Unit - II : Multiple Stacks

Saturday, May 3, 2025 6:40 PM

✅ Description of Multi Stack Program (Array-Based)

This C program implements **two stacks** (Left Stack and Right Stack) in a **single array** using efficient space management.

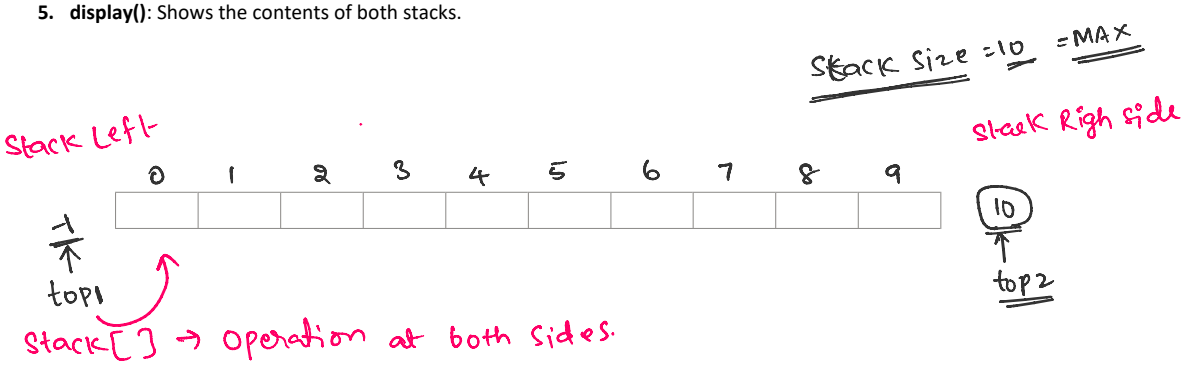
- The **Left Stack** grows from the **beginning (index 0)** of the array.
- The **Right Stack** grows from the **end (index MAX - 1)** of the array.

Two separate top variables are used:

- top1 for the Left Stack (initialized to -1)
- top2 for the Right Stack (initialized to MAX)

✅ Operations Implemented:

1. **push1(value)**: Adds an element to the Left Stack.
2. **push2(value)**: Adds an element to the Right Stack.
3. **pop1()**: Removes and returns the top element of the Left Stack.
4. **pop2()**: Removes and returns the top element of the Right Stack.
5. **display()**: Shows the contents of both stacks.



no elements available ← top1 = -1 Left side

top2 = MAX = 10 Right side

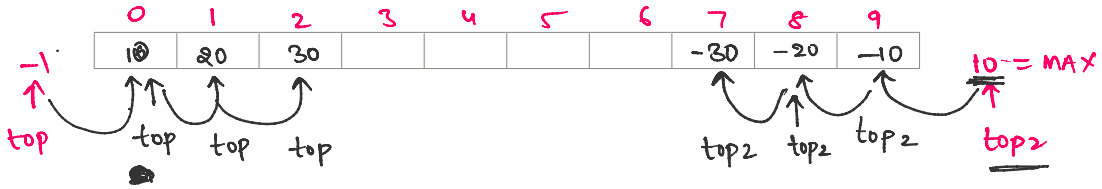
push1( ) → insert at left side

pop1( ) → remove at Left

push2( ) → insert at right side

pop2( ) → remove at right side

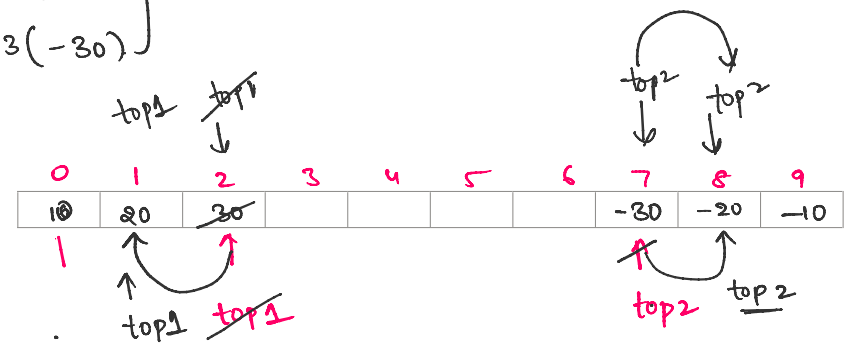
display( ) → Left & right side stack elements ← display1( ) ← display2( )



push1(10)
push1(20)
push1(30)
→ top is incrementing by 1 value

push2(-10)
push2(-20)
→ top is decrement → 1 value

push 2 (-10)
push 2 (-20)
push 3 (-30)
} → top is decrement → 1 value



pop 1( ) → 30 → deleted : here top is decrement by 1 position

pop 2( ) → -30 → deleted : here top is increment by 1 position

Requirements:

MAX = 10

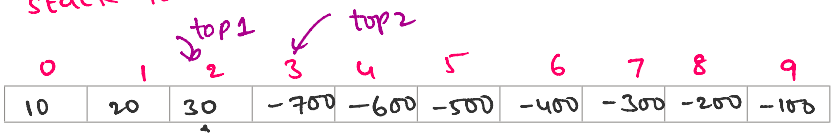
name of stack : stack[MAX] → integer

2 → tops :- top 1 = -1 (left side)

top 2 = MAX (Right side)

push 1( ) :- left side :

Case 1 :- stack is full → over flow



top 1 + 1 = = top 2

2 + 1 = = 3

3 3 → full

full ← { top 1 = 2, top 2 = 3 } top 1 = = top 2 - 1

2 = = 3 - 1

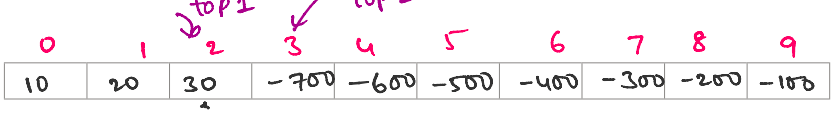
2 = = 2 → full

Case 2 :- It is not full

```
void push1(int *item)
{
    if (top1 + 1 == top2)
    {
        printf("stack is full -> overflow");
    }
    else
    {
        top1 = top1 + 1;
        stack[top1] = *item;
        printf("Element inserted/pushed successfully");
    }
}
```

push2( ) → Right side :

Case 1:- stack is full → over flow



top1 + 1 = = top2

2 + 1 == 3 → full

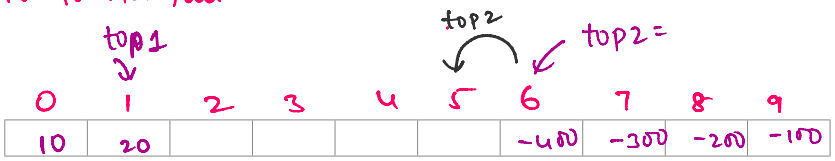
3 3

full ← top1 = 2, top2 = 3

top1 = top2 - 1

2 == 3 - 1 → full

Case 2:- It is not full



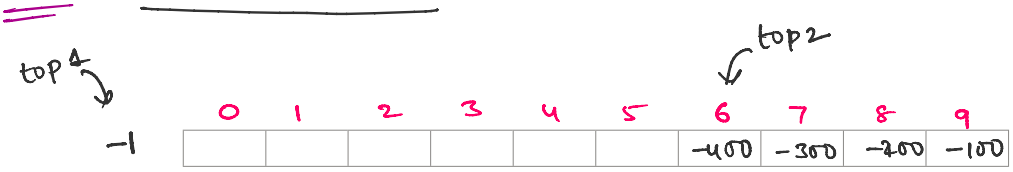

```
void push2(int *item)   → Right side
{
    if (top2 == top1 + 1)
```

```
{
    if ( top1 == top2+1)
    {
        printf(" Stack is full → overflow");
    }
    else
    {
        top2 = top2 -1 ;
      Stack[top2] = item;
      printf(" Stack element inserted/pushed successfully");
    }
}
```

---

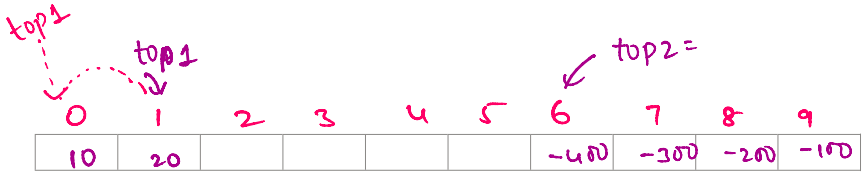
pop1():- delete an element at left side stack → top1 element

Case 1: stack is empty → at left side



top = = -1 → empty condition

Case 2: stack has elements → at left side.

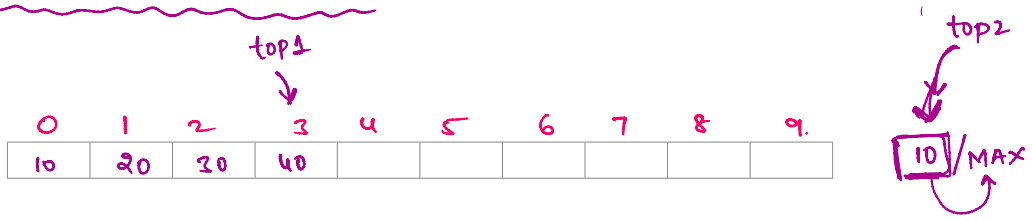



```
void pop1( )
{
```

```
void pop1()
{
    if(top == -1)
    {
        printf("stack is empty → Underflow");
    }
    else
    {
        printf("deleted element is %d", stack[top1]);
        top1 = top1 - 1;
    }
}
```

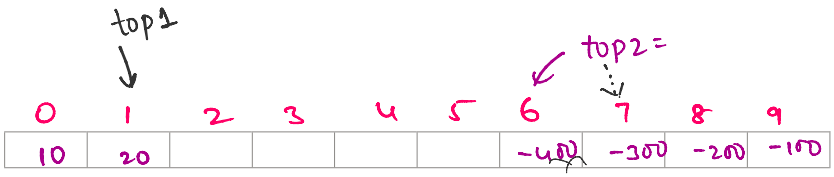
pop2() → delete an element at Right side →

Case 1:- Right side stack is empty



Condition:- top2 == MAX

Case 2:- Right side stack has elements



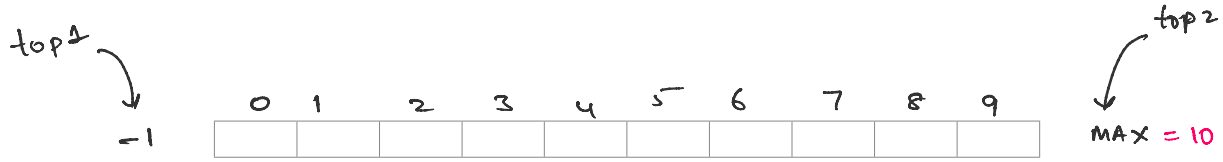

```
Void pop2() → Right side delete
{
    if( top2 == MAX)
    {
```

```
        if( top2 == MAX)
        {
            printf("stack is empty → overflow");
        }
        else
        {
            printf("deleted element is %d", stack[top2]);
            top2= top2 +1;
        }
    }
```

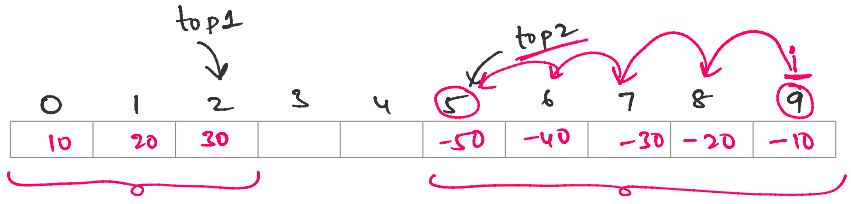
---

display( ) → print both side stacks

Case 1:. Stack is empty → both side



Case 2:. Stack is not empty → both side.




```
void display( )
{
    // Lef side
    if ( top1 == -1)
    {
        printf("Lef side stack -Empty");
    }
    .
```

```
    }
    else
    {
        printf("Left side stack elements are");
        for(i=0; i<=top1; i++)
        {
            printf(" %d", stack[i]);
        }
    }

    // Right side.
    if(top2 == MAX)
    {
        printf("Right Side stack is Empty");
    }
    else
    {
        for(i=MAX-1; i>=top2; i--)
        {
            printf(" %d", stack[i]);
        }
    }
}
```