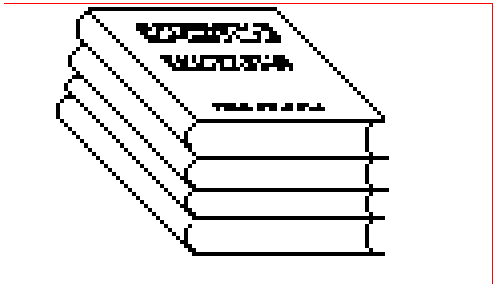
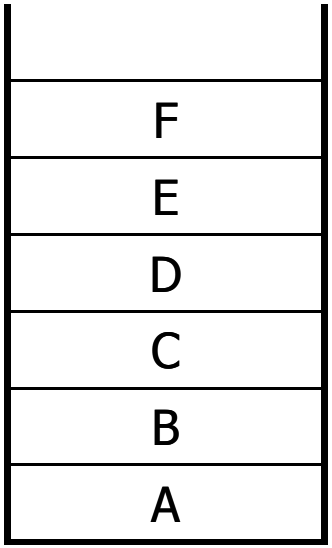
PILAS

Una pila es una colección ordenada de elementos en la que pueden insertarse y suprimirse elementos por un extremo llamado TOPE.





PILAS

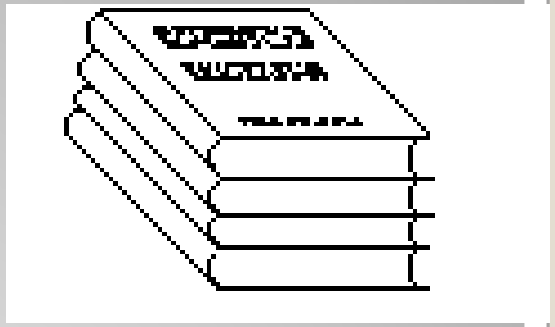
Las pilas son estructuras que se encuentran frecuentemente en la vida diaria. Algunos ejemplos se encuentran en la forma en que se acomodan los platos en algunas cafeterías, la manera en que se colocan libros en un escritorio, una lata de pelotas de tenis o bien algunas tareas que se desean realizar.

La pila es uno de los conceptos más útiles dentro de la ciencia de la computación.

PILAS

Por ejemplo, la forma en que se acomodan los libros en un escritorio puede verse de la siguiente manera:

se puede añadir un libro a una pila poniéndolo hasta arriba de todos, se puede ver qué libro se encuentra hasta arriba y de esta misma forma si se desea retirar uno, se tomará el último que se colocó.



La propiedad anterior se conoce como LIFO (*Last In First Out*), es decir, el último en entrar será el primero en salir y al que se tenga acceso. Otros nombres con los que se conoce a las pilas son 'lista empuja hacia abajo' *(pushdown list*) y UEPS (últimas entradas primeras salidas).

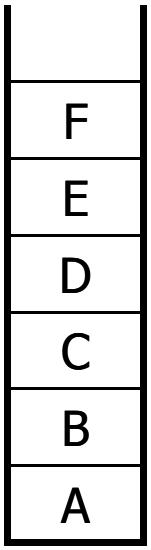
PILAS

Las únicas operaciones que pueden realizarse con pilas son las siguientes:

CreaPila()	Crea una pila vacía.
PilaVacia()	Responde si la pila se encuentra vacía.
Poner(x)	Inserta el elemento x en la pila.
Sacar()	Saca un elemento de la pila.
ValorTope()	Devuelve el valor del tope de la pila, sin modificar la pila.

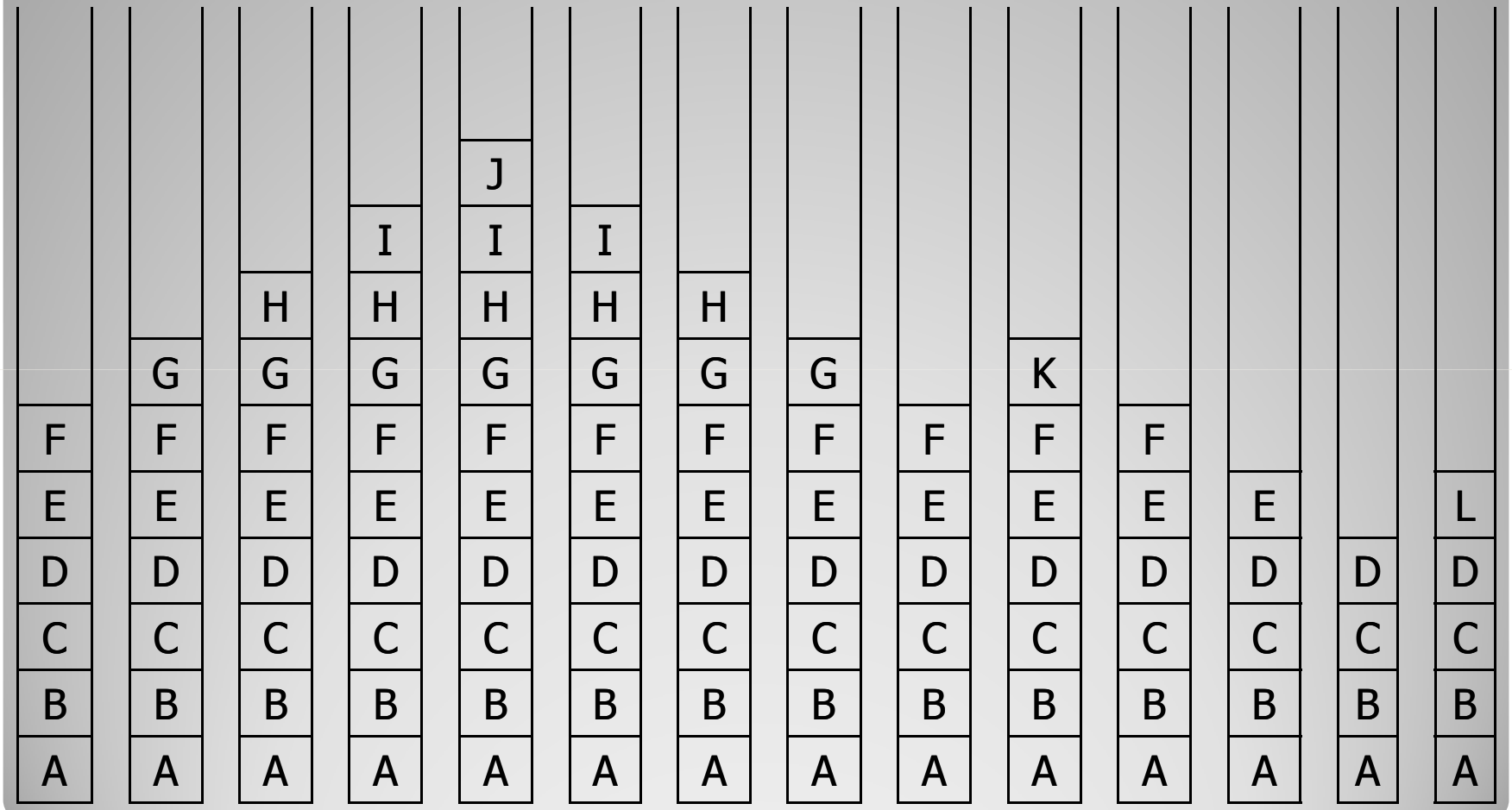
PILAS

Veamos lo que sucede con nuestra pila si seguimos la siguiente secuencia de acciones:



Poner(G)
Poner(H)
Poner(I)
Poner(J)
Sacar()
Sacar()
Sacar()
Sacar()
Poner(K)
Sacar()
Sacar()
Sacar()
Poner(L)

PILAS


F E D C B A
G F E D C B A
H G F E D C B A
I H G F E D C B A
J I H G F E D C B A
I H G F E D C B A
H G F E D C B A
G F E D C B A
F E D C B A
K F E D C B A
F E D C B A
E D C B A
D C B A
L D C B A

PILAS: Un ejemplo

Consideremos una operación matemática que contenga varios paréntesis anidados:

7-((x * ((x + y) / (j – 3)) + y) / (4 – 2.5))

Queremos comprobar que los paréntesis estén anidados en forma correcta; es decir:

1. Que hay el mismo número de paréntesis izquierdos y derechos.
2. Que todo paréntesis derecho este precedido por su pareja izquierda.

PILAS: Un ejemplo

Nuestro programa debería entonces controlar dos cosas:

1. El conteo de paréntesis es 0 al final de la expresión. (o sea que ningún paréntesis ha quedado abierto)
2. El conteo de paréntesis no es negativo en ningún lugar de la expresión. (o sea que no hay ningún paréntesis derecho sin un paréntesis izquierdo)

Problemas como:

((a + b) c o a + b (o) a + b (-

violan las condiciones indicadas.

PILAS: compliquemos el ejemplo

Consideremos una operación matemática que contenga varios tipos de paréntesis anidados:

7- { [x * ((x + y) / [j – 3]) + y] / (4 – 2.5)}

En este caso no alcanza con una pila que cuente, sino que necesitamos ver que el paréntesis de cierre sea del mismo tipo que abrimos.

Intentemos resolverlo.

CASO: 7- { [x * ((x + y) / [j – 3]) + y] / (4 – 2.5)}

```
Funcion = valida
CreaPila()
Repetir mientras cadena valida
        Si symb = {  o symb = [ o symb = (
            Poner( symb )
        Si symb = }  o symb = ] o symb = )
            Si PilaVacia()
                        funcion = invalida
            Sino
                        tope = ValorTope()
                        Sacar()
                        Si !( tope = Opuesto( symb ) )
                                    funcion = invalida
Fin Repetir
Si !PilaVacia()
        funcion = invalida

Imprimir( Funcion )
```

PILAS: como tipo de dato abstracto

```
abstract typedef <eltype> STACK( eltype);

abstract empty( s )
STACK (eltype) s;
postcondition    empty == ( len( s ) == 0 )

abstract eltype pop( s )
STACK( eltype ) s;
precondition     empty( s ) == FALSE
postcondition    pop == first( s´);
                 s == sub( s´, 1, len( s´) – 1 );

abstract push( s, elt )
STACK( eltype ) s;
eltype elt;
postcondition    s == <elt> + s´;
```

PILAS : Ejercicio

- Paso de notación INFIJA a PREFIJA y POSFIJA

INFIJA	A+B	A+B*C	(A+B)*C	(A+B) * (C-D)
PREFIJA	+AB	+A*BC	*+ABC	*+AB-CD
POSFIJA	AB+	ABC*+	AB+C*	AB+CD-*

CASO: *+AB-CD (notacion prefija)

```
CreaPila()
Repetir mientras len( cadena ) > 0
        symb = primerelemento( cadena )
        cadena = sub( cadena, 2, len( cadena ) – 1 )
        Si symb =+  o symb =- o symb =* o symb =/
            Poner( symb )
        Si !(symb =+  o symb =- o symb =* o symb =/)
            tope = ValorTope()
            Si tope =+  o tope =- o tope =* o tope =/
                Poner( symb )
            Sino
                valor1 = Sacar()
                operación = Sacar()
                resultado = Evaluar( valor1, operación,
        symb)
                cadena = resultado + cadena
Fin Repetir

Imprimir( resultado )
```