UNIVERSIDAD NACIONAL AUTÓMATA DE MÉXICO

FACULTAD DE INGENIERÍA

ESTRUCTURA DE DATOS Y ALGORITMOS I

ACTIVIDAD ASÍNCRONA #4 MIÉRCOLES

PÉREZ AGUILAR ROBERTO

(10/06/2021)

Pilas

Definición

Es una estructura de datos que permite el almacenamiento de datos en el orden LIFO (Lats in First Out). La recuperación de los datos es realizada en el orden inverso de su inserción.

Ya que la inserción es siempre hecha al inicio de la lista, el primer elemento de la lista será el último elemento ingresado, por lo tanto estará en la cabeza de la pila. El último elemento ingresado, será el primer elemento recuperado.

Para definir un elemento de la pila será utilizada struct. El elemento de la pila contendrá un campo dato y un puntero.
El puntero tiene que ser de la misma clase que el elemento, sino, no va a poder apuntar hacia el elemento.

```
typedef struct ListaUbicación{
Elemento *inicio;
int tamaño;
} Pila;
```

El puntero inicio indicará la dirección del primer elemento de la lista.
La variable tamaño contiene el número de elemento

Tipos de pilas

Pila vacía: no contiene elemento alguno dentro de la estructura y el tope de la misma apunta al nulo. En esta no es posible realizar POP, debido a que sus estructura no contiene información. Cuando la pila está vacía se puede realizar PUSH, el nodo que entraría a la estructura sería el único elemento de la pila y el tope apuntaría a él.

Pila llena: una estructura de datos tipo pila va tener un tamaño fijo. Cuando la pila ha almacenado el número máximo de nodos, quiere decir que la pila está llena. En una pila llena no es posible hacer PUSH de un nuevo elemento ya que ha alanzado el tamaño permitido. Y cuando está llena se puede hacer POP de la información contenida en la estructura. En tal caso, el tope al elemento de la siguiente estructura

Pila con elementos: una pila que contiene elementos sin que esta llegue a su límite. En una pila con elementos se puede realizar PUSH en tal caso, el tope apuntara al elemento que se insertó y el nuevo elemento apunta al elemento al que apuntaba tope. También es posible realizar POP, el nodo al que apunta tope se extrae y ahora tope apunta al elemento al que apuntaba este.

Operaciones sobre las pilas

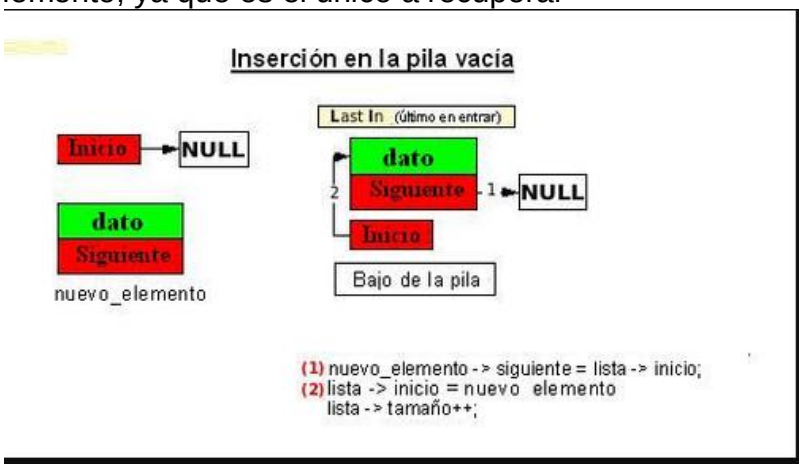
Inicialización:

void inicialización (Pila *tas);
Esta operación debe ser realizada antes de cualquier otra operación sobre la pila. Esta inicializa el puntero **inicio** con el puntero **NULL** y el tamaño con el valor 0. La función

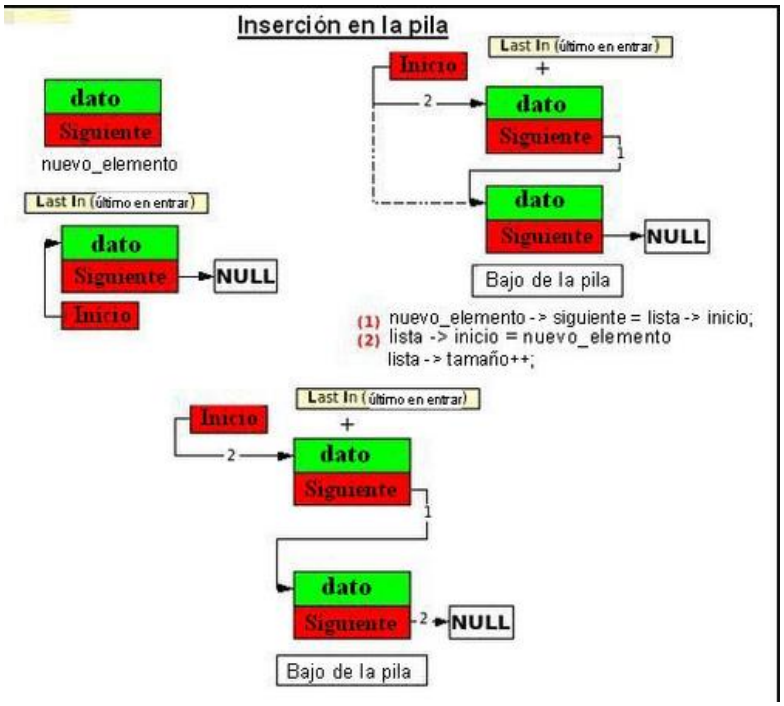
```
void inicialización (Pila * tas){
tas->inicio = NULL;
tas->tamaño = 0;
}
```

Inserción de un elemento en la pila:

int apilar (Pila *tas, char *dato);
La primera imagen muestra el comienzo de la inserción, por lo tanto la lista de tamaño **1** después de la inserción. La característica de la pila no es muy apreciada con un solo elemento, ya que es el único a recuperar



Mientras la siguiente imagen permite observar el comportamiento de la pila. Lo que debemos retener es que la inserción siempre se hace en la parte superior de la pila (al inicio de la lista).



La función

```
/* apilar (añadir) un elemento en la pila */
int apilar (Pila * tas, char *dato){
Elemento *nuevo_elemento;
if ((nuevo_elemento = (Elemento *) malloc (sizeof (Elemento))) == NULL)
return -1;
if ((nuevo_elemento->dato = (char *) malloc (50 * sizeof (char)))
== NULL)
return -1;
strcpy (nuevo_elemento->dato, dato);
nuevo_elemento->siguiente = tas->inicio;
tas->inicio = nuevo_elemento;
tas->tamaño++;
}
```

Eliminar un elemento de una pila:

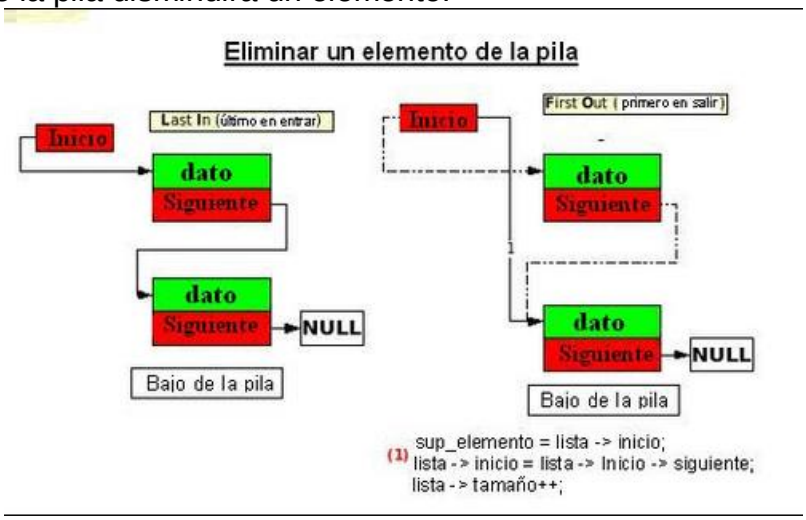
Para poder eliminar un elemento de una pila hay que quitar el elemento hacia el cual apunta el puntero inicio. Esta operación no permite recuperar el dato de cabeza de pila, solo eliminando.

Modelo de la función:

```
int desapilar (Pila *tas);
```

La función da como resultado -1 en caso de error, si no devuelve 0.

Las etapas:
El puntero **sup_elemento** contendrá la dirección del primer elemento.
El puntero **inicio** apuntará hacia el segundo elemento (después de la eliminación del primer elemento, el segundo elemento estará en la cabeza de la pila).
El **tamaño** de la pila disminuirá un elemento.



La función

```
int desapilar (Pila * tas){
Elemento *sup_elemento;
if (tas->tamaño == 0)
return -1;
sup_elemento = tas->inicio;
tas->inicio = tas->inicio->siguiente;
free (sup_elemento->dato);
free (sup_elemento);
tas->tamaño--;
return 0;
}
```

Visualización de una pila

Para poder mostrar una pila entera, es necesario posicionarse al inicio de la pila (el puntero inicio luego, utilizando el puntero siguiente de cada elemento, la pila es recorrida del primero a la condición para detenerse por el tamaño de la pila

La función

```
/* visualización de la pila */
void muestra (Pila * tas){
Elemento *actual;
int i;
actual = tas->inicio;
```

```
for(i=0;i<tas->tamaño;++i){
printf("\t\t%s\n", actual->dato);
actual = actual->siguiente;
}
}
```

Bibliografía:

Guerreo. (21 de marzo 2018). Las pilas en lenguaje C. 10 de junio 2021, de Programa en línea Sitio web: https://www.programaenlinea.net/las-pilas-lenguaje-c/