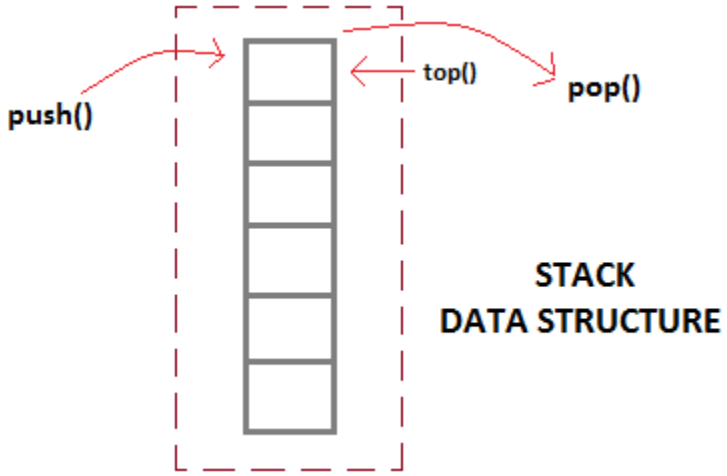
What is Stack Data Structure?

Stack is an abstract data type with a bounded(predefined) capacity. It is a simple data structure that allows adding and removing elements in a particular order. Every time an element is added, it goes on the top of the stack and the only element that can be removed is the element that is at the top of the stack, just like a pile of objects.



Basic features of Stack

1. Stack is an ordered list of similar data types.
2. Stack is a LIFO(Last in First out) structure or we can say FILO(First in Last out).
3. push() function is used to insert new elements into the Stack and pop() function is used to remove an element from the stack. Both insertion and removal are allowed at only one end of Stack called Top.
4. Stack is said to be in Overflow state when it is completely full and is said to be in Underflow state if it is completely empty.

Applications of Stack

The simplest application of a stack is to reverse a word. You push a given word to stack - letter by letter - and then pop letters from the stack.

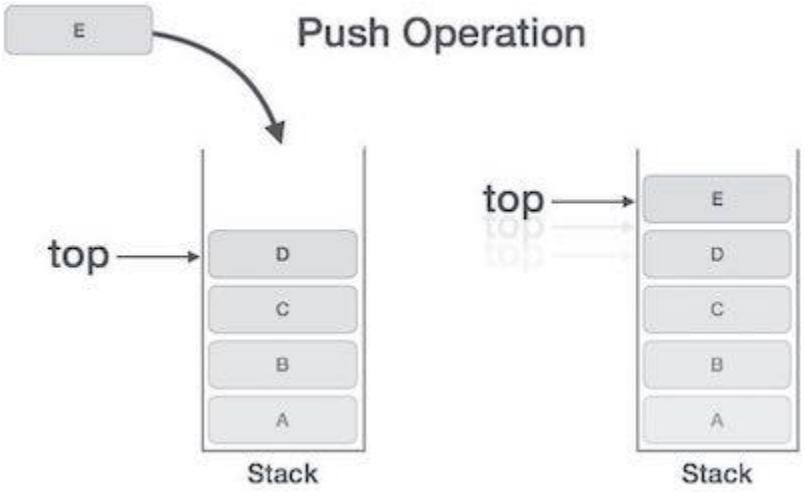
There are other uses also like:

1. Parsing
2. Expression Conversion(Infix to Postfix, Postfix to Prefix, etc)

Push Operation

The process of putting a new data element onto the stack is known as a Push Operation. Push operation involves a series of steps −

- Step 1 − Checks if the stack is full.
- Step 2 − If the stack is full, produces an error and exit.
- Step 3 − If the stack is not full, increments top to point next empty space.
- Step 4 − Adds data element to the stack location, where the top is pointing.
- Step 5 − Returns success.



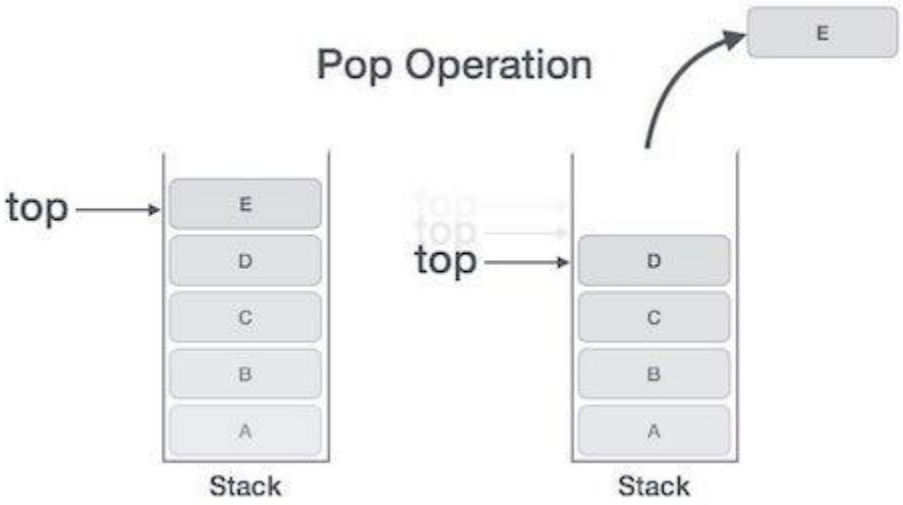
If the linked list is used to implement the stack, then in step 3, we need to allocate space dynamically.

Pop Operation

Accessing the content while removing it from the stack, is known as a Pop Operation. In an array implementation of pop() operation, the data element is not actually removed, instead the top is decremented to a lower position in the stack to point to the next value. But in linked-list implementation, pop() actually removes data elements and deallocates memory space.

A Pop operation may involve the following steps −

- Step 1 − Checks if the stack is empty.
- Step 2 − If the stack is empty, produces an error and exit.
- Step 3 − If the stack is not empty, accesses the data element at which the top is pointing.
- Step 4 − Decreases the value of the top by 1.
- Step 5 − Returns success.



Position of Top	Status of Stack
-1	Stack is Empty
0	Only one element in Stack
N-1	Stack is Full
N	Overflow state of Stack

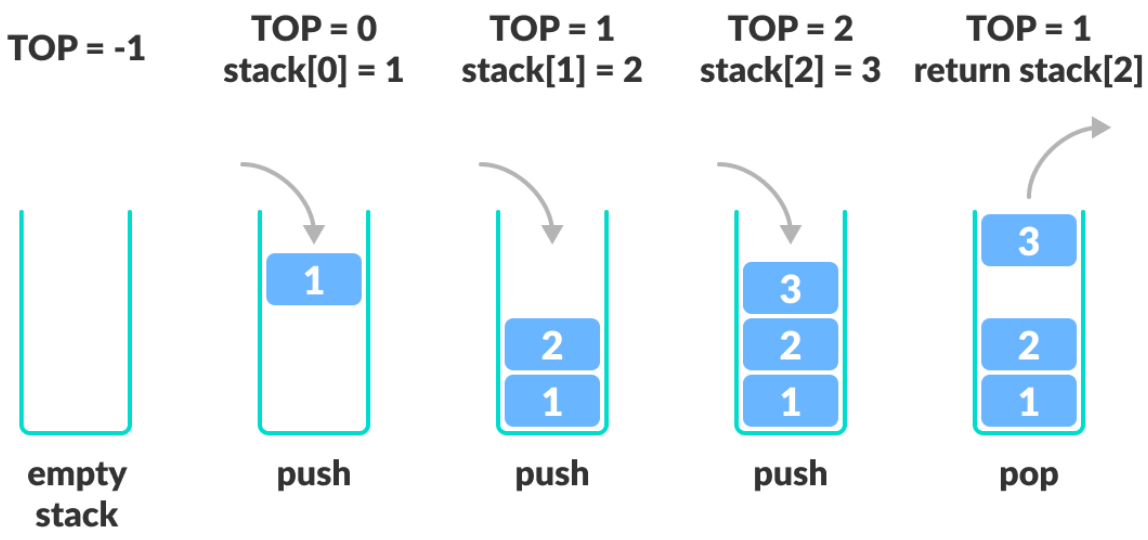
Analysis of Stack Operations

Below mentioned are the time complexities for various operations that can be performed on the Stack data structure.

- Push Operation : O(1)
- Pop Operation : O(1)
- Top Operation : O(1)
- Search Operation : O(n)

The time complexities for push() and pop() functions are O(1) because we always have to insert or remove the data from the top of the stack, which is a one-step process.

Standard Stack Operations



The following are some common operations implemented on the stack:

- **push():** When we insert an element in a stack then the operation is known as a push. If the stack is full then the overflow condition occurs.
- **pop():** When we delete an element from the stack, the operation is known as a pop. If the stack is empty means that no element exists in the stack, this state is known as an underflow state.
- **isEmpty():** It determines whether the stack is empty or not.
- **isFull():** It determines whether the stack is full or not.'
- **peek():** It returns the element at the given position.
- **count():** It returns the total number of elements available in a stack.
- **change():** It changes the element at the given position.
- **display():** It prints all the elements available in the stack.