TAREA #7 PILAS

CINTHIA GUADALUPE OLIVAS CALDERON NO.17212165

Una pila es un tipo especial de lista en la que todas las inserciones y supresiones tienen lugar en un extremo denominado tope. A las pilas se les llama también «listas LIFO» (last in first out) o listas «último en entrar, primero en salir».

Se realiza sólo por un extremo que se denomina cima o tope (top). La pila es una estructura con numerosas analogías en la vida real: una pila de platos, una pila de monedas, una pila de cajas de zapatos, una pila de camisas, una pila de bandejas, etc.

Dado que las operaciones de insertar y eliminar se realizan por un solo extremo (el superior), los elementos sólo pueden eliminarse en orden inverso al que se insertan en la pila. El último elemento que se pone en la pila es el primero que se puede sacar; por ello, a estas estructuras se les conoce por el nombre de LIFO (last-in, first-out, último en entrar, primero en salir).

Las pilas se pueden representar en cualquiera de las tres formas:

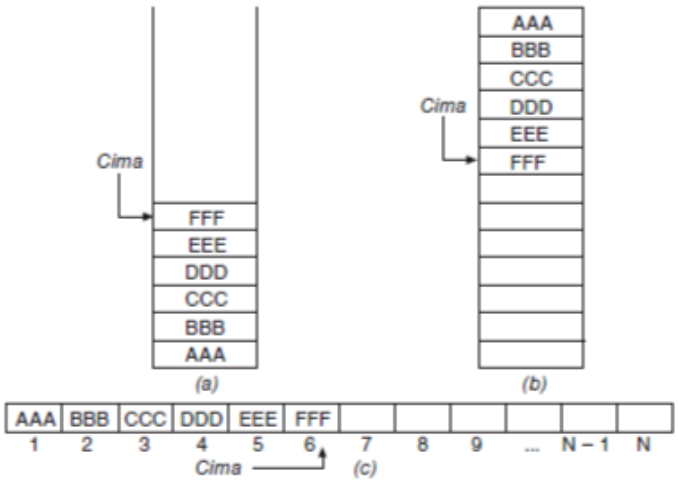


Figura 12.16. Representación de las pilas.

Las pilas son muy utilizadas en programación, para evaluar expresiones, reconocer lenguajes, recorrer árboles y simular procesos recursivos. En todo momento, el único elemento visible de la estructura es el último que se colocó. Se define el tope de la pila

como el punto donde se encuentra dicho elemento, y el fondo, como el punto donde se encuentra el primer elemento incluido en la estructura.

Si la pila no tiene ningún elemento se dice que se encuentra vacía y no tiene sentido referirse a su tope ni a su fondo. Una pila vacía se representa con el símbolo Ø. Por último, se define la longitud de una pila como el número de elementos que la conforman.

Para trabajar fácilmente con pilas es conveniente diseñar subprogramas de poner (push) y quitar (pop) elementos. También es necesario con frecuencia comprobar si la pila está vacía; esto puede conseguirse con una variable o función booleana VACIA, de modo que cuando su valor sea verdadero la pila está vacía y falso en caso contrario.

P = CIMA	Puntero de la pila.
VACIA	Función booleana «pila vacía».
PUSH	Subprograma para añadir, poner o insertar elementos.
POP	Subprograma para eliminar o quitar elementos.
LONGMAX	Longitud máxima de la pila.
X	Elemento a añadir/quitar de la pila.

Aplicaciones de las pilas

Las pilas son utilizadas ampliamente para solucionar una amplia variedad de problemas. Se utilizan en compiladores, sistemas operativos y en programas de aplicación.

- Llamadas a subprogramas

Cuando dentro de un programa se realizan llamadas a subprogramas, el programa principal debe recordar el lugar donde se hizo la llamada, de modo que pueda retornar allí cuando el subprograma se haya terminado de ejecutar

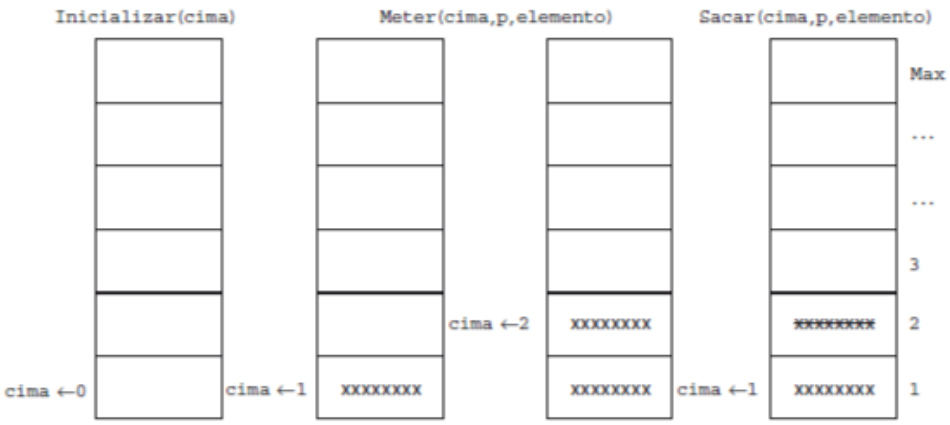
Implementación con arrays

Necesitaremos un array y una variable numérica cima que apunte al último elemento colocado en la pila.

La inserción o extracción de un elemento se realizará siempre por la parte superior.

Su implementación mediante arrays limita el máximo número de elementos que la pila puede contener y origina la necesidad de una función más.

Llena(...) de resultado lógico




```
const Max = <expresión>
tipo
     registro: tipo_elemento
        ... : ...
        ... : ...
   fin_registro
   array[1..Max] de tipo_elemento: arr
var
    entero :      cima
    arr :           p
    tipo_elemento : elemento
inicio
inicializar(cima)
...
fin
procedimiento inicializar(S entero: cima)
inicio
cima ← 0
```

```
fin_procedimiento
logico función vacia(E entero: cima)
inicio
si cima = 0 entonces
devolver (verdad)
si_no
devolver (falso)
fin_si
fin_función
lógico función llena(E entero: cima)
inicio
si cima = Max entonces
devolver (verdad)
si_no
devolver (falso)
fin_si
fin_función
procedimiento consultarCima( E entero: cima; E arr:p;
S tipo_elemento: elemento)
inicio
si no vacia (cima) entonces
elemento ← p[cima]
fin_si
fin_procedimiento
procedimiento meter( E/S entero: cima; E/S arr: p;
E tipo_elemento: elemento)
inicio
si no llena (cima) entonces
```

```
cima ← cima + 1
p[cima] ← elemento
fin_si
fin_procedimiento
procedimiento sacar( E/S entero: cima; E arr: p;
S tipo_elemento: elemento)
inicio
si no vacia (cima) entonces
elemento ← p[cima]
cima ← cima - 1
fin_si
fin_procedimiento
```

REFERENCIAS:

Joyanes Aguilar, L. (2003). Fundamentos de programación: algoritmos y estructura de datos y objetos.

José Fager W. Libardo Pantoja Yépez Marisol Villacrés Luz Andrea Páez

Martínez Daniel Ochoa Ernesto Cuadros-Vargas. (2014). Tipos de Datos

Abstractos.En Estructuras de datos(222). Latinoamérica: LATIn.