# UNIT:- 2

## STACKS:-

### Introduction:- And Defination:-

A Stack is defined as linear data structure in which the insertion and deletion operations are performed on the same end of the stack.

→ The position (or) place of the stack where the insertion and deletion operations are performed are called "TOP" of the stack.

→ Stack follows LIFO mechanism or structure.

→ LIFO can be abrivated as Last-In-First-Out

→ Stack is also called as Last-In-First-Out Structure. And some of the Examples that follows LIFO Structure are :- Ex:- Idly Stand; CD's Stand, Plates on a tray.

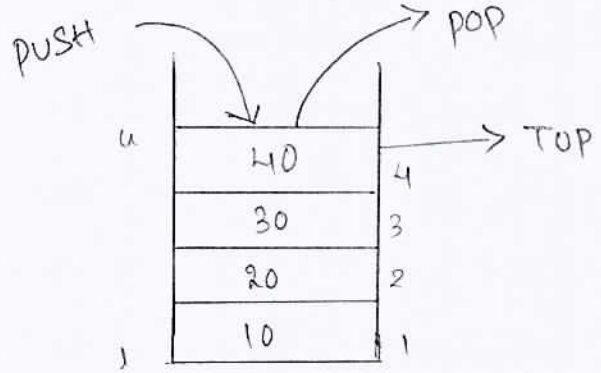



fig:- Stack Structure

### Terminology of Stack:-

In Stack we discuss the following terms.

i) TOP
ii) Stack is empty (or) Stack underflow
iii) Stack is full (or) Stack Overflow
iv) Push Operation
v) Pop Operation

**Top:**

The position (or) location of stack where the insertion and deletion must be performed at same end of stack is called top of the stack

**Stack is Empty (or) Stack is underflow:**

When TOP < 1 then the stack is said to be empty (or) stack is underflow. At this condition the deletion operation is impossible.

**Stack is full (or) Stack is overflow:**

When TOP ≥ 4 then the stack is said to be full (or) stack is overflow. At this condition the insertion operation is impossible.

**PUSH operation (or) Insertion operation:**

1. The process of inserting an element into the stack is called push operation.
2. Always the new element is inserted at only one end in the stack ie Top of the stack.
3. While performing insertion (or) push operation. first we should check the stack is full (or) overflow.
4. Before inserting an element into the stack first the 'TOP' value is increased by '1'

i.e. $\boxed{TOP = TOP + 1}$

5. Then insert an element into the stack.

## Pop Operation (or) Deletion Operation:

1. The process of deleting an element from the stack is called as pop operation.

2. Always the element is deleted from only one end of the stack. (i.e) Top of the stack.

3. While performing Deletion (or) pop operation. first we should check. The stack is Empty (or) underflow.

4. Before deleting an element from the stack first reduce the Top value by '1'.

i.e. `TOP = TOP - 1`

5. Then delete the element from the stack.

# STACK REPRESENTATION:

## Representation of a STACK:-

A stack can be represented in the memory in two ways.

(1) Array Representation of stacks.

(2) Linked. List Representation of stacks.

### 1). Array Representation of Stacks:-

→1. Here the stack is represented by using one-dimensional array.

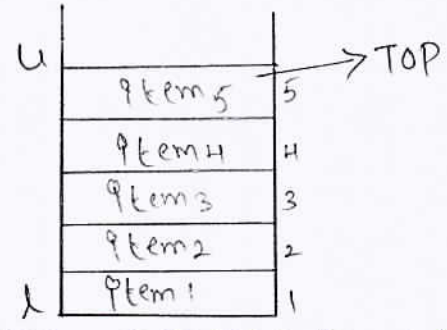
→2. We have to allocated a memory block of sufficient size to accommodate the full stack.

→3. Then, starting from the first location of the memory block, the items of the stack can be stored in a sequential fashion.

→4. Item; denotes the $i^{th}$ item in the stack. An element in the stack is termed as "ITEM".

→5. 'l' (lower bound) and 'u' (upper bound) denotes the index range of the array. And the values of these indices are 'l' and 'SIZE' respectively.

→6. 'Top' is a pointer to point the position of the array upto which it is filled with the items of the stack.




EMPTY = TOP < 1

FULL = TOP ≥ u

SIZE = u+1 − 1

fig: Array representation of a stack.

## 2]. Linked - List Representation of Stack:.

→ The size of the stack may vary during program Execution. A solution to this problem is to represent a stack using a linked list.

→ In linked list the DATA field is for the ITEM and the LINK field is, used to point to the next item.

→ In the linked list representation, the first node on the list is the current item that is the item at the top of the stack, and the last node is the node containing the bottom-most item.

→ A PUSH operation will add a new node in the front and a POP operation will remove a node from the front of the list.

→ The SIZE of the stack is not important here because this representation allows dynamic stacks instead of static stack.

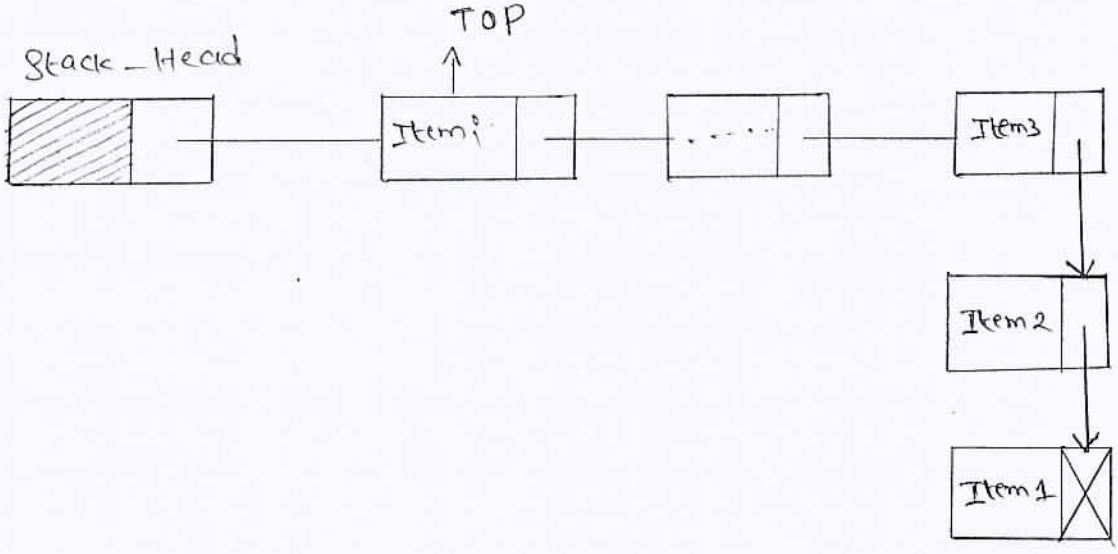



fig:. Linked list representation of a stack:.

## OPERATIONS ON STACKS:

The basic operations required to manipulate a stack are:

1. PUSH → To insert an item into a stack.
2. POP → To remove an item from a stack.
3. STATUS → To know the present state of a stack

The above operations for a stack represented with an array.

1. Algorithm Push-Array:

Steps:-

1. If TOP ≥ SIZE then
2. print "Stack is full".
3. Else
4. TOP = TOP+1
5. A[TOP] = ITEM
6. Endif.
7. Stop

2. Algorithm Pop-Array:

Steps:

1. If TOP < 1 then
2. Print "Stack is empty".
3. Else
4. ITEM = A[TOP]
5. TOP = TOP-1
6. Endif
7. Stop.

3. Algorithm Status - Array:

Steps:

1. If TOP < 1 then
2. print "Stack is Empty".
3. Else
4. If (TOP ≥ SIZE) then
5. Print "Stack is full".
6. Else
7. Print "The element at TOP is, A[TOP]
8. free = (SIZE - TOP) / SIZE * 100
9. Print "Percentage of free stack is", free
10. Endif
11. Endif.
12. Stop

The Operations for a Stack represented with Linked-list.

1) Algorithm PUSH - LL

Steps:

1. new = GetNode(NODE)
2. new → DATA = ITEM
3. new → LINK = TOP
4. TOP = new
5. STACK - HEAD → LINK = TOP
6. STOP.

2) Algorithm POP - LL

Steps:

1. If TOP = NULL
2. Print "Stack is empty".
3. Exit
4. Else
5. Ptr = TOP → LINK
6. ITEM = TOP → DATA
7. STACK_HEAD → LINK = Ptr
8. TOP = Ptr.
9. Endif.
10. Stop.

3) Algorithm STATUS - LL

Steps:

1. Ptr = STACK_HEAD → LINK
2. if (Ptr = NULL) then
3. Print "Stack is empty".
4. Else
5. node Count = 0
6. While (ptr ≠ NULL) do
7. node Count = node Count + 1
8. Ptr = Ptr → LINK
9. End while
10. Print " The Item at the front is ", TOP → DATA, "Stack Contains", Node Count, "Number of Items".
11. Endif
12. Stop

# APPLICATIONS OF STACKS:

The following are the various applications of stacks:

1. Evaluation of Arithmetic Expressions.
2. Code Generation for Stack Machines
3. Implementation of Recursion.
4. Factorial Calculation.
5. Quick sort
6. Tower of Hanoi problem
7. Activation Record Management.