Let us consider a stack of integers

#define SIZE 5 # Stack can hold max 5 elements

struct stack

{

int data[SIZE]; // to store the elements

int top; // to determine the position of topmost element in stack

};

The above is only a definition of stack with no memory allocated to it. To allocate memory, we need to create a variable of structure stack

struct stack s; // once structure is created, we can create any no of stacks. struct stack s1, s2, s3, …;

We can assign any name to the structure. However as the behaviour of this structure will be similar to a physical stack, we will keep the name “stack”.

s.top = -1 (If var of structure is used, refer to the member of structure using . (dot) syntax)

s

|  |
| --- |
|  |
|  |
| 30 |
| 20 |
| 10 |

4

3

top=2 data[5]

top=1

top =0

top = -1 (This indicates that the stack is empty/initialised). While inserting the elements i.e. push operation, we increment top first & then add the element at that position. When top reaches SIZE-1 position i.e. 4 in this case, then it indicates “stack full” condition.

After pop operation, top will decrement & finally top will reach -1 which indicates “stack empty” condition.

CALL BY VALUE 🡪 push(s,10) CALL BY REFERENCE 🡪 push(&s,10)

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

s s

|  |
| --- |
|  |
|  |
|  |
|  |
| 10 |

main() data[ ]

top = 0

600 top = -1 address of s 🡺 600 top = -1

s (copy of stack s from main)

ptr\_s (pointer to stack s of main())

600

|  |
| --- |
|  |
|  |
|  |
|  |
| 10 |

push()

ptr\_s->top++;

ptr\_s->dat>top] = 10;

top = 0

750

Changes made in stack by push() should be reflected

in stack s of main()