# STACKS

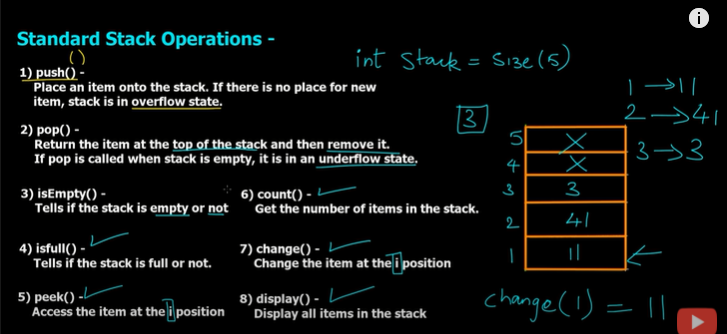
# Definition:

Stack is a linear data structure that works in last in first out LIFO or first in last out FILO pattern. It is an abstract data type with bounded (pre-defined) capacity. It is a simple data structure that allows adding and removing elements in particular order.

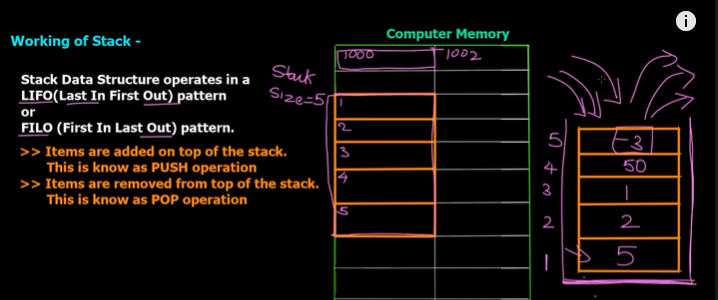
A real life example of stack data structure is a pile of books. You collect some books and stack them on top of another. The book on the bottom which was firstly added to the stack now can be obtained at the last. (First in last out). You can only add a new book on the top most of the stack and that book will be accessible first among all the books in that stack.

## Mainly the following three basic operations are performed in the stack:

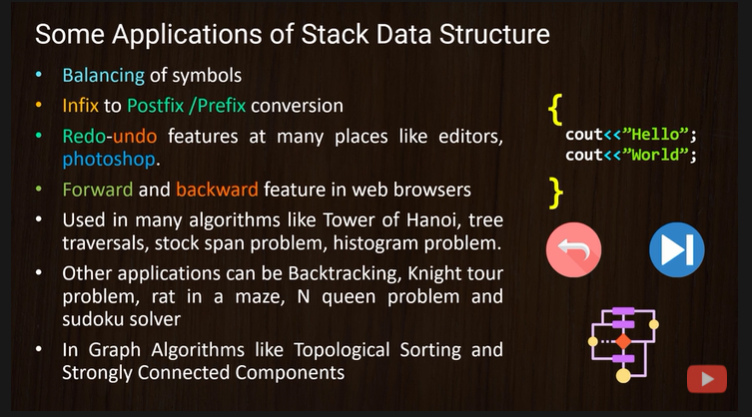
1. **Push:**Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
2. **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
3. **Peek or Top:** Returns the top element of the stack.
4. **isEmpty:**Returns true if the stack is empty, else false.
5. **isFull:** Returns true if the stack is full, else false.
6. **Count:** Get the number of items in the stack.
7. **Change:** Change the item at the ith position.
8. **Display:** Display all the items in the stack.



Working of stack data structure



Applications of stack:



# Code:

#include <iostream>

using namespace std;

class Stack

{

    int top;

    int arr[5];

public:

    Stack()

    {

        top = -1;

        for (int i = 0; i < 5; i++)

        {

            arr[i] = 0;

        }

    }

    // 1. isEmpty function

    bool isEmpty()

    {

        if (top == -1)

        {

            return true;

        }

        else

        {

            return false;

        }

    }

    // 2. isFull function

    bool isFull()

    {

        if (top == 4)

        {

            return true;

        }

        else

        {

            return false;

        }

    }

    // 3. push function

    void push(int val)

    {

        if (isFull())

        {

            cout << "Stack Overflow \n";

        }

        else

        {

            top++;

            arr[top] = val;

        }

    }

    // 4. pop function

    int pop()

    {

        if (isEmpty())

        {

            cout << "Stack underflow. \n";

            return 0;

        }

        else

        {

            int popval = arr[top];

            arr[top] = 0;

            top--;

            return popval;

        }

    }

    // 5. count function

    int count()

    {

        return (top + 1);

    }

    // 6. peek function

    int peek(int pos)

    {

        if (isEmpty())

        {

            cout << "Stack underflow. \n";

            return 0;

        }

        else

        {

            return arr[pos];

        }

    }

    // 7. change function

    void change(int pos, int val)

    {

        arr[pos] = val;

        cout << "Value changed at location " << pos << "\n";

    }

    // 8. display function

    void display()

    {

        cout << "Printing all the items in the stack : \n";

        for (int i = 4; i >= 0; i--)

        {

            cout << arr[i] << endl;

        }

    }

};

int main()

{

    Stack s1;

    int option, position, value;

    do

    {

        cout << "What operation do you want to perform? Select option number. Enter 0 to quit\n";

        cout << "1. push()" << endl;

        cout << "2. pop()" << endl;

        cout << "3. isEmpty()" << endl;

        cout << "4.isFull()" << endl;

        cout << "5. peek()" << endl;

        cout << "6. count()" << endl;

        cout << "7. change()" << endl;

        cout << "8. display" << endl;

        cout << "9. clear" << endl;

        cout << "Enter your choice :";

        cin >> option;

        switch (option)

        {

        case 1:

            cout << "Enter an element to push into the stack : ";

            cin >> value;

            s1.push(value);

            break;

        case 2:

            cout << "pop function called - poped value is " << s1.pop() << endl;

            break;

        case 3:

            (s1.isEmpty()) ? cout << "Stack is empty." << endl : cout << "stack is not empty" << endl;

            break;

        case 4:

            (s1.isFull()) ? cout << "Stack is full." << endl : cout << "stack is not full" << endl;

            break;

        case 5:

            cout << "Enter the position of the item you want to peek : ";

            cin >> position;

            cout << "peek function called. Item at " << position << " is " << s1.peek(position) << endl;

            break;

        case 6:

            cout << "count function called. Count is " << s1.count() << endl;

            break;

        case 7:

            cout << "change function called." << endl;

            cout << "Enter the position and new value you want to set : ";

            cin >> position >> value;

            s1.change(position, value);

            cout << "Item changed at position" << position << ". New value is " << value << endl;

            break;

        case 8:

            cout << "display function called. Displaying all items in the stack. \n";

            s1.display();

            break;

        case 9:

            system("cls");

            break;

        default:

            cout << "Enter a valid choice!\n";

            break;

        }

    } while (option != 0);

    return 0;

}