Experiment No.1
Implement Stack ADT using array.
Name: Vedika Pawar
Roll no:46
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.

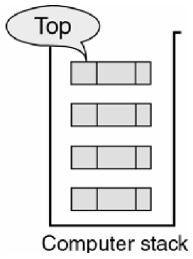
2) Understand the method of defining stack ADT and implement the basic operators.

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

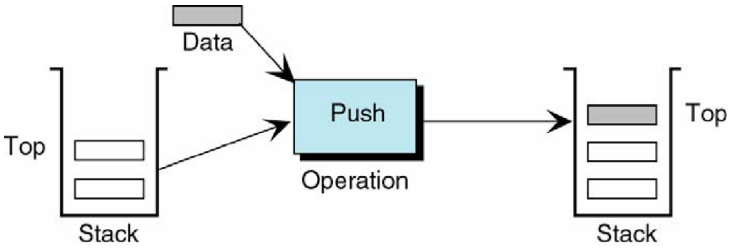
Theory:

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

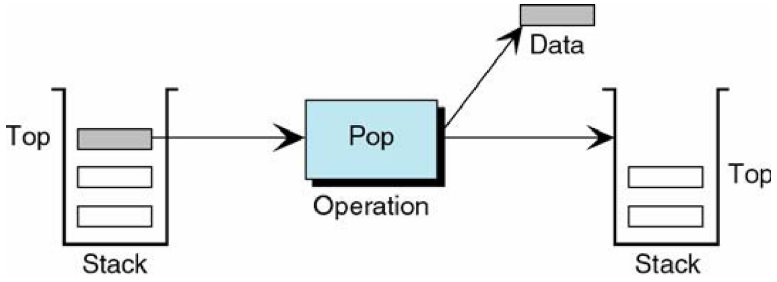
these conditions. The "top" variable is typically initialized to -1 before any insertions into the stack.



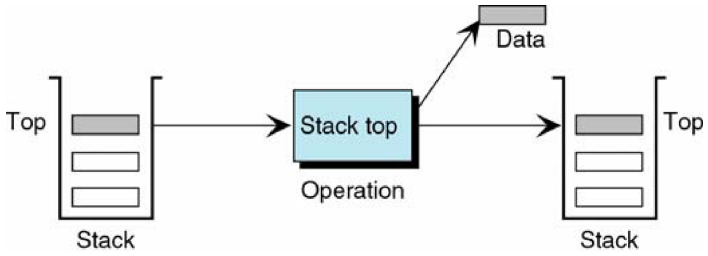
Push Operation



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>
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:

```
Enter the size of STACK[MAX=100]:10

         STACK OPERATIONS USING ARRAY
        --------------------------------
         1.PUSH
         2.POP
         3.DISPLAY
         4.EXIT
 Enter the Choice:1
 Enter a value to be pushed:12

 Enter the Choice:1
 Enter a value to be pushed:24

 Enter the Choice:1
 Enter a value to be pushed:98

 Enter the Choice:3

 The elements in STACK

98
24
12
 Press Next Choice
 Enter the Choice:2

         The popped elements is 98
 Enter the Choice:3

 The elements in STACK

24
12
 Press Next Choice
 Enter the Choice:4

         EXIT POINT
```

Conclusion:

Q.What is the structure of Stack ADT?

The Stack Abstract Data Type (ADT) is a linear data structure that follows the Last-In-First Out (LIFO) principle.

It has two primary operations:

1. Push: This operation adds an element to the top of the stack.

2. Pop: This operation removes and returns the element from the top of the stack. In addition to these fundamental operations, a stack typically has the following characteristics:

Peek (or Top): This allows you to view the element at the top of the stack without removing t. IsEmpty: This function checks if the stack is empty.

- Size: This function returns the number of elements in the stack. A simple real-life analogy is a stack of plates: you add plates to the top (Push) and remove them from the top (Pop).

Q.List various applications of stack?

1. Expression evaluation

2. Function call management

3. Backtracking algorithms

4. Undo functionality

5. Memory management

6. Syntax parsing

7. Browser history

8. Task management

9. Expression matching

10. Postfix calculations

11. Undo/redo in text editors

12. Playlists

13. Call history

14. Task scheduling

15. Routing algorithms

16. Symbol balancing

17. Navigation systems

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

The stack operation used when a recursive function call is returming to the calling function is called "Pop." When the recursive function completes its execution and returns a value, the information related to that function call (such as local variables, return address, and other context) is popped or removed from the call stack. This allows the program to resume execution in the calling function from the point where the recursive call was made.