Experiment No.1
Implement Stack ADT using array.
Name: Nitish Jha
Roll no: 18
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:

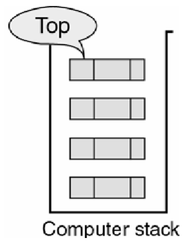
1) Understand the Stack Data Structure and its basic operators.

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

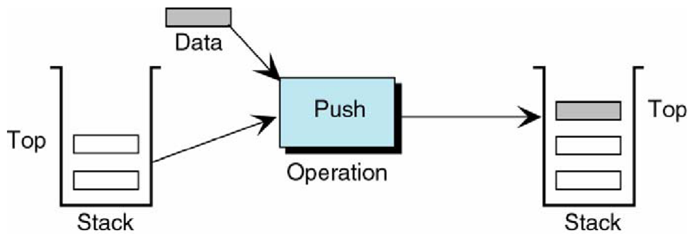
3) Learn how to create objects from an ADT and invoke member functions.

Theory:

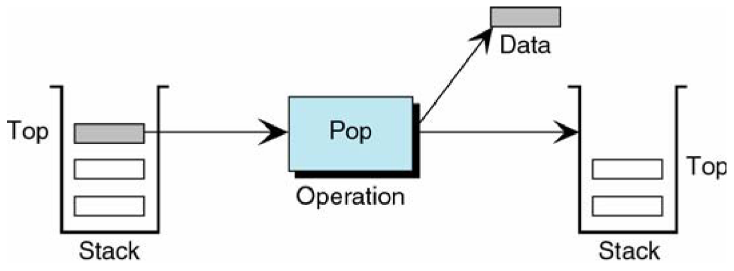
A stack is a data structure where all insertions and deletions occur at one end, known as the top. It follows the Last In First Out (LIFO) principle, meaning the last element added to the stack will be the first to be removed. Key operations for a stack are "push" to add an element to the top, and "pop" 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.



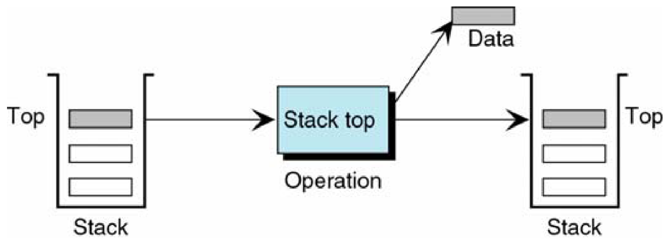
Push Operation



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
```

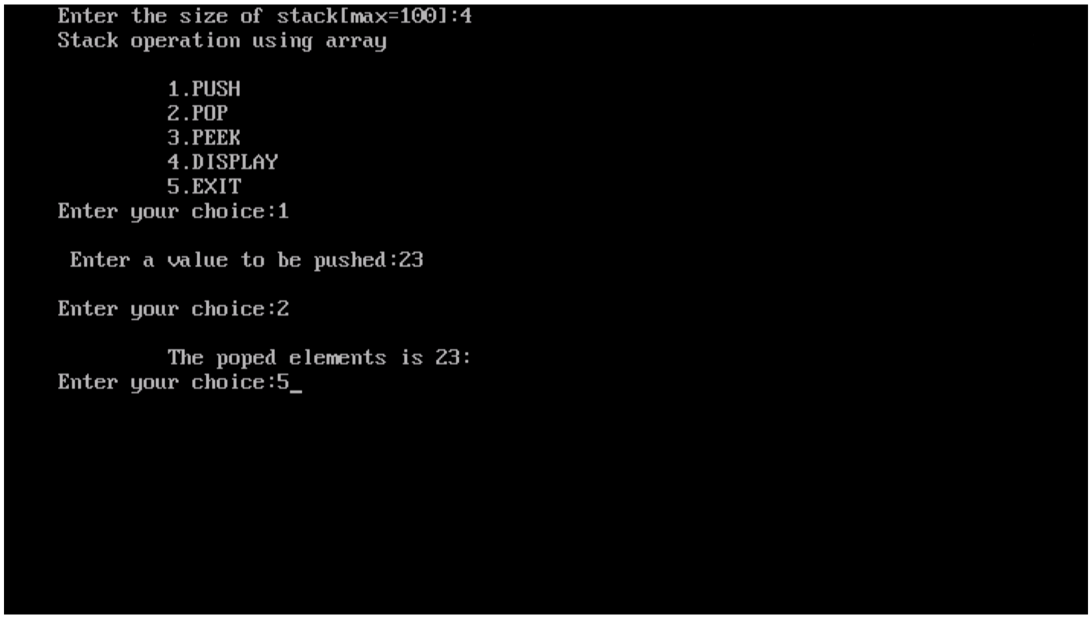
Code:

```
#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:



```
Enter the size of stack[max=100]:4
Stack operation using array

         1.PUSH
         2.POP
         3.PEEK
         4.DISPLAY
         5.EXIT
Enter your choice:1

 Enter a value to be pushed:23

Enter your choice:2

         The poped elements is 23:
Enter your choice:5_
```

Conclusion:

1)What is the structure of Stack ADT?

- A Stack Abstract Data Type (ADT) is a linear structure that adheres to the Last-In-First-Out (LIFO) principle. It's essentially a container for elements, with two primary operations: "push" allows you to add an element to the top of the stack, and "pop" lets you remove the top element. Stacks often include other operations like "peek," which allows you to look at the top element without taking it off the stack, and "isEmpty" to determine if the stack is devoid of elements.

2)List various applications of stack?

- Stacks find application in various scenarios:

- They're crucial for managing function calls in programming, especially during recursion.

- Software applications rely on stacks for implementing undo mechanisms.

- Stacks are used for efficient evaluation of expressions, particularly in arithmetic expressions with parentheses.

- In computer systems, stacks play a role in memory management, particularly in call stacks.

- Compilers employ stacks for parsing and syntax analysis.

- Stacks are instrumental in backtracking algorithms, aiding in solving puzzles like Sudoku and mazes.

- Web browsers use stacks for managing browser history, facilitating smooth navigation.

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

- In the realm of recursive function calls, the pivotal operation is "pop." It comes into play when a function is done and needs to hand control back to the calling function. This operation involves removing the top item from the function call stack, which effectively allows the program to smoothly transition back to executing the calling function. This method of "popping" is what enables the recursion stack to efficiently manage the sequence of function calls and their return values, following the Last-In-First-Out (LIFO) principle.