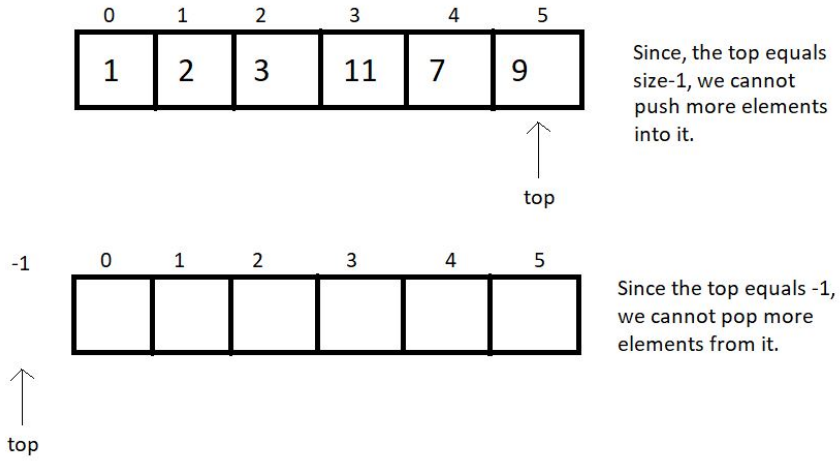
Push, Pop and Other Operations in Stack Implemented Using an Array

codewithharry.com/videos/data-structures-and-algorithms-in-hindi-25

We had already finished the basics of a stack, and its implementation using arrays. We have gained enough confidence by writing the codes for implementing stacks using arrays in C. Now we can learn about the operations one can perform on a stack while executing them using arrays.

We concluded our last tutorial with two of the most important points:

1. One cannot push more elements into a full-stack.
2. One cannot pop any more elements from an empty stack.

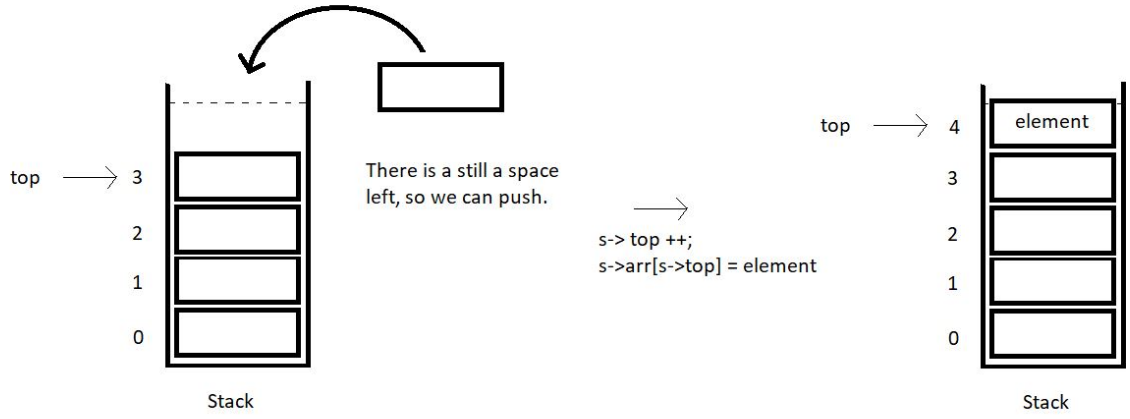


Declaring a stack was done in the last tutorial as well. Let's keep that in mind as we proceed.

Operation 1: Push-

1. The first thing is to define a stack. Suppose we have the creating stack and declaring its fundamentals part done, then pushing an element requires you to first check if there is any space left in the stack.

2. Call the *isFull* function which we did in the previous tutorial. If it's full, then we cannot push anymore elements. This is the case of stack overflow. Otherwise, increase the variable *top* by 1 and insert the element at the index *top* of the stack.



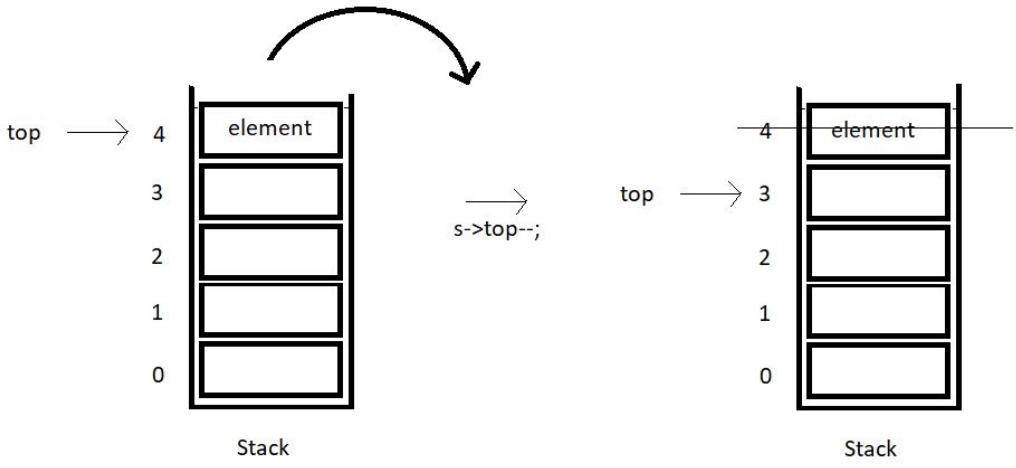
3. So, this is how we push an element in a stack array. Suppose we have an element *x* to insert in a stack array of size 4. We first checked if it was full, and found it was not full. We retrieved its *top* which was 3 here. We made it 4 by increasing it once. Now, just insert the element *x* at index 4, and we are done.

Operation 2: Pop-

Popping an element is very similar to inserting an element. You should first give it a try yourself. There are very subtle changes.

1. The first thing again is to define a stack. Get over with all the fundamentals. You must have learnt that by now. Then popping an element requires you to first check if there is any element left in the stack to pop.

2. Call the *isEmpty* function which we practiced in the previous tutorial. If it's empty, then we cannot pop any element, since there is none left. This is the case of stack underflow. Otherwise, store the topmost element in a temporary variable. Decrease the variable *top* by 1 and return the temporary variable which stored the popped element.



3. So, this is how we pop an element from a stack array. Suppose we make a pop call in a stack array of size 4. We first checked if it was empty, and found it was not empty. We retrieved its *top* which was 4 here. Stored the *element* at 4. We made it 3 by decreasing it once. Now, just return the element *x,* and popping is done.

So, this was known about the concepts behind pushing and popping elements in a stack. You'll enjoy it more when we'll implement their codes in C in the next tutorial. I always encourage self-motivation. Try them yourselves first and let me know if you could. We still have a lot to do. We still have many operations to complete. We'll cover them all slowly and steadily. Enjoy the ride.

I appreciate your support throughout. I hope you enjoyed the tutorial. If you genuinely appreciate my work, please let your friends know about this course too. If you haven't checked out the whole playlist yet, move on to codewithharry.com or my YouTube channel to access it. See you all in the next tutorial where we'll code these operations of stacks in C. Till then keep learning.