Introduction to Stack in Data Structures

CodeWithHarry Menu

Login

Show Course Contents

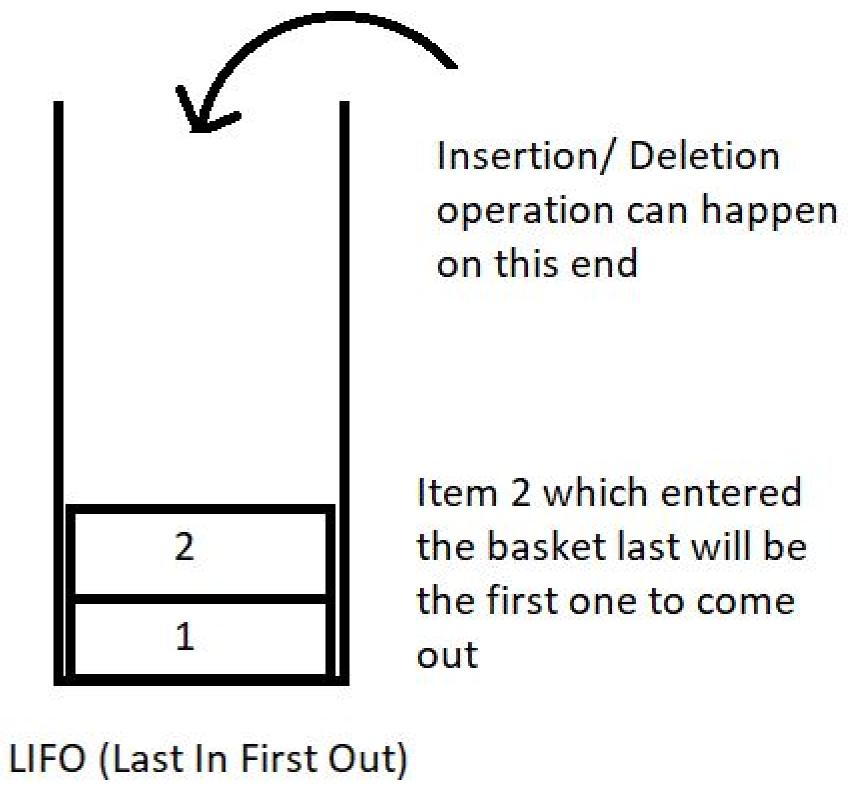
Overview Q&A Downloads Announcements

Introduction to Stack in Data Structures

It has been a while since we started this DSA course. We saw array ADT, linked lists and their variants, their implementation, and their operations. From this tutorial on, we will start learning about stack data structures.

Introduction:

A stack is a linear data structure. Any operation on the stack is performed in LIFO (Last In First Out) order. This means the element to enter the container last would be the first one to leave the container. It is imperative that elements above an element in a stack must be removed first before fetching any element.
An element can be pushed in this basket-type container illustrated below. Any basket has a limit, and so does our container too. Elements in a stack can only be pushed to a limit. And this extra pushing of elements in a stack leads to stack overflow.


Insertion/ Deletion operation can happen on this end
2
1
Item 2 which entered the basket last will be the first one to come out
LIFO (Last In First Out)

Applications of Stack:

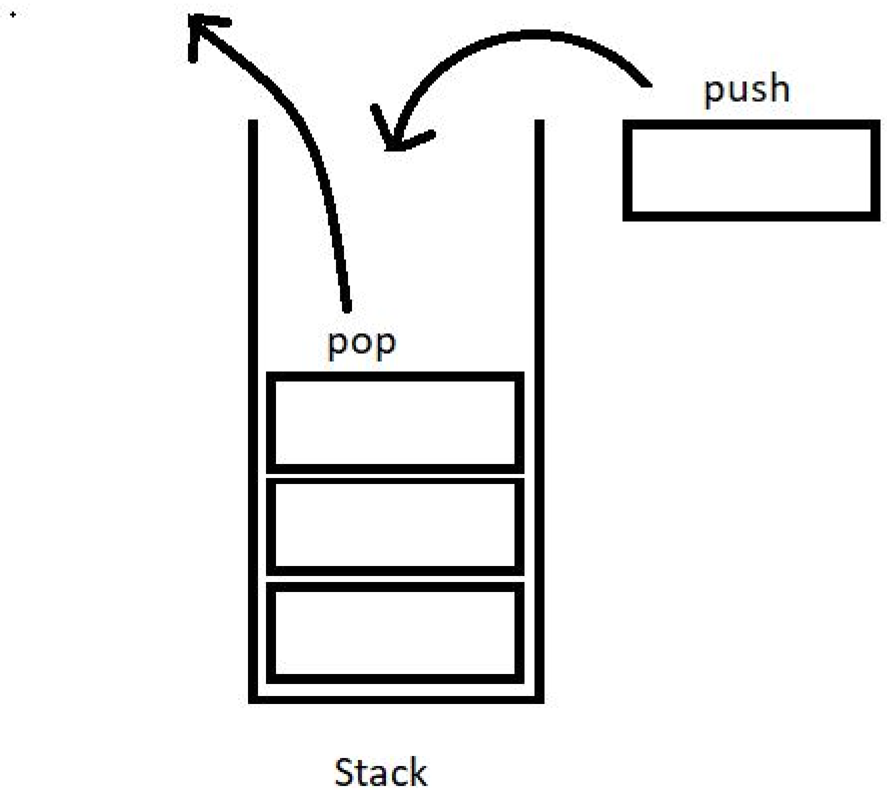
1. We have talked about function calls before as well. A function until it returns reserves a space in the memory stack. Any function embedded in some function comes above the parent function in the stack. So, first, the embedded function ends, and then the parent one. Here, the function called last ends first. (LIFO).
2. Infix to postfix conversion (and other similar conversions) will be dealt with in the coming tutorials.
3. Parenthesis matching and many more...

Stack ADT:

In order to create a stack, we need a pointer to the topmost element to gain knowledge about the element which is on the top so that any operation can be carried about. Along with that, we need the space for the other elements to get in and their data.

Here are some of the basic operations we would want to perform on stacks:

1. push(): to push an element into the stack
2. pop(): to remove the topmost element from the stack



3. peek(index): to return the value at a given index
4. isempty() / isfull() : to determine whether the stack is empty or full to carry efficient push and pull operations.

Implementation:

A stack element can be implemented by both an array and a linked list. We'll see both these methods in the coming tutorials.
A stack is a collection of elements with certain operations following the LIFO (Last in First Out) discipline. If you are dealing with stacks, you should not forget about this. So, we'll consider everything about stacks from the basics. Just stay.
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. Don't forget to

download the notes from the link given below. 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 implement stacks using arrays. Till then, keep learning.

Download Notes Here

Previous Next

CodeWithHarry