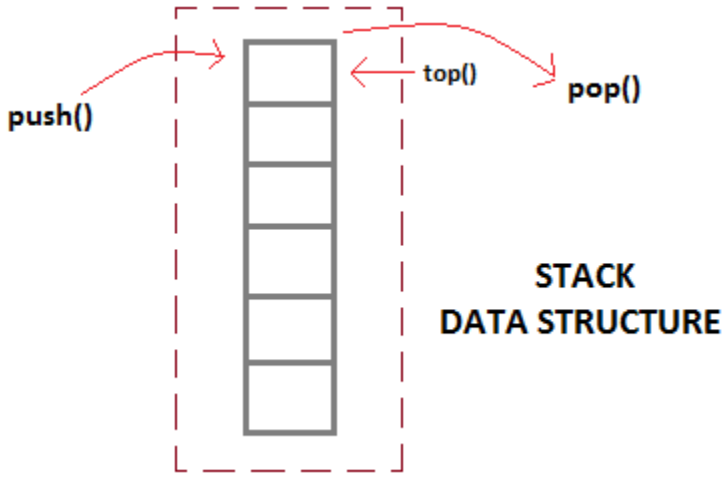
What is Stack Data Structure?

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 type**.
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

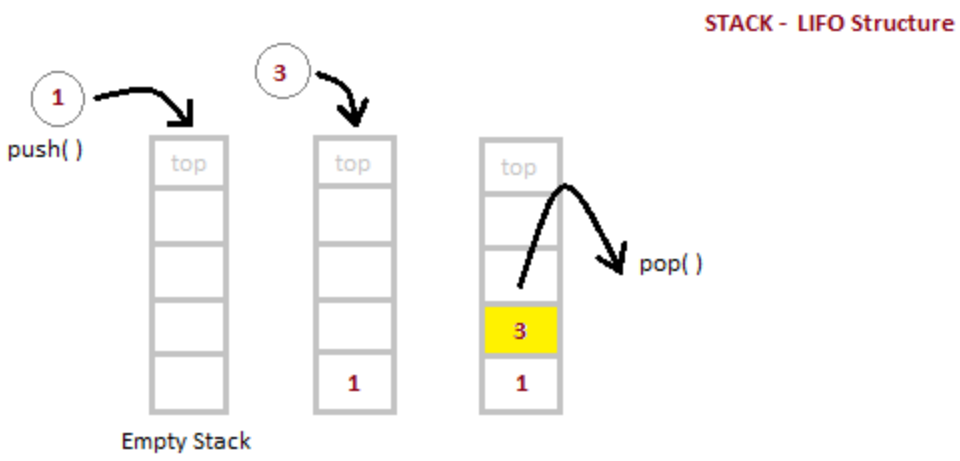
The simplest application of a stack is to reverse a word. You push a given word to stack - letter by letter - and then pop letters from the stack.

There are other uses also like:

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

Implementation of Stack Data Structure

Stack can be easily implemented using an Array or a Linked List. Arrays are quick, but are limited in size and Linked List requires overhead to allocate, link, unlink, and deallocate, but is not limited in size. Here we will implement Stack using array.



In a Stack, all operations take place at the "**top**" of the stack. The "**push**" operation adds an item to the top of the Stack.
The "**pop**" operation removes the item on top of the stack.

Algorithm for PUSH operation

1. Check if the stack is **full** or not.
2. If the stack is full, then print error of overflow and exit the program.
3. If the stack is not full, then increment the top and add the element.

Algorithm for POP operation

1. Check if the stack is empty or not.
2. If the stack is empty, then print error of underflow and exit the program.
3. If the stack is not empty, then print the element at the top and decrement the top.

Below we have a simple C++ program implementing stack data structure while following the object oriented programming concepts.

If you are not familiar with C++ programming concepts, you can learn it form here.

```
/*  Below program is written in C++ language  */


# include<iostream>


using namespace std;


class Stack
{
    int top;
    public:
    int a[10];  //Maximum size of Stack
    Stack()
```

```
    {
        top = -1;
    }

    // declaring all the function
    void push(int x);
    int pop();
    void isEmpty();
};

// function to insert data into stack
void Stack::push(int x)
{
    if(top >= 10)
    {
        cout << "Stack Overflow \n";
    }
    else
    {
        a[++top] = x;
        cout << "Element Inserted \n";
    }
}

// function to remove data from the top of the stack
int Stack::pop()
{
    if(top < 0)
    {
        cout << "Stack Underflow \n";
```

```
        return 0;

    }

    else

    {

        int d = a[top--];

        return d;

    }

}


// function to check if stack is empty

void Stack::isEmpty()

{

    if(top < 0)

    {

        cout << "Stack is empty \n";

    }

    else

    {

        cout << "Stack is not empty \n";

    }

}


// main function

int main() {


    Stack s1;

    s1.push(10);

    s1.push(100);

    /*

        preform whatever operation you want on the stack
```

```
    */

}
```

Copy

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.

Now that we have learned about the Stack in Data Structure, you can also check out these topics:

- Queue Data Structure
- Queue using stack
