



**DOPE** (/blog/)

# Stacks in C

🕒 May 26, 2017    📁 C (/blog/tag/c/?tag=c) STACK (/blog/tag/stack/?tag=stack) DATA STRUCTURE (/blog/tag/data-structure/?tag=data-structure)    👁 4516



Become an Author

[\(/blog/submit-article/\)](/blog/submit-article/)

**Download Our App.**



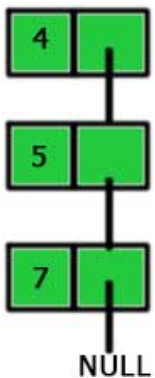
(<https://play.google.com/store/apps/details?id=com.blogsdope&pcampaignid=MKT-Other-global-all-co-prtnr-py-PartBadge-Mar2515-1>)

A stack in computer science is very much similar to the stacks in our day to day life. In a stack, we add or remove any item from the top. In computer science also, we add or remove any node only from the top. We put the newest element on the top of the stack and while removing, we remove this lastly-added element from the top i.e., Last-in, First-out (LIFO). The picture below represents a stack of coins.



Thus, a stack is an abstract data type where new nodes can be added and removed only at the top. There are some terms and operations we use with a stack. Let's go through them one by one.

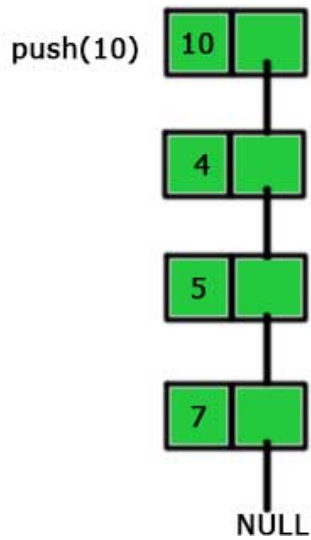
***The bottom of a stack is represented by NULL.***



You can see that the bottom of the stack represented in the above picture is NULL.

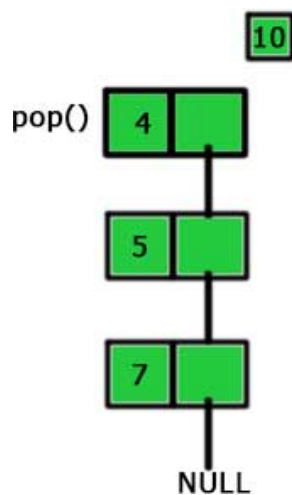
***Two major operations possible on a stack are push and pop.*** Let's go through them.

- ***push*** → 'push' is a function in a stack which adds a new element in the stack. As a new element is added on the top of the stack, so 'push' adds a new node on the top of the stack.



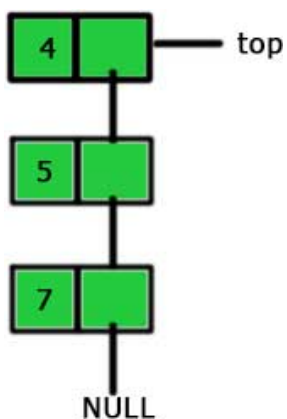
Here, 'push(10)' added a new node on the top of the previous stack.

- **pop** → 'pop' is opposite of push. It returns and removes a node from the top.



In the picture shown above, 'pop' returned 10 and removed that node from the previous stack.

*We also have a pointer which always points at the top of the stack. We call this pointer 'top'.*



There are also some other operations which we can use on a stack. They are listed below.


- ***initialize()*** → To initialize a stack
- ***top()*** → To return the topmost element of the stack
- ***isempty()*** → To check whether the stack is empty or not.
- ***isfull()*** → To check whether the stack is full or not.

This article was just about introducing you to the stack. Next two articles are on coding up a stack using linked list and array.

Next:


- Making a stack using linked list in C  
(<https://www.codesdope.com/blog/article/making-a-stack-using-linked-list-in-c/>)
- Making a stack using an array in C  
(<https://www.codesdope.com/blog/article/making-a-stack-using-an-array-in-c/>)

## Liked the post?

 (<https://www.facebook.com/sharer/sharer.php?u=https://www.codesdope.com/blog/article/stacks-in-c/>)

 (<https://twitter.com/intent/tweet?url=https://www.codesdope.com/blog/article/stacks-in-c/&text=Stacks in C &via=codesdope>)

 (<https://plus.google.com/share?url=https://www.codesdope.com/blog/article/stacks-in-c/>)

 (<https://www.linkedin.com/shareArticle?url=https://www.codesdope.com/blog/article/stacks-in-c/&title=Stacks in C>)

 ([https://pinterest.com/pin/create/bookmarklet/?media=https://www.codesdope.com/media/blog\\_images/1/2017/5/31/bank-20795\\_640.jpg&url=https://www.codesdope.com/blog/article/stacks-in-c/&description=Stacks in C](https://pinterest.com/pin/create/bookmarklet/?media=https://www.codesdope.com/media/blog_images/1/2017/5/31/bank-20795_640.jpg&url=https://www.codesdope.com/blog/article/stacks-in-c/&description=Stacks in C))