EXPERIMENT NO 1

Topic : To implement stack ADT using arrays

Name : Vaibhav R Tatkare

Div / Roll No.: Comps 3 (C) / 52

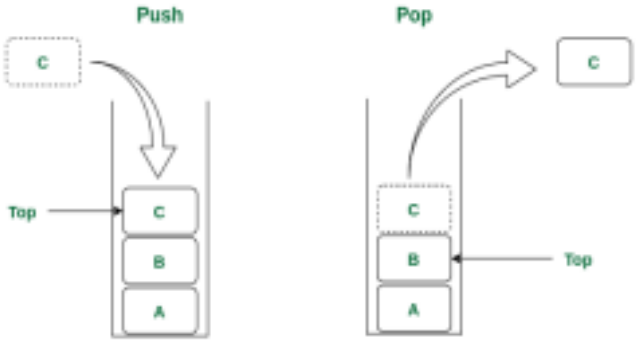
Aim: To implement stack ADT using arrays

Objective:

1) Understand the Stack Data Structure and its basic operators.
2) Understand the method of defining stack ADT and implement the basic operators. 3) Learn how to create objects from an ADT and invoke member functions.

Theory:

A stack is a linear data structure which uses the LIFO (Last-In-First-Out).principle. The elements in the stack are added and removed only from one end called the top. In the computer's memory, stacks can be represented as a linear array. Every stack has a variable called TOP associated with it, which is used to store the address of the topmost element of the stack. It is this position where the element will be added to or deleted from. There is another variable called MAX, which is used to store the maximum number of elements that the stack can hold. If TOP = NULL, then it indicates that the stack is empty and if TOP = MAX-1, then the stack is full.



Stack Data Structure

A stack supports a few basic operations : push, pop and display.

Push - The push operation is used to insert an element into the stack. The new element is added at the topmost position of the stack.

Pop - The pop operation is used to delete the topmost element from the stack.

Display – The display operation is used to display all the elements of the stack.

Peek - It is an operation that returns the value of the topmost element of the stack without deleting it from the stack.

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 "underflow"

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:

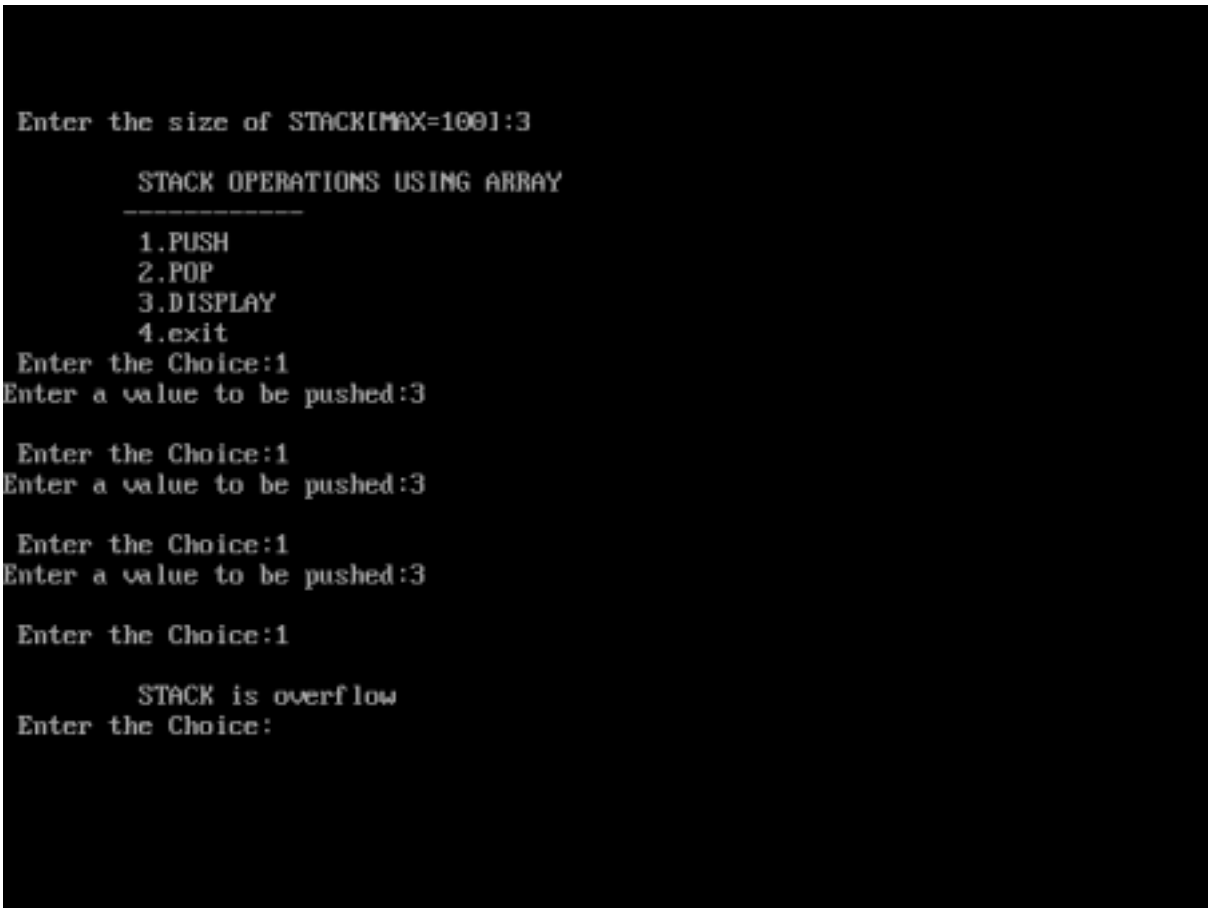
```
#include<stdio.h>
#include<conio.h>
int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{
top=-1;
printf("\n Enter the size of
STACK[MAX=100]:"); scanf("%d",&n);
```

```
printf("\n\t STACK OPERATIONS USING
ARRAY"); printf("\n\t----------");
printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t
4.exit"); do
{
printf("\n Enter the Choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:
{
push();
break;
}
case 2:
{
pop();
break;
}
case 3:
{
display();
break;
}
case 4:
{
printf("\n\t EXIT POINT");
```

```
break;
}
default:
{
printf("\n\t Please Enter a Valid
Choice(1/2/3/4)"); }
}
}
while(choice!=4);
return 0;
}
void push()
{
if(top>=n-1)
{
printf("\n\t STACK is overflow");
}
else
{
printf("Enter a value to be pushed:");
scanf("%d",&x);
top++;
stack[top]=x;
}
}
void pop()
{
```

```
if(top<=1)
{
printf("\n\t Stack is underflow");
}
else
{
printf("\n\t The popped elements is
%d",stack[top]); top--;
}
}
void display()
{
if(top>=0)
{
printf("\n The elements in STACK \n");
for(i=top;i>=0;i--)
printf("\n%d",stack[i]);
printf("\n The Stack is empty");
}
}
```

Output:



Conclusion:

We can conclude that the implementation of a stack Abstract Data Type (ADT) using arrays in C provides a straightforward and efficient way to manage Last-In-First-Out (LIFO) data structures. By utilizing an array for storage and tracking the top index, we achieve constant-time complexity for push and pop operations. However, care must be taken to handle potential overflow and underflow scenarios to ensure the integrity of the stack.