Stack

When we collect a list of data items we organize them by defining a relationship in between them. According to this relationship the structure of data could be only either static or dynamic. A static structure is stored as a collection of mathematically related data items while a dynamic structure is stored as a collection of logically related data items.

There are so many ~~str~~ situations in which the above organisations can't fulfid our requirement in such situation we need to organise the array or linked list in some other logical way by specifying some constants. Stack is such a logicall data structure. Although; a stack is also stored as an array or as a linked list in the physical memory but in this structure we specify a point to where any where new items can be put and from where

any item can be deleted. The point from where insertion and deletion are done is called as top of the stack. This limita-tation enforces the stack to behave as a "LIFO" (Last in first out). The items which is placed on the stack at last is the items which is placed on the stack at last is the item also which will be deleted from the stack.

The best example of use of stack is run-time function stack managed by the operating system (OS). This ~~functio~~ run time stack makes the main program enable to be started first and ended at last when all the lower level functions are terminated. This is the concept of structu-red programming possible.

We can perform operation with the stack :-

(i) Creation of the stack
(ii) Insertion of data element to the stack
(iii) Deletion of data items from the stack
(iv) Traversal of the stack.

Inserting a data item to a stack is also called as push operation while deletion from the stack is called as pop operation.

Array representation of stack

To store the stacks as an array we need an array of same type and size and a variable to represent the top of the stack. Both these components can be declared seperately or we can define a structure having these ~~element~~ component as its member. In any way to make an empty stack is assigned with -1 that is increased before Pushing a data item and decreased after poping a data items.

Operation on array based stack

(a) Creation of stack - To create a stack first of all we define a structure with two members an array, and integer variable top after then we declare a

variable of this structure and assign the top with -1. It will follow the given algorithm —

Step1:- Create()
Step2:- Define structure stack with members array val * and variable Top.
Step3:- Declare variable S of stack type.
Step4:- let top of S = -1
Step5:- end

Push to the stack — The push operation is done by checking the stack first that either it has space or not to hold the new items, then after increasing the top and assigning the value to a A of top.
The algorithm is as given —

Step1:- Push(v)
Step2:- If Top = max-1 Then,
2.A:- Print stack is full
2.B:- Return
Step3:- End of if

Step 4:- let Top of S = Top of S+1
Step 5:- let val of S(Top of S) = V
Step 6:- Return

Popping from stack :- Deletion of an element from a stack is called Pop. It is done by checking the stack that either it is empty or not, decreasing the value of top.
The algorithm is —

Step 1:- Start Pop()
Step 2:- Declare variabe on
Step 3:- If top = -1 then,
3.A:- Print "stack empty"
3.B:- Return with 0
Step 4:- end of if
Step 5:- let n = S' val (S' Top)
Step 6:- let top of S = Top of S-1
Step 7:- Return with n.

Traversing the stack :- An array based stacked can be traversed from both the direction either from 0 to top or from top to 0 (zero). In the traversal algorithm we should also check for emptyness of the stack.

The algorithm is given below:-

Step 1:- Start show
Step 2:- Declare variable n
Step 3:- If Top = -1 Then
3.A:- Printf ("stack empty")
3.B:- return
Step 4:- end of If
Step 5:- for n=0 to top of s
Step 6:- Printf ("value=" +s'val[n])
Step 7:- Next n
Step 8:- Return

Linked list representation of stack-

The array representation limits the stack to hold a restric-ted number of elements. To overcome this limitation the stack can be stored as a linked list, which makes it dynamic and unrestricted. An ~~array~~ any other linked list the stack will also be a collec-tion of nodes. With the restric-tion that the insertion and deletion of the nodes can be done only from the beginning.

Creation of linked based stack :-

To create such a stack at first we define the structure node and then declare a pointer that is assigned with null. The algorithm is given below -

Step 1:- Start create
Step 2:- Define structure stack with the members val and pointer link.
Step 3:- Declare a pointer variable top of a stack type.
Step 4:- Let top = null.
Step 5:- return

Push operation :- To push a value to a linked stack we need to allocate a node, keep the value in the node and then adjust it with the top. The algorithm is given below -

Step 1:- start push(v)
Step 2:- Declare pointer ptr of stack type
Step 3:- Allocate memory to ptr
Step 4:- Let val of ptr = v
Step 5:- Let link of ptr = top

Step 6:- let top = ptr
Step 7:- return

Poping from linked :- It is based on stack to pop a value from a linked based stack at first we check that is the stack empty. if it is we just return back otherwise we keep the value of top in a pointer, set the top with the link of pointer and then deallocate the pointer. The algorithm is given below :-

Step 1 :- Start pop()
Step 2 :- Declare Ptr of stack type variable v
Step 3 :- If top = Null then,
3.A :- Print "Stack empty"
3.B :- Return with 0
Step 4 :- end of if
Step 5 :- Let ptr = Top
Step 6 :- Let Top = ptr's link
Step 7 :- Let v = val of ptr
Step 8 :- Deallocate ptr
Step 9 :- Return with v.