

**K. J. Somaiya College of Engineering, Mumbai**  
(A Constituent College of Somaiya Vidyavihar University)  
**Department of Computer Engineering**

**Batch: B4      Roll No.: 16010122828**

**Experiment / assignment / tutorial No. 3**

**Grade: AA / AB / BB / BC / CC / CD / DD**

**Signature of the Staff In-charge with date**

**Title:** Implementation of Basic operations on stack for the assigned application using Array and Linked List- Create, Insert, Delete, Peek.

**Objective:** To implement Basic Operations on Stack i.e. Create, Push, Pop, Peek for the given application

**Expected Outcome of Experiment:**

CO	Outcome
1	Explain the different data structures used in problem solving

**Books/ Journals/ Websites referred:**

1. *Fundamentals Of Data Structures In C* – Ellis Horowitz, Satraj Sahni, Susan Anderson-Fred
2. *An Introduction to data structures with applications* – Jean Paul Tremblay, Paul G. Sorenson
3. *Data Structures A Pseudo Approach with C* – Richard F. Gilberg & Behrouz A. Forouzan
4. <https://www.cprogramming.com/tutorial/computersciencetheory/stack.html>
5. <https://www.geeksforgeeks.org/stack-data-structure-introduction-program/>
6. <https://www.thecrazyprogrammer.com/2013/12/c-program-for-arrayrepresentation-of-stack-push-pop-display.html>

**K. J. Somaiya College of Engineering, Mumbai**  
(A Constituent College of Somaiya Vidyavihar University)  
**Department of Computer Engineering**

**Abstract:**

(Define stack, enlist stack operations). **Stack:**

A stack is a linear data structure. Any operation on the stack is performed in LIFO (Last In First Out) order. This means the element to enter the container last would be the first one to leave the container. It is imperative that elements above an element in a stack must be removed first before fetching any element.

**Operations on a Stack:**

**Push:**

By pushing, we mean inserting an element at the top of the stack. While using the arrays, we have the index to the top element of the array. So, we'll just insert the new element at the index (top+1) and increase the top by 1. It's intuitive to note that this operation is valid until (top+1) is a valid index and the array has an empty space.

**Pop:**

Pop means to remove the last element entered in the stack, and that element has the index top. So, this becomes an easy job. We'll just have to decrease the value of the top by 1, and we are done. The popped element can even be used in any way we like.

**Peek:**

It refers to looking for the element at a specific index in a stack. Peek operation requires the user to give a position to peek at as well. Here, position refers to the distance of the current index from the top element +1

**List 5 Real Life applications:**

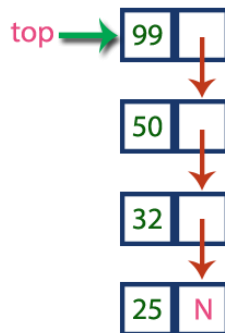
1. Reversing a String
2. Checking for parenthesis matching
3. Infix to Postfix conversion
4. Evaluation of Postfix expression
5. Undo-Redo operations

**K. J. Somaiya College of Engineering, Mumbai**  
(A Constituent College of Somaiya Vidyavihar University)  
**Department of Computer Engineering**

**Algorithm for stack operations using array/Linked list :**

To implement a stack using a linked list, we need to set the following things before implementing actual operations.

- **Step 1** - Include all the **header files** which are used in the program. And declare all the **user defined functions**.
- **Step 2** - Define a '**Node**' structure with two members **data** and **next**.
- **Step 3** - Define a **Node** pointer '**top**' and set it to **NULL**.
- **Step 4** - Implement the **main** method by displaying Menu with list of operations and make suitable function calls in the **main** method.



**K. J. Somaiya College of Engineering, Mumbai**  
(A Constituent College of Somaiya Vidyavihar University)  
**Department of Computer Engineering**

**Stack ADT:**

**Implementation of Stack ADT using Arrays**

```
#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:
```

## **K. J. Somaiya College of Engineering, Mumbai**

(A Constituent College of Somaiya Vidyavihar University)

### **Department of Computer Engineering**

```
{
    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");
    }
}
```

**K. J. Somaiya College of Engineering, Mumbai**  
(A Constituent College of Somaiya Vidyavihar University)  
**Department of Computer Engineering**

```
}  
else  
{  
    printf("\n The STACK is empty");  
}  
  
}
```

**Output Screenshots:**

```
Output  
/tmp/OcquJOFvDV.o  
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:45  
Enter the Choice:1  
Enter a value to be pushed:89  
Enter the Choice:1  
Enter a value to be pushed:01  
Enter the Choice:3  
The elements in STACK  
  
1  
89  
45  
Press Next Choice  
Enter the Choice:2  
The popped elements is 1  
Enter the Choice:3  
The elements in STACK  
  
89  
45  
Press Next Choice  
Enter the Choice:4  
EXIT POINT
```

**Explain the Importance of the approach followed by you**

In postfix , as the name suggests, the operator is placed afterwards For example, if an expression is written as  $A+B$  in infix notation, the same expression can be written as  $AB+$  in postfix notation. The order of evaluation of a postfix expression is always from left to right. Even brackets cannot alter the order of evaluation.

The expression  $A + B * C$  can be written as  $\Rightarrow AB+C*$  in the postfix notation

**PostLab Questions:**

**1) Explain how Stacks can be used in Backtracking algorithms with example.**

- Consider the N-Queens problem for an example. The solution of this problem is that N queens should be positioned on a chessboard so that none of the queens can attack another queen. In the generalized N-Queens problem, N represents the number of rows and columns of the board and the number of queens which must be placed in safe positions on the board.
- The basic strategy we will use to solve this problem is to use backtracking. In this particular case, backtracking means we will perform a safe move for a queen at the time we make the move. Later, however, we find that we are stuck and there is no safe movement for the next Queen, whom we are trying to put on the blackboard, so we have to go back.
- This means we return to the queen's previous move, undo this step, and continue to search for a safe place to place this queen.
- We will use the stack to remember where we placed queens on the board, and if we need to make a backup, the queen at the top of the stack will be popped, and we will use the position of that queen to resume the search for next safe location.
- Hence, stacks can be used for the allocation of memory for most of the backtracking problems.

**K. J. Somaiya College of Engineering, Mumbai**  
(A Constituent College of Somaiya Vidyavihar University)  
**Department of Computer Engineering**

**2) Illustrate the concept of Call stack in Recursion.**

- a. In recursion, the goal—the end condition—is always the **base case**.
- b. It's the bottom of the barrel, the end of the recursive calls. If a recursive method calls itself, it has to stop when it hits the base case and work its way back up to the original method.
- c. Each recursive function calls itself until hitting the base. Then, when the case becomes true, the result floats back to the top of the **call stack**, a list in the computer that keeps track of all the functions called in a program.
- d. The fact that programming languages have call stacks is hugely important to recursive functions.

**Conclusion:**

The aim of the experiment is verified.