Tutorial Link https://codequotient.com/tutorials/Stack - Introduction/5a12f3ec46765b2b63e347fe

TUTORIAL

Stack - Introduction

Chapter

Stack is very useful concept in Computer Science. Stack is a linear data structure which allows elements to be inserted as well as deleted only from one end. Stack is also known as LIFO data structure. Everyday examples of such a structure are very common viz. a stack of dishes, a stack of books, a stack of coins and a stack of cloths, etc. as shown in figure below:

A Stack of plates

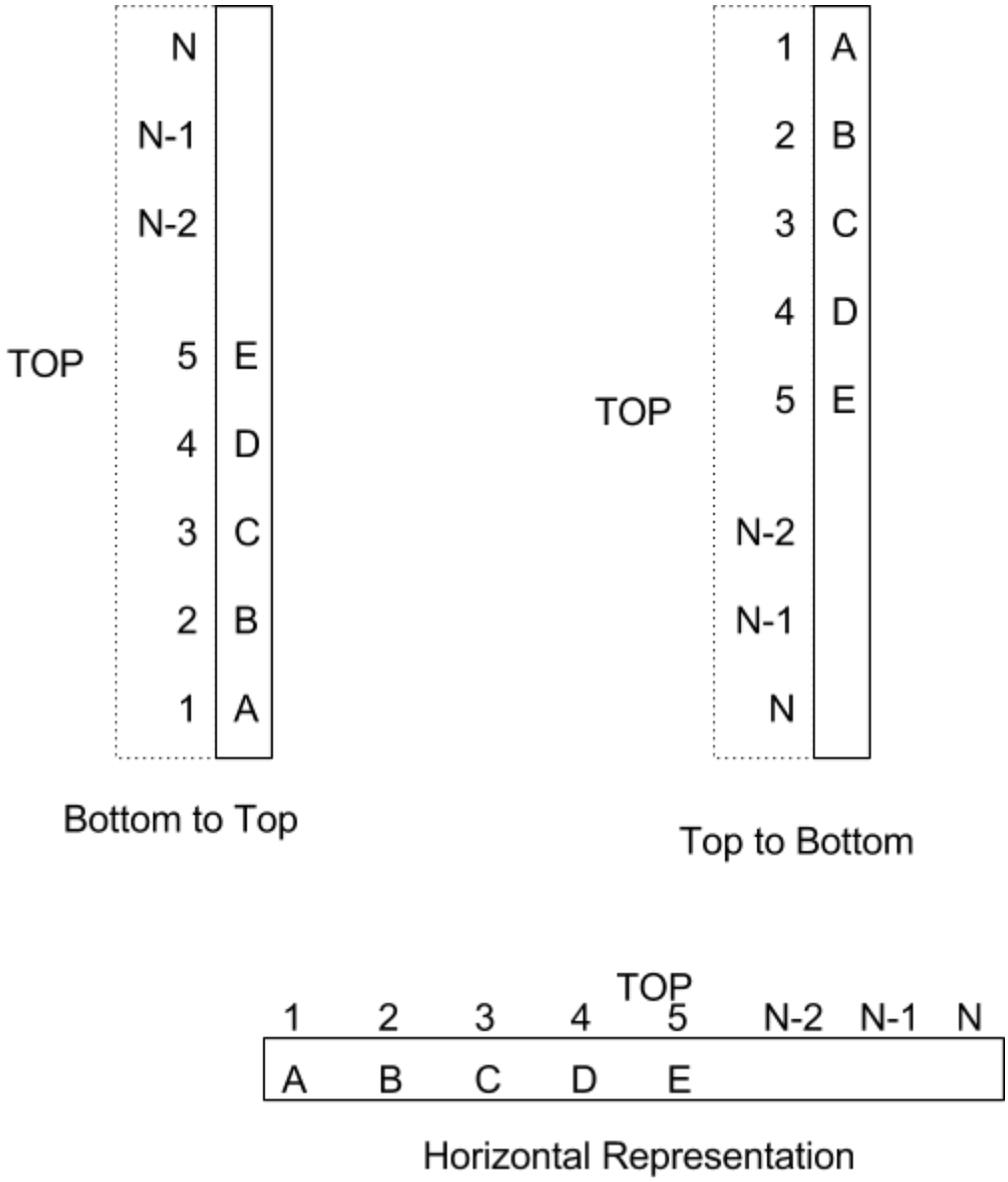
Stacks are also called last-in first-out (LIFO) lists. This means, that elements which are inserted last will be removed first. Other names generally used for stacks are "piles" and "push-down lists". Stack has many important applications in the field of computer science. Special terminology is used for two basic operation associated with stacks:

(a) "Push" is the term used to insert an element into a stack in O(1) time.

(b) "Pop" is the term used to delete an element from a stack in O(1) time.

Example: Suppose that 5 elements are pushed onto an empty stack A, B, C, D, E

Figure below shows three ways of picturing such a stack from top to bottom as well as from bottom to top.



A Stack of 5 elements

Stack analogy is the tasks you perform during a typical workday. You're busy on a long-term project (A), but you're interrupted by a coworker asking you for temporary help with another project (B). While you're working on B, someone in accounting stops by for a meeting about travel expenses (C), and during this meeting you get an emergency call from someone in sales and spend a few minutes troubleshooting a bulky product (D). When you're done with call D, you resume meeting C; when you're done with C, you resume project B, and when you're done with B you can (finally!) get back to project A. Lower priority projects are "stacked up" waiting for you to return to them.

Placing a data item on the top of the stack is called pushing it. Removing it from the top of the stack is called popping it. These are the primary stack operations. A stack is said to be a Last-In-First-Out (LIFO) storage mechanism, because the last item inserted is the first one to be removed.

Computer Representation of Stack

There are two ways to represent stack in computers by Array or by Linked List

a. Implementation of Stacks using Arrays

Insertion: When we are adding a new element, first, we must test whether there is a free space in the stack for the new item; if not, then we have the condition known as overflow. If this condition is not there, then the value of TOP is changed before the insertion in PUSH. After changing the value of TOP, insertion is done.

Algorithm : PUSH (STACK, ITEM)

```
If TOP=MAXSTK, then Write OVERFLOW and Exit
  TOP = TOP + 1
  STACK [TOP] = ITEM
Exit
```

Deletion: In executing the procedure POP, we must first test whether there is an element in the stack to be deleted; if not; then we have the condition known as underflow. The item to be deleted is first stored in some variable, then the value of TOP is changed after the deletion in POP.

Algorithm: POP (STACK, ITEM)

```
If TOP = 0, then Write UNDERFLOW and Exit
ITEM = STACK[TOP]
TOP = TOP-1
Return Item
Exit
```

A Stack contains an ordered list of elements and an array is also used to store ordered list of elements. Hence, it would be very easy to manage a stack using an array. However, the problem with an array is that we are required to declare the size of the array before using it in a program. Therefore, the size of stack would be fixed.

Though an array and a stack are totally different data structures, an array can be used to store the elements of a stack. We can declare the array with a maximum size large enough to manage a stack. Following is the implementation in different languages:

Javascript

```
class CQStack{
    // Initialize the stack object
    constructor(capacity){
        this.MAX_CAPACITY = capacity
        this.stack = new Array(capacity).fill(0)
        this.top = -1
    }

    // Add data to the stack
    push(data){
        if(this.isFull()){
```

```
            console.log('Overflow')
        }else{
            console.log(`${data} pushed to stack`)
            this.stack[++(this.top)] = data
            console.log(`Top is now at ${this.top}`)
        }
    }

    // Remove the data from stack
    pop(){
        if(this.isEmpty()){
            console.log('Underflow')
        }else{
            let temp = this.stack[this.top--]
            console.log(`${temp} popped from stack`)
            console.log(`Top is now at ${this.top}`)
            return temp
        }
    }

    // Returns true if the stack is empty
    isEmpty(){
        return this.top === -1
    }

    // Returns true if the stack is full
    isFull(){
        return this.top === this.MAX_CAPACITY
    }

}

function main(){
    const theStack = new CQStack(100); // make new stack
    theStack.push(12);
    theStack.push(23);
    let temp = theStack.pop();
    theStack.push(54);
    temp=theStack.pop();
    temp=theStack.pop();
```

```
    temp=theStack.pop();
}

main()
```

C

```c
#include<stdio.h>

#define SIZE 10

int Stack[SIZE], top=-1;

int isFull()
{
  return top==(SIZE-1);
}

int isEmpty()
{
  return top==-1;
}

// Function to add an item to stack.  It increases top by 1
int push(int item)
{
  if (isFull())
  {
    printf("OVERFLOW");
    return -1;
  }
  printf("%d pushed to stack\n",item);
  Stack[++top] = item;
  printf("Top is now at %d\n", top);
}

// Function to remove an item from stack.  It decreases top by 1
int pop()
{
  int temp;
  if (isEmpty())
```

```
  {
    printf("UNDERFLOW \n");
    return -1;
  }
  temp=Stack[top--];
  printf("%d popped from stack\n", temp);
  printf("Top is now at %d\n", top);
  return temp;
}

int main()
{
  int temp;
  push(12);
  push(23);
  temp=pop();
  push(54);
  temp=pop();
  temp=pop();
  temp=pop();
  return 0;
}

```

Java

```
class CQStack
{
  private int maxSize; // size of stack array
  private int[] stackArray;
  private int top; // top of stack

  public CQStack(int s) // constructor
  {
    maxSize = s; // set array size
    stackArray = new int[maxSize]; // create array
    top = -1; // no items yet
  }
  public void push(int j) // put item on top of stack
  {
    if(isFull())
    {
```

```
      System.out.print("OVERFLOW");
    }
    else
    {
      System.out.println(j + " pushed to stack");
      stackArray[++top] = j; // increment top, insert item
      System.out.println("Top is now at " + top);
    }
  }
  public int pop() // take item from top of stack
  {
    if (isEmpty())
    {
      System.out.println("UNDERFLOW");
      return -1;
    }
    else
    {
      int temp=stackArray[top--];
      System.out.println(temp + " popped from stack");
      System.out.println("Top is now at " + top);
      return temp; // access item, decrement top
    }
  }
  public boolean isEmpty() // true if stack is empty
  {
    return (top == -1);
  }
  public boolean isFull() // true if stack is full
  {
    return (top == maxSize-1);
  }
}

class Main
{
  public static void main(String[] args)
  {
    CQStack theStack = new CQStack(10); // make new stack
```

```
    int temp;
    theStack.push(12);
    theStack.push(23);
    temp = theStack.pop();
    theStack.push(54);
    temp=theStack.pop();
    temp=theStack.pop();
    temp=theStack.pop();
  }
}
```

Python 3

```
class CQStack:
    def __init__(self,capacity):
        self.stack = [0]*capacity
        self.MAX_CAPACITY = capacity
        self.top = -1

    def isEmpty(self):
        return self.top == -1

    def isFull(self):
        return self.top == self.MAX_CAPACITY

    def push(self,data):
        if(self.isFull()):
            print('Overflow')
        else:
            self.top+=1
            self.stack[self.top] = data
            print(data,'pushed to stack')
            print('Top is now at',self.top)

    def pop(self):
        if(self.isEmpty()):
            print('Underflow')
            return -1
        else:
            temp = self.stack[self.top]
            self.top-=1
            print(temp,'popped from stack')
```

```
            print('Top is now at',self.top)
            return temp


if __name__ == '__main__':
    theStack = CQStack(100);
    theStack.push(12);
    theStack.push(23);
    temp = theStack.pop();
    theStack.push(54);
    temp=theStack.pop();
    temp=theStack.pop();
    temp=theStack.pop();
```

C++

```
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;


class CQStack{
    int *stack;
    int top;
    int size;
    int MAX_CAPACITY;
    public:
        // Constructor for initializing the stack object
        CQStack(int capacity){
            MAX_CAPACITY = capacity;
            stack = new int[MAX_CAPACITY];
            size = 0;
            top = -1;
        }

                // Pushing data into stack
        void push(int data){
            if(isFull()){
                cout<<"Overflow";
            }else{
                cout<<data<<" pushed to stack"<<endl;
```

```
                stack[++top] = data;
                cout<<"Top is now at "<<top<<endl;
                size++;
            }
        }

                // Removing the top element from stack
        int pop(){
            if(isEmpty()){
                cout<<"Underflow";
                return -1;
            }else{
                int temp = stack[top--];
                cout<<temp<<" popped from stack"<<endl;
                cout<<"Top is now at "<<top<<endl;
                size--;
                return temp;
            }
        }

                // Returns true if the stack is empty
        bool isEmpty(){
            return top == -1;
        }

                // Returns true if the stack is full
        bool isFull(){
            return top == MAX_CAPACITY;
        }

};


int main() {
    CQStack *theStack = new CQStack(100);
    int temp;
    theStack->push(12);
    theStack->push(23);
    temp = theStack->pop();
    theStack->push(54);
```

```
    temp=theStack->pop();
    temp=theStack->pop();
    temp=theStack->pop();
}
```

Video Solution

<iframe width="560" height="315" src="https://www.youtube.com/embed/0-kAfWagqlk" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>