Name: Suvidhi V. Pareek

Div: 3/ ROLL NO. 10

BATCH: A

**EXPERIMENT NO. 1**

**TO IMPLEMENT STACK ADT USING ARRAYS**

**AIM: To implement stack ADT using arrays.**

**OBJECTIVE:**

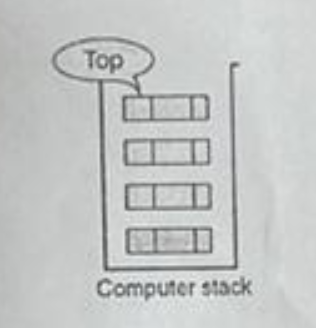
**1 Understand the stack Data Structure and its basic operators.**

**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:**

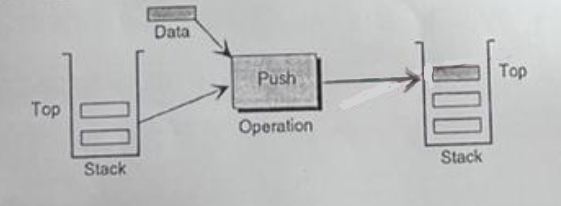
**A stack is a list in which all insertions and deletions are made at one end, called the top. It is a collection of contiguous cells, stacked on top of each other. The last element to be inserted into the stack will be the first to be removed. Thus stacks are sometimes referred to as Last In First Out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

****

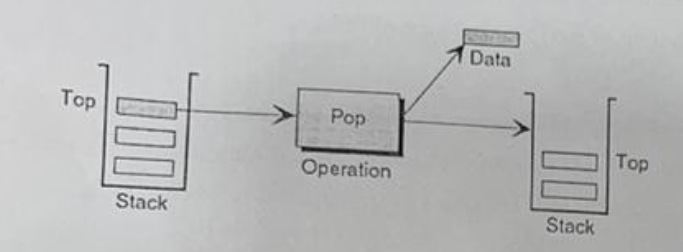
**The operations that can be performed on a stack are push, pop which are main operations while auxiliary operations are peek, isEmpty and isFull. Push is to insert an element at the top of the stack Pop is deleting an element that is at the top most position in the stack. Peek simply examines and returns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

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

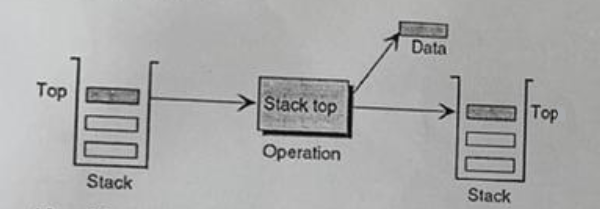
**Push Operations**

****

**Pop Operations**

****

**Peek Operations**

****

**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”**

**CODE:**

**#include<stdio.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\tSTACK is over flow");**

**}**

**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 under flow");**

**}**

**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 Press Next Choice");**

**}**

**else**

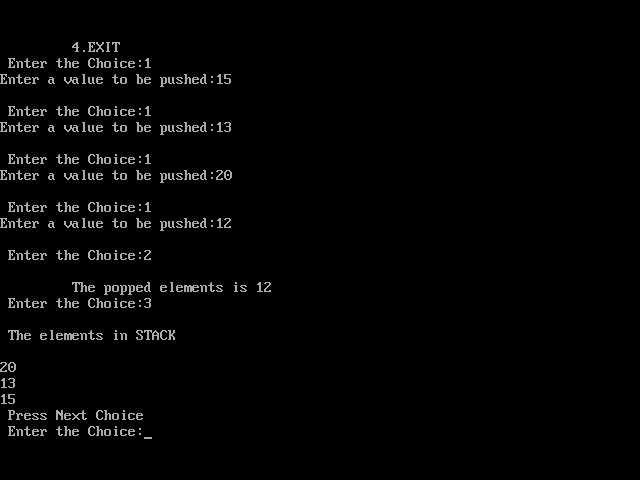
**{**

**printf("\n The STACK is empty");**

**}**

**}**

**OUTPUT:**

****

**Conclusion:**

1. **Implementing stack ADT using arrays involves use of array to store the elements of the stack and keeping track of the top element using a variable.**
2. **The size of the stack is fixed, so it cannot increase and decrease stack implementation using an array.**
3. **More efficient in terms of time.**