Experiment No.1
Implement Stack ADT using array.
Name: Tejashree Anand Karekar
Roll no: 20
Date of Performance:
Date of Submission:
Marks:
Sign:

Experiment No. 1: To implement stack ADT using arrays

Aim: To implement stack ADT using arrays.

Objective:

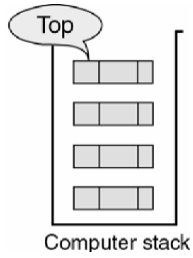
1) Understand the Stack Data Structure and its basic operators.

2) Understand the method of defining stack ADT and implement the basic operators.

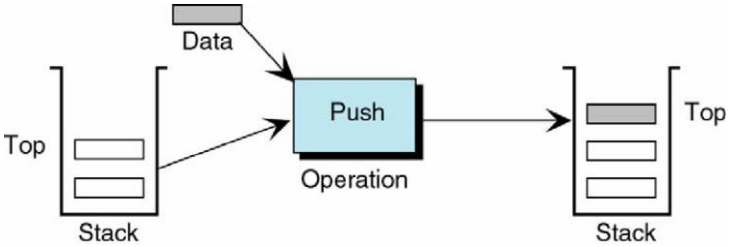
3) Learn how to create objects from an ADT and invoke member functions.

Theory:

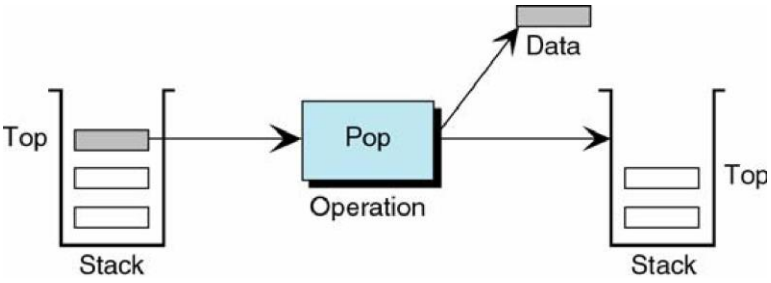
A stack is a data structure where all insertions and deletions occur at one end, known as the top. It follows the Last In First Out (LIFO) principle, meaning the last element added to the stack will be the first to be removed. Key operations for a stack are "push" to add an element to the top, and "pop" to remove the top element. Auxiliary operations include "peek" to view the top element without removing it, "isEmpty" to check if the stack is empty, and "isFull" to determine if the stack is at its maximum capacity. Errors can occur when pushing to a full stack or popping from an empty stack, so "isEmpty" and "isFull" functions are used to check these conditions. The "top" variable is typically initialized to -1 before any insertions into the stack.



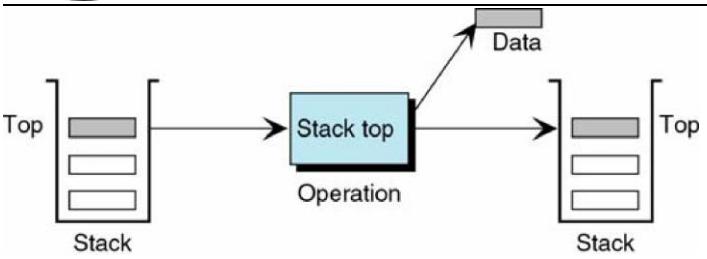
Push Operation



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 "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. Return item

ISEMPTY()

1. If(top = -1)then

return 1

2. return 0

ISFULL()

1. If(top = max)then

return 1

2. return 0

Code:

```
#include<stdio.h>

#include<stdlib.h>


int stack[100],choice,n,top,x,i;

void push(void);

void pop(void);

void display(void);

void peek();


int main()

{

top=-1;

clrscr();

printf("Enter the size of stack[max=100]:");

scanf("%d",&n);

printf("Stack operation using array\n");
```

```
printf("\n\t 1.PUSH \n\t 2.POP \n\t 3.PEEK \n\t 4.DISPLAY \n\t 5.EXIT");
        do
        {
                printf("\nEnter your choice:");
                scanf("%d",&choice);

                switch(choice)
                {
                        case 1:
                        {
                                push();
                                break;
                        }
                        case 2:
                        {
                                pop();
                                break;
                        }
                        case 3:
                        {
                                peek();
                                break;
                        }
                        case 4:
```

```
                        {
                                display();
                                break;
                        }
                        case 5:
                        {
                                printf("\n\tEXIT POINT");
                                break;
                        }
                        default:
                        {
                                printf("\n\t Please enter a valid choice(1/2/3/4)");
                        }
                }
        }
        while(choice!=5);
return 0;
}

void push()
{
        if(top>=n-1)
        {
                printf("\n\t Stack is 'OVERFLOW' ");
```

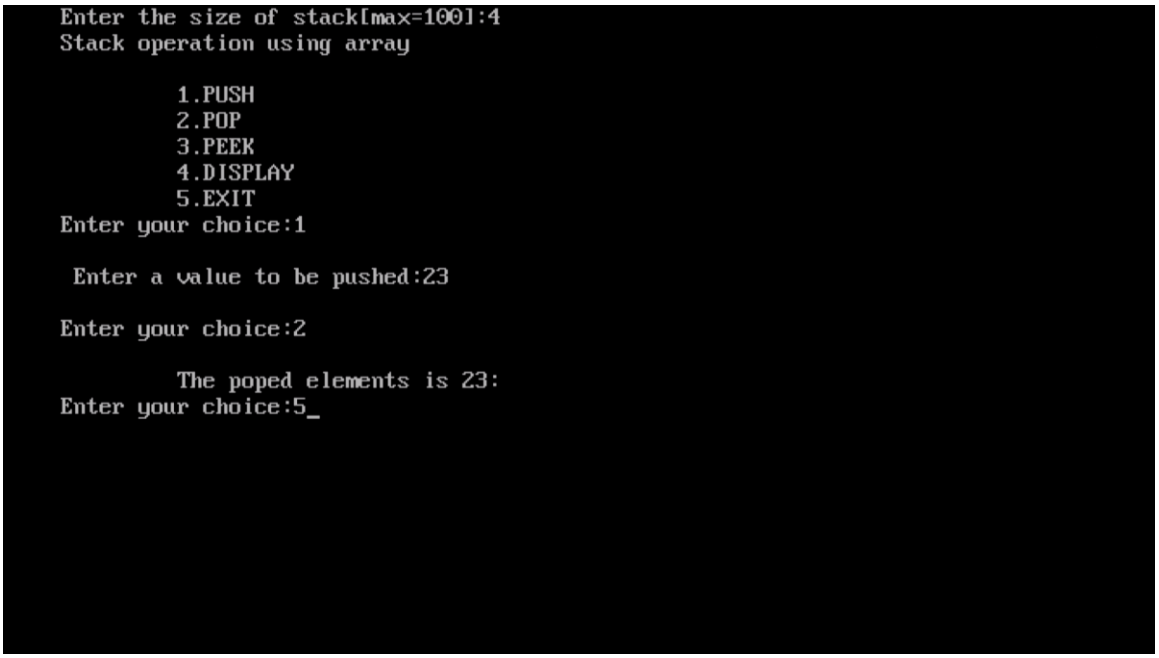
```
        }
        else
        {
                printf("\n Enter a value to be pushed:");
                scanf("%d",&x);
                top++;
                stack[top]=x;
        }
}
void pop()
{
        if(top<=-1)
        {
                printf("\nStack is 'UNDERFLOW' ");
        }
        else
        {
                printf("\n\t The poped elements is %d:",stack[top]);
                top--;
        }
}
void display()
{
        if(top>=0)
```

```
        {
                printf("\n The element in stack:");
                for(i=top;i>=0;i--)
                {
                        printf("\n%d",stack[i]);
                        printf("\nPress next choice");
                }
        }
        else
        {
                printf("\nThe stack is empty");
        }

}
void peek()
{
        if(top<=-1)
        {
                printf("\n stack is Underflow");
        }
        else
        {
                printf("\n The peek element is %d:",stack[top]);
        }
```

}

Output:



Conclusion:

1)What is the structure of Stack ADT?

A Stack is an abstract data type (ADT) that represents a collection of elements with two primary operations: push and pop. It follows the Last-In-First-Out (LIFO) principle, which means that the most recently added element is the first one to be removed

2)List various applications of stac

- Various applications of stacks include:

- Functionality
- ,Backtracking
- ,Expression Evaluation,
- Function Call Stack

3)Which stack operation will be used when the recursive function call is returning to the calling function?

When a recursive function call is returning to the calling function, the stack operation used is "pop." This operation involves removing the context (local variables and return address) of the current function call from the top of the call stack. This action allows the program to continue executing from the point where the function was originally called.