EXPERIMENT NO. 01 : To implement stack ADT using arrays

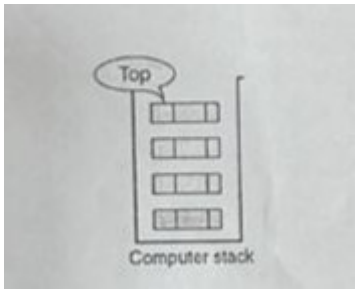
AIM: To implement stack ADT using arrays.

OBJECTIVE:

1.Understand the stack Data Structure and its basic operation.
2. Understand the method of defining stack ADT and implement the basic operation.
3. Learn how to create object from and ADT and invoke member function.

THEORY:

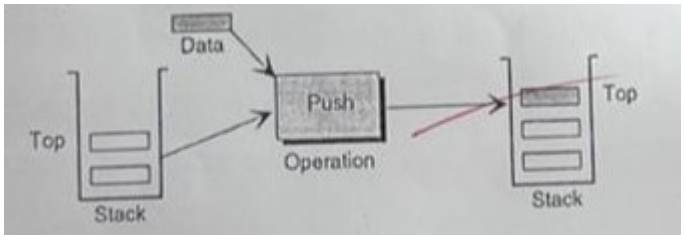
A stack is a list in which all insertions and detections are made at one end, called the top. It is a collection of contiguous cells, stacked on top of each other. The last element to be inserted into the stack will be the first to be removed. Thus stacks are sometimes referred to as Last In First Out (LIFO) lists.



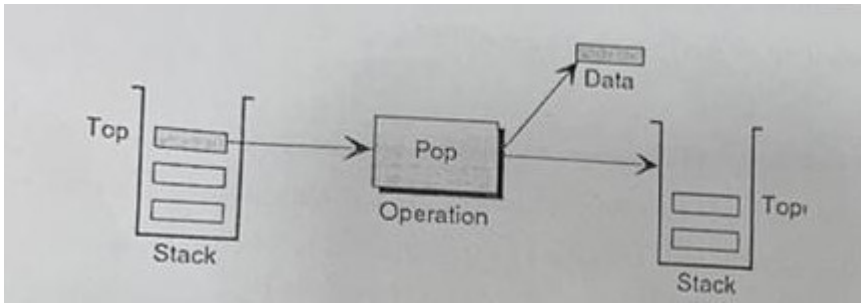
The operations that can be performed on a stack are push, pop which are main operations while auxiliary operations are peek, is Empty and is Full. Push is to insert an element at the top of the stack. Pop is deleting an element that is at the top most position in the stack. Peek simply examines and returns the top most value in the stack without deleting it.

Push on an already filled stack and pop on an empty stack result in serious errors so is Empty and is Full function checks for stack empty and stack full respectively. Before any insertion, the value of the variable top is initialized to -1.

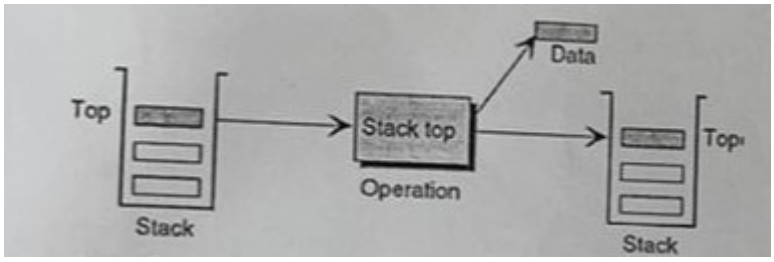
PUSH Operation



POP Operation



PEEK Operation



ALGORITHM:

PUSH(item)

1. If (stack is full)

   print "overflow"

2. top=top+1

3. stack[top]=item

   Return

POP()

1. if (stack is empty)

   print "undeflow"

2. Item=stack[top]

3. top=top-1

4. Return item

PEEK()

1. If (stack is empty)

   Print “underflow”

2. item=stack[top]

3. top=1

4. Return item

Code:

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

int x,stack[100],i,top,n,choice;
void push(void);
void pop(void);
void peak(void);
void display(void);
void main()
{
clrscr();
top=-1;
printf("ENTER THE SIZE OF Stack\n");
scanf("%d",&n);
printf("1.push\n2.pop\n3.peak\n4.display\n5.exit\n");
do
{
printf("ENTER YOUR Choice\n");
scanf("%d",&choice);
switch(choice)
{
    case 1:
    {
            push();
            break;
}
case 2:{
pop();
break;
}
case 3:{
peak();
break;
}
case 4:{
display();
break;
}
case 5:{
```

```
printf("EXIT POINT\n");
break;
}
default:{
printf("enter your valid choice:");
}
}
}
while(choice!=5);


return 0;
}

void push()
{
if(top>n-1)
{
printf("STACK IS OVERFLOW\n");

}

else
{
printf("ENTER A VALUE TO BE PUSHED:\n");
scanf("%d",&x);
top++;
stack[top]=x;
}
}
void pop()
{
if(top<=-1)
{
printf("STACK IS underflow\n");
}
else
{
printf("THE POPED ELEMENT is %d\n",stack[top]);
top--;
}
}
void peak()
{
printf("THE TOP ELEMENT OF THE STACK IS : %d\n",x);
}
void display()
{
if(top==0)
{
```

```
printf("THE ELEMENT IN THE STACK ARE:\n");
for(i=top;i>=0;i--)
{
printf("%d\n",stack[i]);
}
}
else
{
printf(" THE STACK IS EMPTY\n");
}
}
```

Output:

```
ENTER THE SIZE OF Stack
100
1.push
2.pop
3.peak
4.display
5.exit
ENTER YOUR Choice
1
ENTER A VALUE TO BE PUSHED:
45
ENTER YOUR Choice
4
THE ELEMENT IN THE STACK ARE:
45
ENTER YOUR Choice
1
ENTER A VALUE TO BE PUSHED:
55
ENTER YOUR Choice
2
THE POPED ELEMENT is 55
ENTER YOUR Choice
3
THE TOP ELEMENT OF THE STACK IS : 55
ENTER YOUR Choice
```

Conclusion :

In this experiment, the goal was to implement a stack Abstract Data Type (ADT) using arrays in C programming. A stack is a Last In First Out (LIFO) data structure where elements are added and removed from the top. The main operations - push (add), pop (remove), peek (view top), and display - were implemented using an array-based approach. Users interacted with the stack through a menu-driven interface.