Elsy Joselyn Godinez Juarez | GTIDI41

R1 Unidad 2 "Ejercicios sobre Pilas (stacks)"

Ejercicio 1. Simulacion de Operaciones en una Pila.

1: Push (10)
Pila: [10]
2: Push (20)
Pila: [20, 10]
3: Push (5)
Pila: [5, 20, 10]
4: Pop() → se elimina el 5
Pila: [20, 10]
5: Push (8)
Pila [8, 20, 10]
6: Pop () → Se elimina el 8
Pila: [20, 10]
7: Push (12)
Pila: [12, 20, 10]

Preguntas

¿Cual es el contenido final de la pila (de arriba hacia abajo)?
R= 12, 20, 10
¿Cuantos elementos contiene?
R= 3 elementos

Escaneado con CamScanner

01	Dic	25	Elsy Joselyn Cadinez Juarez

Ejercicio 2 Verificar si una palabra es Palindroma usando una Pila".

1) Inserción (Push) de cada letra "estado de pila despues de cada Push

Palabra : RADAR

- Push('R')

Top → |R|

- Push('A')

Top → |A|
|R|

- Push('D')

Top → |A|
|R|

- Push('A')

Top → |A|
|D|
|A|
|R|

- Push('R')

Top → |R|
|A|
|D|
|A|
|R|

2) Extraccion (Pop) y Comparacion por letra

Paso	indice (0-based)	letra en la palabra	Pop
1	0	R	R
2	1	A	A
3	2	D	D
4	3	A	A
5	4	R	R

ádeko

Escaneado con CamScanner

Elsy Joselyn Godinez Juarez

GIIDIY1
01 | DIC | 25

"Ejercicio 3 Convertir una expresion de infijo a Postfijo"

Estado Inicial

1: Pila: [ ]
Salida: ""

2. Token A (operado)
Pila: [ ]
Salida: A

3 Token + (operador) - aplicar +
Pila: [+]
Salida: A

4 Token B (operado)
Pila: [+]
Salida: A B

5 Token * (operador) - * tiene mayor precedencia que + asi que se apila sin despilar +
Pila: [*, +] (tope a la izquierda: *)
Salida: A B
"Algunos muestran la pila con el tope a la derecha; aqui indico el tope ala izquierda para mayor claridad"

6: Token C (operado)
Pila: [*, +]
Salida: A B C

7 Token - (operador) - al procesar - desapilamos operadores del tope con mayor o igual precedencia que -
- Tope es * (precedencia mayor) → desapilar * y llevar a salida
- Nuevo tope es + (misma precedencia que -) → tambien desapilar + y llevar a salida
- Ahora la pila esta vacia; apilar -

Despues de desapilar / aplicar:
Pila: [-]
Salida: A B C * +

adeko

Escaneado con CamScanner

8 Token D (operado)
Pila: [-]
Salida: A B C * + D
9. Fin de la expresion - desapilar todo lo que quede en la pila (tope -)
a la salida
Pila: [ ]
Salida Final: A B C * + D -
Resultado Final
A B C * + D -

Ejercicio 4. Pila aplicada: Deshacer (UNDO)

Simula un sistema de editor de texto con una pila de acciones

- Acciones realizadas

1: Escribir "A"
2: Escribir "B"
3: Escribir "C"
4: UNDO
5: Escribir "D"
6: UNDO
7: UNDO

Estado de la pila y del texto
1: Escribir "A"
Pila: [A]
Texto: "A"
2: Escribir "B"
Pila: [A, B]
Texto: "AB"
3: Escribir "C"
Pila: [A, B, C]
Texto: "ABC"
4: UNDO → elimina "C"
Pila: [A, B]
Texto: "AB"

Escaneado con CamScanner

Elsy Josely Godinez Juarez

GTID141		
01	Dic	25

Pila: [A, B, D]
Texto: "ABD"
G: UNDO → elimina "D"
Pila: [A, B]
Texto: "AB"
J: UNDO → elimina "B"
Pila: [A]
Texto B
Preguntas
- ¿Cual es el contenido final del texto?
  R = A
- ¿Que queda en la pila?
  R = [A]

Ejercicio 5 Completa la Semi-Implementacion de una Pila en Java"

```
Public Class Pila {

  Private int[] datos;
  Private int tope;

Public Pila(int capacidad ){
  datos = new int[capasidad];
  tope = -1;
}
  // Insertar un elemento en la Pila
Public void push(int valor ){
  //TODO: Verificar si esta llena (overflow)
```

Escaneado con CamScanner

GITHUB

01	Dic	25	Elsy Joselyn Godinez Juarez

Ejercicio 5 Completa la Semi-Implementacion de una Pila en Java":

Completa la clase Pila implementada con arreglo estatico, sin usar Stack ArrayList ni colecciones de Java

Clase Pila

```
Public class Pila {
 Private int [] datos;
 Private int tope;

Public Pila (int Capacidad){
   datos = new int[capacidad];
   tope = -1;
}
   // Insertar un elemento sin pila
Public void Push(int valor){
    // TODO: Verificar si esta llena (overflow)
    if (estaLlena()){
  System.out.println("Error : Pila llena (overflow)");
   return;
}
   // TODO: Incrementar tope y asignar valor
    tope++;
   datos [tope] = valor;
}
   // Elimina el elemento del tope
 Public int Pop(){
   // TODO: verificar si esta vacia (underflow)
   if (estaVacia()){
System.out.println("Error : Pila vacia (underflow)");
 return -1;
 }
```

Escaneado con CamScanner

```
// TODO: Retornar elemento y disminuir tope
    int valor = datos[tope];
    tope--;
    return valor;
}
// Consultar el valor del tope sin eliminarlo
    Public int peek(){
    // TODO: Retornar el elemento del tope si existe
    if(estaVacia()){
    System.out.println("Error: Pila vacia (sin elementos)");
    return -1;
}
return datos[tope];
}
Public boolean estaVacia(){
    // TODO: Verificar si tope == -1
    return tope == -1;
}
Public boolean estaLlena(){
    // TODO: verificar si tope == datos.length - 1
    return tope == datos.length - 1;
}
Public void mostrarPila(){
    // TODO: Mostrar elementos de arriba hacia abajo
    if(estaVacia()){
    System.out.println("Pila vacia");
    return;
}
System.out.println("Contenido de la pila (de arriba a abajo):");
    for(int i = tope; i >= 0; i--){
    System.out.println(datos[i]);
    }
}
```

adeko

Escaneado con CamScanner

01	Dic	25	Elsy Joselyn Godinez Juarez

Ejercicio 6. Aplicacion Conceptual

Explica en papel que estructura seria mas adecuada para cada caso y por que

1: Navegacion "atras" es un navegador

Estrutura adecuada: Pila (stack)

Justificacion:

Es un navegador, cuando visitas paginas, cada nueva pagina se apila sobre la anterior. Cuando presionas "Atras", el navegador necesita registrar a la ultima pagina visitada es decir, a la mas reciente

Lifo: Last In First Out

(Es ultimo en entrar es el primero en salir

2: Evaluacion de expresiones matematicas

Estrutura adecuada: Pilas (dos pilas)

Justificacion

Para evaluar expresiones como:

3 + 5 * (2 - 1)

los ultimos clasicos (como el de Dijkstra onotacion postfixa) utilizan pilas:

- Una pila para operados
- Una pila Para operadores

Las pilas (pertenecen) permiten procesar la expresion en el orden correcto usando LIFO recuperando lo mas reciente cuando es necesario

3: Verificar de Parentesis balanceados

Estrutura adecuada: Pila

Justificacion:

([[]])

Esta bien balanceada, se utiliza una pila por que:

- Cada vez que aparecen un parentesis que abre, se inserta en la pila

Escaneado con CamScanner

Elsy Joselyn Godinez Juarez | 01 | Dic | 25

- Cuando aparece uno que cierra, debe coincidir con el que esta en el tope de la pila

Si al final la pila esta vacia, la expresion esta balanceada

→ Funcion por que lo ultimo que abrio debe ser lo primero en cerrarse (LIFO)

4: Implementacion de un Sistema de deshacer (UNDO)

Estructura de datos

Justificacion:

En aplicaciones como word, Photoshop o editores de texto, el sistema de UNDO guarda cada accion del usuario en una pila

Cuando presionas Ctrl+Z se debe revertir la ultima accion realizada, no la primera

→ Es exatamente el comportamiento de una pila

Deshacer = Pop() del ultimo cambio

Escaneado con CamScanner