

TAD Pila

Algoritmos y estructuras de datos

Ignacio Cassol



Descripción formal

- Una pila es una estructura de datos, caracterizada por ser una secuencia de elementos en la que las operaciones de inserción y extracción se realizan por el mismo extremo.
- Las pilas también se llaman estructuras LIFO (del inglés Last In First Out), debido a que el último elemento en entrar será el primero en salir.
- Analogías



Descripción formal II

- Se trata de un grupo de datos ordenados de acuerdo al tiempo en que llevan en la estructura de datos.
- El canal por donde entran y salen los datos se llama *tope*.

Operaciones

- Constructoras: `Stack()`
- Modificadoras: `push(x)` , `pop()`
- Analizadoras: `peek()` , `isEmpty()` , `size()`
- Destructoras:
- Realizar la especificación de la Pila.

Interface de Java

```
public interface Stack<T> {  
    void push(T t);  
    void pop();  
    T peek();  
    boolean isEmpty();  
    int size();  
    void empty();  
}
```

En Java

1 de 3

```
public class StaticStack<T> implements Stack<T>
{

    private int top;
    private int capacity;
    private Object[] data;

    public StaticStack(int x){
        top = -1;
        capacity = x;
        data = new Object [capacity];
    }
    public void push(T o) {
        if(top+1==data.length){
            grow();
        }
        top++;
        data[top]= o;
    }
    public void pop() {
        top--;
    }
}
```

En Java

2 de 3

```
public boolean isEmpty() {
    if (top == -1){
        return true;
    }
    return false;
}

public void empty() {
    top = -1;
}

@SupressWarnings("unchecked")
public T peek() {
    if (!isEmpty()){
        return (T) data[top];
    }
    return null;
}
```

En Java

3 de 3

```
private void grow(){
    Object[] data2 = new Object[2*capacity];
    for (int i =0; i<capacity;i++){
        data2[i] = data[i];
    }
    data = data2;
}
```


En C

```
#include<stdio.h>
#include<stdlib.h>

#define TAM 6
#define MAX TAM-1

typedef struct
{
    int tope;
    int item[TAM];
} pila;

int full(pila *);
int empty(pila *);
void push(pila *, int);
void pop(pila *,int *);

int main() { ...
```

En C 2 de 4

```
int main()
{
    pila p,t;
    int dato,opc,elemento,flag=0;
    p.tope=0;
    do
    {
        system("cls");//Borrar pantalla
        printf("\nMENU-PILA");
        printf("\n1-> Insertar elemento");
        printf("\n2-> Eliminar elemento");
        printf("\n3-> Visualizar");
        printf("\n4-> Salir");
        printf("\n\nIndique su opcion : ");
        scanf("%d",&opc);

        switch(opc)
        { ...
```

En C 3 de 4

```
switch(opc)
{
case 1:
    if(!full(&p)) // si pila no esta llena
    {
        printf("\nDe el elemento a insertar: ");
        scanf("%d",&dato);
        push(&p,dato);
        printf("\nElemento insertado...");
    }
    else
    {
        printf("\nERROR: Pila llena");
    }
    break;

case 2:
    if(!empty(&p))
    {
        pop(&p,&dato);
        printf("\nEl elemento eliminado es %d",dato);
    }
    else
    {
        printf("\nERROR: Pila vacia");
    }
    break;

case 3:
    if(!empty(&p))
    {
        t.tope=0;
        pop(&p,&dato);
        printf("\n%d",dato);
        push(&t,dato);
    }

case 4:
    flag=1;
    break;

case 5:
    flag=0;
    break;

default:
    printf("\nOpcion no valida...");
}

if(!flag)
{
    printf("\n\nPres. una tecla...");
    getch();
}
}while(!flag);

return 0;
}
```

En C 4 de 4

```
int full(pila *p)
{
    return (p->tope==MAX) ;
}
```

```
int empty(pila *p)
{
    return (p->tope==0) ;
}
```

```
void push(pila *p,int dato)
{
    if(!full(p))
    {
        (p->tope)++;
        p->item[p->tope]=dato;
    }
    else
        printf("\nOVERFLOW" );
}

void pop(pila *p,int *dato)
{
    if(!empty(p))
    {
        *dato=p->item[p->tope];
        (p->tope)--;
    }
    else
        printf("\nUNDERFLOW" );
}
```