**STACKS**

* linear Data Structure
* elements are arranged in sequential
* Follows Last In First Out mechanism
* Example --- plates in a tray

|  |  |
| --- | --- |
| push | pop |
| Insertion of an element in to the stack | Deleting an element from stack |

* Both insertion & deletion takes place on a single side i.e. top of stack.
* Initially when there are no elements n the stack top =-1;
* To insert an element into the stack incrementing top.
* After deleting an element top will be decremented.
* **Using Arrays:-**

We have to decide the size of the stack

* While implementing stack we need to verify two conditions
* Overflow condition(push):- If size is 5 try to insert 6th element then overflow.
* Underflow condition(pop):-Try to delete from empty stack.
* Inserting an element which is already full.

Implementation:-

size=10;

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

top=-1

stack[]

push()

{

scanf("%d",&element);

if(top==size-1) 1

printf("Overflow"); 0

else

{ top=-1

top++;

stack[top]=ele;

}}

pop()

{

if(top==-1)

printf("stack underflow");

else

{

ele=stack[top];

printf("%d is popped", &ele);

top--;

}

}

Display()

{

if(top!=-1)

{

for (i=top;i>=0;i--)

{

printf("%d \t", stack[0]);

}

}

else

printf("empty stack");

}

**Stack Implementation using SLL(Linked List) :-**

using Arrays :-

Drawback:- we cannot place elements more than the size of an array.

**FILO**

struct node

{

int data;

struct node\*next;

}\*new\*temp,\*

**operatins on stack:-**

**push():-**

do

{

new=(struct node \*) malloc(sizeof(struct node);

scanf("%d", &element);

|  |
| --- |
|  |
| 30 |
| 20 |
| 10 |

|  |  |  |
| --- | --- | --- |
| top | 30 | 2000 |
|  |  | ↓ |
| top | 20 | 1000 |
|  |  | ↓ |
| top | 10 | NULL |

top=NULL

if(top=NULL)

{

new→data=element;

new→next=NULL;

top=new;}

else{

new→data=element;

new→next=top;

top=new;

}}

**POP()**

↓ top top ↓

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 30 | 2000 | → | 20 | 1000 | → | 10 | NULL |
| 3000 | |  | 2000 | |  | 1000 | |

↑temp

if(top==NULL)

{ top

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 20 | 1000 | → | 10 | NULL |
| 2000 | |  | 1000 | |

printf("stack is empty");

}

else

{

temp=to[;

ele=top→data;

printf("%d element is deleted", top→data);

top=top→next;

temp→next=NULL;

free(temp);

}

**Display():-** top

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 30 | 2000 | → | 20 | 1000 | → | 10 | NULL |
| 3000 | |  | 2000 | |  | 1000 | |

display()

{

if(top==NULL)

{

printf("stack is empty");

}

else

{

temp=top;

while(temp!=NULL)

{

printf("%d", temp→data);

temp=temp→next;

}

}