## **Stack**

It is a linear data structure that follows a particular order in which the operations are performed.

To implement the stack, it is required to maintain the **pointer to the top of the stack**, which is the last element to be inserted because **we can access the elements only on the top of the stack.**

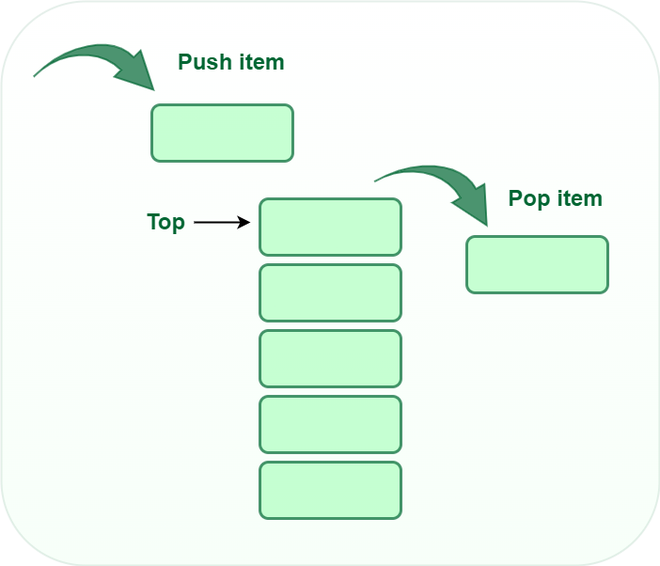
**LIFO( Last In First Out ):**

*This strategy states that the element that is inserted last will come out first. You can take a pile of plates kept on top of each other as a real-life example. The plate which we put last is on the top and since we remove the plate that is at the top, we can say that the plate that was put last comes out first.*

## **Basic Operations on Stack**

In order to make manipulations in a stack, there are certain operations provided to us.

* **push()** to insert an element into the stack
* **pop()** to remove an element from the stack
* **top()** Returns the top element of the stack.
* **isEmpty()** returns true if stack is empty else false.
* **size()** returns the size of stack.



*Stack*

## **Push:**

Adds an item to the stack. If the stack is full, then it is said to be an **Overflow condition.**

**Algorithm for push:**

begin

if stack is full

return

endif

else

increment top

stack[top] assign value

end else

end procedure

## **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.**

**Algorithm for pop:**

begin

if stack is empty

return

endif

else

store value of stack[top]

decrement top

return value

end else

end procedure

## **Top:**

Returns the top element of the stack.

**Algorithm for Top:**

begin

return stack[top]

end procedure

## **isEmpty:**

Returns true if the stack is empty, else false.

**Algorithm for isEmpty**:

begin

if top < 1

return true

else

return false

end procedure

### **Understanding stack practically:**

*There are many real-life examples of a stack. Consider the simple example of plates stacked over one another in a canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow the LIFO/FILO order.*

## **Complexity Analysis:**

* **Time Complexity**

| **Operations** | **Complexity** |
| --- | --- |
| push() | O(1) |
| pop() | O(1) |
| isEmpty() | O(1) |
| size() | O(1) |

## **Types of Stacks:**

* **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.
* **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.

## **Applications of the stack:**

* [Infix to Postfix](https://www.geeksforgeeks.org/stack-set-2-infix-to-postfix/) /Prefix conversion
* Redo-undo features at many places like editors, photoshop.
* Forward and backward features in web browsers
* Used in many algorithms like [Tower of Hanoi,](https://www.geeksforgeeks.org/recursive-functions/) [tree traversals](https://www.geeksforgeeks.org/618/), [stock span problems](https://www.geeksforgeeks.org/the-stock-span-problem/), and [histogram problems](https://www.geeksforgeeks.org/largest-rectangular-area-in-a-histogram-set-1/).
* Backtracking is one of the algorithm designing techniques. Some examples of backtracking are the Knight-Tour problem, N-Queen problem, find your way through a maze, and game-like chess or checkers in all these problems we dive into someway if that way is not efficient we come back to the previous state and go into some another path. To get back from a current state we need to store the previous state for that purpose we need a stack.
* In Graph Algorithms like [Topological Sorting](https://www.geeksforgeeks.org/topological-sorting/) and [Strongly Connected Components](https://www.geeksforgeeks.org/strongly-connected-components/)
* In Memory management, any modern computer uses a stack as the primary management for a running purpose. Each program that is running in a computer system has its own memory allocations
* String reversal is also another application of stack. Here one by one each character gets inserted into the stack. So the first character of the string is on the bottom of the stack and the last element of a string is on the top of the stack. After Performing the pop operations on the stack we get a string in reverse order.
* Stack also helps in implementing function call in computers. The last called function is always completed first.
* Stacks are also used to implement the undo/redo operation in text editor.

## **Implementation of Stack:**

There are two ways to implement a stack

* Using array
* Using linked list

## **Implementing Stack using Arrays:**

Recommended Problem

Implement Stack using Linked List

[Linked List](https://practice.geeksforgeeks.org/explore?page=1&category%5B%5D=Linked%20List&sortBy=submissions)

[Stack](https://practice.geeksforgeeks.org/explore?page=1&category%5B%5D=Stack&sortBy=submissions)

+1 more

[Codenation](https://practice.geeksforgeeks.org/explore?page=1&company%5B%5D=Codenation&sortBy=submissions)

[FactSet](https://practice.geeksforgeeks.org/explore?page=1&company%5B%5D=FactSet&sortBy=submissions)

+3 more

[Solve Problem](https://practice.geeksforgeeks.org/problems/implement-stack-using-linked-list/1?utm_source=gfg&utm_medium=article&utm_campaign=bottom_sticky_on_article)

Submission count: 86.9K

* C++
* C
* Java
* Python3
* C#
* Javascript

| /\* Java program to implement basic stack  operations \*/  class Stack {  static final int MAX = 1000;  int top;  int a[] = new int[MAX]; // Maximum size of Stack    boolean isEmpty()  {  return (top < 0);  }  Stack()  {  top = -1;  }    boolean push(int x)  {  if (top >= (MAX - 1)) {  System.out.println("Stack Overflow");  return false;  }  else {  a[++top] = x;  System.out.println(x + " pushed into stack");  return true;  }  }    int pop()  {  if (top < 0) {  System.out.println("Stack Underflow");  return 0;  }  else {  int x = a[top--];  return x;  }  }    int peek()  {  if (top < 0) {  System.out.println("Stack Underflow");  return 0;  }  else {  int x = a[top];  return x;  }  }    void print(){  for(int i = top;i>-1;i--){  System.out.print(" "+ a[i]);  }  }  }    // Driver code  class Main {  public static void main(String args[])  {  Stack s = new Stack();  s.push(10);  s.push(20);  s.push(30);  System.out.println(s.pop() + " Popped from stack");  System.out.println("Top element is :" + s.peek());  System.out.print("Elements present in stack :");  s.print();  }  } |
| --- |

**Output**

10 pushed into stack

20 pushed into stack

30 pushed into stack

30 Popped from stack

Top element is : 20

Elements present in stack : 20 10

## **Advantages of array implementation:**

* Easy to implement.
* Memory is saved as pointers are not involved.

## **Disadvantages of array implementation:**

* It is not dynamic i.e., it doesn’t grow and shrink depending on needs at runtime. [But in case of dynamic sized arrays like vector in C++, list in Python, ArrayList in Java, stacks can grow and shrink with array implementation as well].
* The total size of the stack must be defined beforehand.

## **Implementing Stack using Linked List:**

* C++
* C
* Java
* Python3
* C#
* Javascript

| // Java Code for Linked List Implementation    public class StackAsLinkedList {    StackNode root;    static class StackNode {  int data;  StackNode next;    StackNode(int data) { this.data = data; }  }    public boolean isEmpty()  {  if (root == null) {  return true;  }  else  return false;  }    public void push(int data)  {  StackNode newNode = new StackNode(data);    if (root == null) {  root = newNode;  }  else {  StackNode temp = root;  root = newNode;  newNode.next = temp;  }  System.out.println(data + " pushed to stack");  }    public int pop()  {  int popped = Integer.MIN\_VALUE;  if (root == null) {  System.out.println("Stack is Empty");  }  else {  popped = root.data;  root = root.next;  }  return popped;  }    public int peek()  {  if (root == null) {  System.out.println("Stack is empty");  return Integer.MIN\_VALUE;  }  else {  return root.data;  }  }    // Driver code  public static void main(String[] args)  {    StackAsLinkedList sll = new StackAsLinkedList();    sll.push(10);  sll.push(20);  sll.push(30);    System.out.println(sll.pop()  + " popped from stack");    System.out.println("Top element is " + sll.peek());  }  } |
| --- |

**Output**

10 pushed to stack

20 pushed to stack

30 pushed to stack

30 popped from stack

Top element is 20

Elements present in stack : 20 10

## **Advantages of Linked List implementation:**

* The linked list implementation of a stack can grow and shrink according to the needs at runtime.
* It is used in many virtual machines like JVM.

## **Disadvantages of Linked List implementation:**

* Requires extra memory due to the involvement of pointers.
* Random accessing is not possible in stack.