

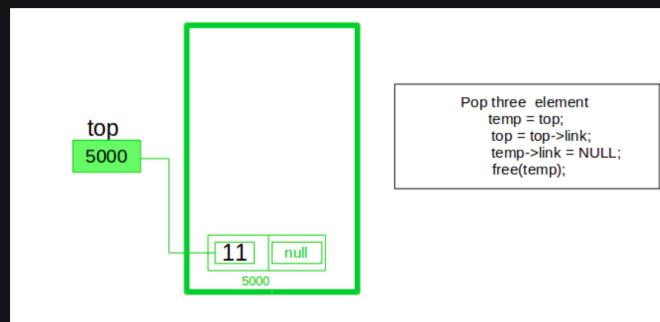
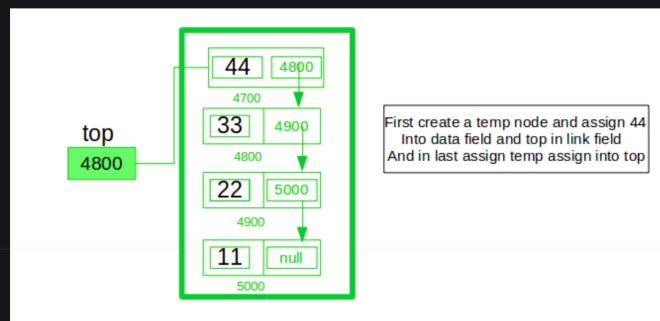
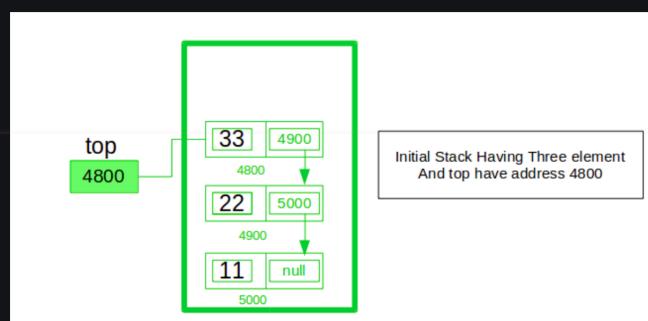
[Circular Queue | Set 2 \(Circular Linked List Implementation\)](#)[Circular Queue | Set 1 \(Introduction and Array Implementation\)](#)[Queue | Set 1 \(Introduction and Array Implementation\)](#)[Queue – Linked List Implementation](#)[Implement a stack using singly linked list](#)[Stack Data Structure \(Introduction and Program\)](#)[Finding sum of digits of a number until sum becomes single digit](#)[Program for Sum of the digits of a given number](#)[Compute sum of digits in all numbers from 1 to n](#)

Implement a stack using singly linked list

Difficulty Level : Easy • Last Updated : 24 Jun, 2022



To implement a [stack](#) using singly linked list concept , all the singly [linked list](#) operations are performed based on Stack operations LIFO(last in first out) and with the help of that knowledge we are going to implement a stack using single linked list. Using singly linked lists , we implement stack by storing the information in the form of nodes and we need to follow the stack rules and implement using singly linked list nodes . So we need to follow a simple rule in the implementation of a stack which is last in first out and all the operations can be performed with the help of a top variable .Let us learn how to perform **Pop** , **Push** , **Peek** ,**Display** operations in the following article .



WHAT'S NEW

Data Structures & Algorithms- Self Paced Course

[View Details](#)

Complete Interview Preparation- Self Paced Course

[View Details](#)

Practice Problems, POTD Streak, Weekly Contests & More!

[View Details](#)

A stack can be easily implemented using the linked list. In stack implementation, a stack contains a top pointer, which is "head" of the stack where pushing and popping items happens at the head of the list. First node have null in link field and second node link have first node address in link field and so on and last node address in "top" pointer.

The main advantage of using linked list over an arrays is that it is possible to implement a stack that can shrink or grow as much as needed. In using array will put a restriction to the maximum capacity of the array which can lead to stack overflow. Here each new node will be dynamically allocate. so overflow is not possible.

Stack Operations:

1. [**push\(\)**](#) : Insert a new element into stack i.e just inserting a new element at the beginning of the linked list.
2. [**pop\(\)**](#) : Return top element of the Stack i.e simply deleting the first element from the linked list.
3. [**peek\(\)**](#) : Return the top element.

4. **display()**: Print all elements in Stack.



Complete Interview Preparation - Self Paced

By Sandeep Jain Beginner to Advance Level ★★★★☆

Find 360 solution to all of your interview woes. Learn 4 years' worth of programming knowledge in just 6 months and ace coding interviews at top tech companies.

[Explore Now](#)

Below is the implementation of the above approach:

C++ Java Python3 C# Javascript

```
'''Python supports automatic garbage collection so deallocation of memory is done implicitly. However to force it to deallocate each node after usage add the following code:  
  
import gc #added at the start of program  
gc.collect() #to be added wherever memory is to be deallocated  
...  
  
class Node:  
  
    # Class to create nodes of linked list  
    # constructor initializes node automatically  
    def __init__(self,data):  
        self.data = data  
        self.next = None  
  
class Stack:  
  
    # head is default NULL  
    def __init__(self):  
        self.head = None  
  
    # Checks if stack is empty  
    def isempty(self):  
        if self.head == None:  
            return True  
        else:  
            return False  
  
    # Method to add data to the stack  
    # adds to the start of the stack  
    def push(self,data):  
  
        if self.head == None:  
            self.head=Node(data)  
  
        else:  
            newnode = Node(data)  
            newnode.next = self.head  
            self.head = newnode  
  
    # Remove element that is the current head (start of the stack)  
    def pop(self):  
  
        if self.isempty():  
            return None  
  
        else:  
            # Removes the head node and makes  
            #the preceding one the new head  
            poppednode = self.head  
            self.head = self.head.next  
            poppednode.next = None  
            return poppednode.data  
  
    # Returns the head node data  
    def peek(self):  
  
        if self.isempty():  
            return None  
  
        else:  
            return self.head.data  
  
    # Prints out the stack  
    def display(self):  
  
        internode = self.head  
        if self.isempty():  
            print("Stack Underflow")  
  
        else:
```

AD



RStudio Academy helps your team to become more data-literate.
rstudio.com/academy

Learn Data Science with RStudio Academy

What if everyone in your company could learn how to do data science with R and Python? And build skills to tackle specific challenges they face on the job? With RStudio Academy, it's possible. Your colleagues will work through a curriculum...

RStudio PBC

[Learn more](#)

```

        while(iernode != None):
            print(iernode.data,"->",end = " ")
            iernode = iernode.next
        return

    # Driver code
    MyStack = Stack()

    MyStack.push(11)
    MyStack.push(22)
    MyStack.push(33)
    MyStack.push(44)

    # Display stack elements
    MyStack.display()

    # Print top element of stack
    print("\nTop element is ",MyStack.peek())

    # Delete top elements of stack
    MyStack.pop()
    MyStack.pop()

    # Display stack elements
    MyStack.display()

    # Print top element of stack
    print("\nTop element is ", MyStack.peek())

    # This code is contributed by Matheus George

```

Output:

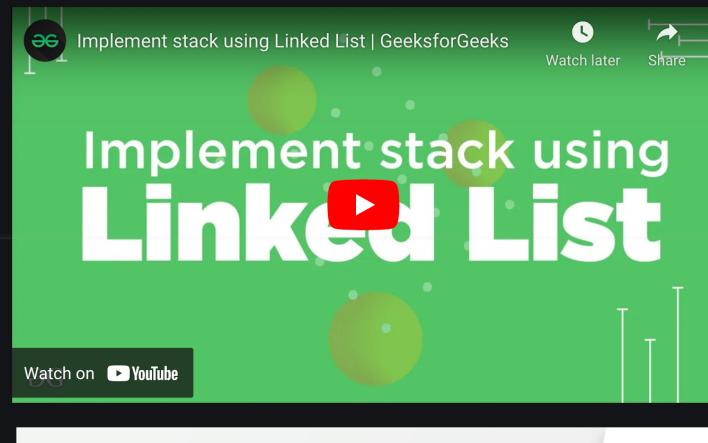
```

44->33->22->11->
Top element is 44
22->11->
Top element is 22

```

The time complexity for all push(), pop(), and peek() operations is O(1) as we are not performing any kind of traversal over the list. We perform all the operations through the current pointer only.

Space complexity: O(n) where n is size of the stack



DSA Self-Paced Course

Curated by experts
Trusted by 1 Lac+ students.

[Enrol Now](#)

GeeksforGeeks

Like 95

< Previous

Queue - Linked List Implementation

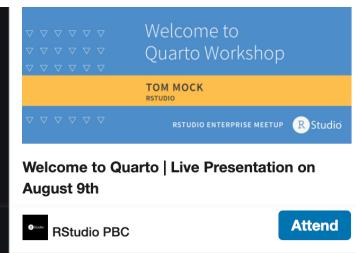
Next >

Stack Data Structure (Introduction and Program)

AD

August 9th @ 12p ET





RECOMMENDED ARTICLES

Page : 1 2 3

- 01 [Program to implement Singly Linked List in C++ using class](#)

22, Jul 21

- 02 [Convert Singly Linked List to XOR Linked List](#)

18, Jan 19

- 03 [Difference between Singly linked list and Doubly linked list](#)

07, Jan 19

- 04 [Convert singly linked list into circular linked list](#)

22, Sep 17

- 05 [How to implement a Stack using list in C++ STL](#)

14, Jul 21

- 06 [C program to create copy of a singly Linked List using Recursion](#)

20, Oct 20

- 07 [How to insert a Node in a Singly Linked List at a given Position using Recursion](#)

02, Dec 21

- 08 [Delete all Prime Nodes from a Singly Linked List](#)

17, Oct 18

● ● ●

Article Contributed By :



Rajput-Ji
@Rajput-Ji

Vote for difficulty

Current difficulty : [Easy](#)

[Easy](#) [Normal](#) [Medium](#) [Hard](#) [Expert](#)

Improved By : [29AjayKumar](#), [AmAnhasNoName](#), [SumanyuChaudhary](#), [7knight](#), [ryadav2](#), [rag2127](#), [gabaa406](#), [geethika1129](#), [polymatir3j](#)

Article Tags : [Technical Scripter 2018](#), [Linked List](#), [Stack](#), [Technical Scripter](#)

Practice Tags : [Linked List](#), [Stack](#)

[Improve Article](#)

[Report Issue](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



 GeeksforGeeks

📍 A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

✉️ feedback@geeksforgeeks.org



Company

About Us

Careers

In Media

Contact Us

Privacy Policy

Copyright Policy

Learn

Algorithms

Data Structures

SDE Cheat Sheet

Machine learning

CS Subjects

Video Tutorials

Courses

News

Top News

Technology

Work & Career

Business

Finance

Lifestyle

Knowledge

Languages

Python

Java

CPP

HTML

Golang

C#

SQL

Kotlin

Web Development

Web Tutorials

Django Tutorial

HTML

JavaScript

Bootstrap

ReactJS

NodeJS

Contribute

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships

Video Internship

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !