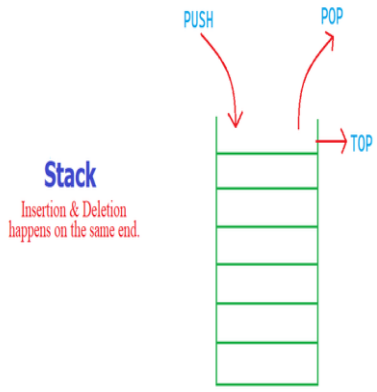
Stack Data Structure

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be **LIFO(Last In First Out) or FILO(First In Last Out).**



LIFO means, jo element sabse last me andar gya vo sabse phele bahar aayega.

FILO means, jo element sabse phele andar gya vo sabse baad(last) me bahar aayega.

Both are same.

Some of its main operations are:

If we try to pop from an empty stack then it is known as underflow and if we try to push an element in a stack that is already full, then it is known as overflow.

1. **push() :** push/insert element in stack.
2. **pop() :** remove element from the stack.
3. **Peek or top() :** Returns the top element of the stack.
4. **isEmpty() :** Returns true if the stack is empty, else false.
5. **size() :** size() function is used to return the size of the stack container or the number of elements in the stack container.

Time Complexities of operations on the stack:

push(), pop(), isEmpty() and peek() all take **O(1)** time. We do not run any loop in any of these operations.

Types of Stacks:

1. **Register Stack:** This type of stack is also a memory element present in the memory unit and can handle a small amount of data only. The height of the register stack is always limited as the size of the register stack is very small compared to the memory.
2. **Memory Stack:** This type of stack can handle a large amount of memory data. The height of the memory stack is flexible as it occupies a large amount of memory data.

Stack in C++ STL:

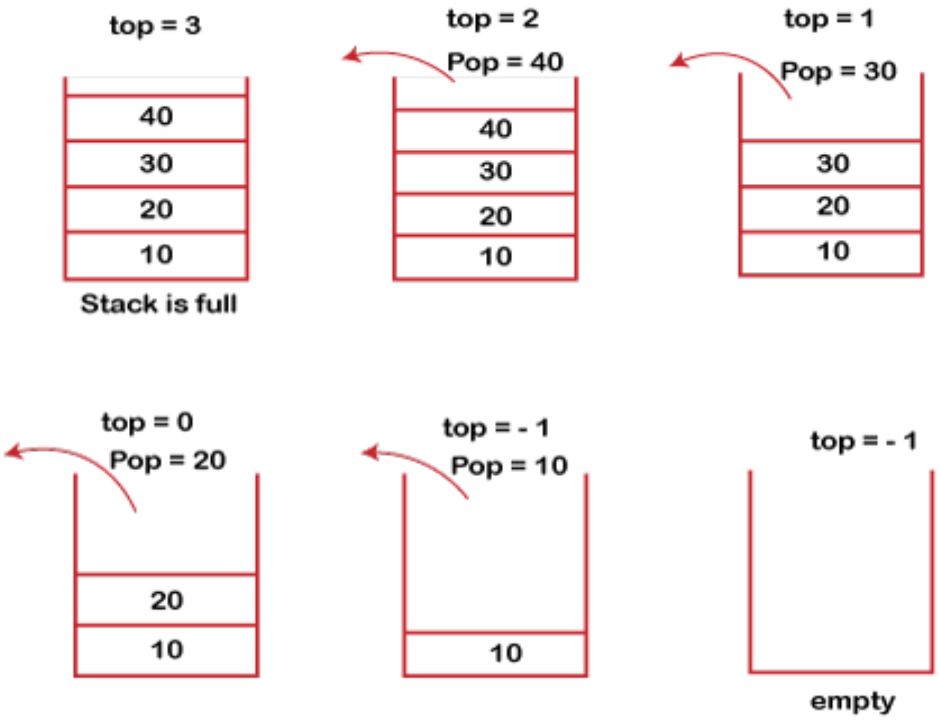
The functions associated with stack are:

1. **empty()** – Returns whether the stack is empty – Time Complexity : O(1)
2. **size()** – Returns the size of the stack – Time Complexity : O(1)
3. **top()** – Returns a reference to the top most element of the stack – Time Complexity : O(1)
4. **push(g)** – Adds the element ‘g’ at the top of the stack – Time Complexity : O(1)
5. **pop()** – Deletes the top most element of the stack – Time Complexity : O(1)

Implementation: There are two ways to implement a stack:

1. Using array/vector
2. Using linked list

In stack you can perform operation at one end i.e., from top.



In above example, elements are inserted in order, first is 10, then 20, then 30, and 40.

But when we are printing elements it came out to be, first is 40, then 30, then 20, and last is 10.