DATA STRUCTURES LABORATORY

Course Code: 1BCSL306

VTU 2025 Scheme – Semester 3

EXPERIMENT – 03

Stack Operations

Array Implementation of Stack

Simple Code • Easy to Remember • Easy to Write in Exam

Your VTU Study Partner

Experiment 03

1. Aim

Aim

To develop a menu-driven C program to perform stack operations using an array and to demonstrate stack overflow and underflow.

2. VTU Prescribed Experiment

VTU Prescribed Experiment

Develop a menu driven Program in C for the following operations on **STACK of Integers** (Array Implementation of Stack with maximum size **MAX** $>$ **15**).

a. Push an Element on to Stack.

b. Pop an Element from Stack.

c. Demonstrate Overflow and Underflow situations on Stack.

d. Display the status of Stack at any given point of time.

e. Exit.

Support structured program with appropriate functions for each of the above operations.

3. Simple Theory

Stack:

A stack is a linear data structure in which insertion and deletion take place only at one end called **TOP**.

Stack follows:

$$\boxed{LIFO}$$

where:

$$\boxed{LIFO = Last\ In\ First\ Out}$$

Example:

$$10,\ 20,\ 30$$

Here, 30 is inserted last and will be removed first.

4. Basic Stack Operations

4.1. Push

Push means inserting an element into the stack.

$$\boxed{top = top + 1}$$

Then:

$$\boxed{stack[top] = item}$$

4.2. Pop

Pop means deleting the top element from the stack.
First:

$$\boxed{item = stack[top]}$$

Then:

$$\boxed{top = top - 1}$$

4.3. Overflow

Overflow occurs when we try to insert an element into a full stack.
Condition:

$$\boxed{top == MAX - 1}$$

4.4. Underflow

Underflow occurs when we try to delete an element from an empty stack.
Condition:

$$\boxed{top == -1}$$

4.5. Initial Value of TOP

Initially:

$$\boxed{top = -1}$$

This means the stack is empty.

5. Algorithm

5.1. Main Algorithm

1. Start.
2. Declare stack array of size `MAX`.
3. Initialize:

$$top = -1$$

4. Display menu:
   - Push
   - Pop
   - Display
   - Exit
5. Read user's choice.
6. Call the required function.
7. Repeat until Exit is selected.
8. Stop.

5.2. Algorithm for Push

1. Check whether:

$$top == MAX - 1$$

2. If true, display:

$$\boxed{Stack\ Overflow}$$

3. Otherwise read the element.
4. Increment:

$$top = top + 1$$

5. Insert the element:

$$stack[top] = item$$

5.3. Algorithm for Pop

1. Check whether:

$$top == -1$$

2. If true, display:

$$\boxed{Stack\ Underflow}$$

3. Otherwise display the element:

$$stack[top]$$

4. Decrement:

$$top = top - 1$$

5.4. Algorithm for Display

1. Check whether:

$$top == -1$$

2. If true, display "Stack is Empty".
3. Otherwise start from `top`.
4. Display all elements up to index 0.

Easy Exam Memory

Just remember these three conditions:

$$\boxed{top = -1}$$

means Stack Empty.

$$\boxed{top == MAX - 1}$$

means Overflow.

$$\boxed{top == -1}$$

means Underflow.

For Push:

$$\boxed{++top \rightarrow stack[top] = item}$$

For Pop:

$$\boxed{stack[top] \rightarrow --top}$$

6. C Program – Easy Student Version

```c
#include<stdio.h>

#define MAX 20

int stack[MAX];
int top=-1;

void push()
{
    int item;

    if(top==MAX-1)
    {
        printf("Stack Overflow\n");
        return;
    }

    printf("Enter element: ");
    scanf("%d",&item);

    stack[++top]=item;

    printf("Element Pushed\n");
}

void pop()
{
    if(top==-1)
    {
        printf("Stack Underflow\n");
        return;
    }

    printf("Deleted element = %d\n",stack[top]);

    top--;
}

void display()
{
    int i;

    if(top==-1)
    {
        printf("Stack is Empty\n");
        return;
    }

    printf("Stack Elements:\n");

```

```
    for(i=top;i>=0;i--)
        printf("%d\n",stack[i]);
}

int main()
{
    int ch;

    do
    {
        printf("\n1.Push");
        printf("\n2.Pop");
        printf("\n3.Display");
        printf("\n4.Exit");

        printf("\nEnter Choice: ");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1:
                push();
                break;

            case 2:
                pop();
                break;

            case 3:
                display();
                break;

            case 4:
                printf("Program Ended\n");
                break;

            default:
                printf("Invalid Choice\n");
        }

    }while(ch!=4);

    return 0;
}
```


7. Simple Program Explanation

7.1. Stack Declaration

```
#define MAX 20
```

```
int stack[MAX];
int top=-1;
```

`MAX` is set to 20.

Therefore:

$$MAX > 15$$

as required in the syllabus.

Initially:

$$top = -1$$

which means the stack is empty.

7.2. push() Function

First check:

```
if(top==MAX-1)
```

If true:

$$\boxed{Overflow}$$

Otherwise:

```
stack[++top]=item;
```

This increments `top` and inserts the new element.

7.3. pop() Function

First check:

```
if(top==-1)
```

If true:

$$\boxed{Underflow}$$

Otherwise:

```
printf("%d",stack[top]);
top--;
```

The top element is removed.

7.4. display() Function

The display function starts from:

$$\boxed{top}$$

and prints elements up to:

$$\boxed{0}$$

Therefore, the topmost element is displayed first.

8. Stack Example

Suppose we Push:

$$10,\ 20,\ 30$$

Then the stack becomes:

$30 \leftarrow TOP$
20
10

If Pop is performed:

$$30$$

is deleted.

New stack:

$20 \leftarrow TOP$
10

Easy Exam Memory

The complete program is only:

$$\boxed{Array \rightarrow Top \rightarrow Push \rightarrow Pop \rightarrow Display \rightarrow Main}$$

Push:

$$\boxed{Check\ Full \rightarrow Read\ Item \rightarrow ++top \rightarrow Insert}$$

Pop:

$$\boxed{Check\ Empty \rightarrow Print\ Top \rightarrow top--}$$

Display:

$$\boxed{for(i = top; i >= 0; i--)}$$

9. Sample Input and Output

```
1.Push
2.Pop
3.Display
4.Exit

Enter Choice: 1

Enter element: 10
Element Pushed


Enter Choice: 1

Enter element: 20
Element Pushed


Enter Choice: 1

Enter element: 30
Element Pushed


Enter Choice: 3

Stack Elements:
30
20
10


Enter Choice: 2

Deleted element = 30


Enter Choice: 3

Stack Elements:
20
10


Enter Choice: 2

Deleted element = 20


Enter Choice: 2
```

```
Deleted element = 10


Enter Choice: 2

Stack Underflow


Enter Choice: 4

Program Ended
```

10. Overflow Demonstration

The stack size is:

$$MAX = 20$$

When:

$$top = 19$$

the stack is full.

If Push is attempted again:

$$top == MAX - 1$$

Therefore the program displays:

```
Stack Overflow
```

11. Underflow Demonstration

When:

$$top = -1$$

the stack is empty.

If Pop is attempted:

$$top == -1$$

Therefore the program displays:

```
Stack Underflow
```

12. Simple Test Cases

Test	Operation	Expected Result
1	Push 10	10 should be inserted.
2	Push 20 and 30	Stack should contain 30, 20, 10.
3	Pop once	30 should be deleted.
4	Display Stack	20 and 10 should be displayed.
5	Pop from empty stack	Stack Underflow should be displayed.
6	Push when stack is full	Stack Overflow should be displayed.

13. Result

Result

Thus, the C program for implementing a stack using an array was successfully executed. The program performs:

- Push Operation
- Pop Operation
- Stack Overflow Demonstration
- Stack Underflow Demonstration
- Display of Stack Elements

14. Important Viva Questions

Important Viva Questions

1. What is a stack?
A stack is a linear data structure where insertion and deletion take place at one end called TOP.

2. Which principle does stack follow?

$$\boxed{LIFO}$$

Last In First Out.

3. What is Push?
Push is the operation of inserting an element into the stack.

4. What is Pop?
Pop is the operation of deleting the top element from the stack.

5. What is TOP?
TOP stores the index of the current top element of the stack.

6. What is the initial value of TOP?

$$\boxed{-1}$$

7. What is stack overflow?
Overflow occurs when Push is performed on a full stack.

8. What is the overflow condition?

$$\boxed{top == MAX - 1}$$

9. What is stack underflow?
Underflow occurs when Pop is performed on an empty stack.

10. What is the underflow condition?

$$\boxed{top == -1}$$

11. What happens to TOP during Push?
TOP is incremented by one.

$$\boxed{top = top + 1}$$

12. What happens to TOP during Pop?
TOP is decremented by one.

$$\boxed{top = top - 1}$$

13. What is the time complexity of Push?

$$\boxed{O(1)}$$

14. What is the time complexity of Pop?

15. 1-Minute Exam Revision

Easy Exam Memory

Principle

$$\boxed{LIFO}$$

Initial TOP

$$\boxed{top = -1}$$

Push

$$\boxed{stack[++top] = item}$$

Pop

$$\boxed{stack[top]}$$

then:

$$\boxed{top--}$$

Overflow

$$\boxed{top == MAX - 1}$$

Underflow

$$\boxed{top == -1}$$

Display

$$\boxed{for(i = top; i >= 0; i--)}$$

16. Easy Code Memory Map

Remember

Remember just:

$$\boxed{MAX \rightarrow stack[] \rightarrow top = -1}$$

Then three functions:

$$\boxed{push() \rightarrow pop() \rightarrow display()}$$

Finally:

$$\boxed{main() \rightarrow menu \rightarrow switch}$$

The easiest memory trick is:

$$\boxed{Push = Check\ Full + Increase\ TOP}$$

$$\boxed{Pop = Check\ Empty + Decrease\ TOP}$$

SEARCH CREATORS

1BCSL306 – Data Structures Laboratory

EXPERIMENT 03
COMPLETED

Stack Operations

Array Implementation of Stack

Understand → Remember → Write → Execute → Viva

Your VTU Study Partner