Implement two stacks in an array

Create a data structure *twoStacks* that represents two stacks. Implementation of *twoStacks* should use only one array, i.e., both stacks should use the same array for storing elements. Following functions must be supported by *twoStacks*.

push1(int x) –> pushes x to first stack
push2(int x) –> pushes x to second stack

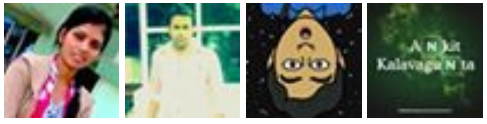
pop1() –> pops an element from first stack and return the popped element
pop2() –> pops an element from second stack and return the popped element

Implementation of *twoStack* should be space efficient.

Method 1 (Divide the space in two halves)
A simple way to implement two stacks is two divide the array in two halves and assign the half half space to two stacks, i.e., use arr[0] to arr[n/2] for stack1, and arr[n/2+1] to arr[n-1] for stack2 where arr[] is the array to be used to implement two stacks and size of array be n.

The problem with this method is inefficient use of array space. A stack push operation may result in stack overflow even if there is space available in arr[]. For example, say the array size is 6 and we push 3 elements to stack1 and do not push anything to second stack2. When we push 4th element to stack1, there will be overflow even if we have space for 3 more elements in array.

Method 2 (A space efficient implementation)
This method efficiently utilizes the available space. It doesn't cause an overflow if there is space available in arr[]. The idea is to start two stacks from two extreme corners of arr[]. stack1 starts from the leftmost element, the first element in stack1 is pushed at index 0. The stack2 starts from the rightmost corner, the first element in stack2 is pushed at index (n-1). Both stacks grow



Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

(or shrink) in opposite direction. To check for overflow, all we need to check is for space between top elements of both stacks. This check is highlighted in the below code.

```
#include<iostream>
#include<stdlib.h>

using namespace std;

class twoStacks
{
    int *arr;
    int size;
    int top1, top2;
public:
   twoStacks(int n)  // constructor
   {
       size = n;
       arr = new int[n];
       top1 = -1;
       top2 = size;
   }

   // Method to push an element x to stack1
   void push1(int x)
   {
       // There is at least one empty space for new element
       if (top1 < top2 - 1)
       {
           top1++;
           arr[top1] = x;
       }
       else
       {
           cout << "Stack Overflow";
           exit(1);
       }
   }

   // Method to push an element x to stack2
   void push2(int x)
   {
       // There is at least one empty space for new element
       if (top1 < top2 - 1)
       {
           top2--;
           arr[top2] = x;
       }
```


Popular Posts

```
        }
        else
        {
            cout << "Stack Overflow";
            exit(1);
        }
    }

    // Method to pop an element from first stack
    int pop1()
    {
        if (top1 >= 0 )
        {
           int x = arr[top1];
           top1--;
           return x;
        }
        else
        {
            cout << "Stack UnderFlow";
            exit(1);
        }
    }

    // Method to pop an element from second stack
    int pop2()
    {
        if (top2 < size)
        {
           int x = arr[top2];
           top2++;
           return x;
        }
        else
        {
            cout << "Stack UnderFlow";
            exit(1);
        }
    }
};


/* Driver program to test twStacks class */
int main()
{
    twoStacks ts(5);
```

```
    ts.push1(5);
    ts.push2(10);
    ts.push2(15);
    ts.push1(11);
    ts.push2(7);
    cout << "Popped element from stack1 is " << ts.pop1();
    ts.push2(40);
    cout << "\nPopped element from stack2 is " << ts.pop2();
    return 0;
}
```

Output:

```
Popped element from stack1 is 11
Popped element from stack2 is 40
```

Time complexity of all 4 operations of *twoStack* is O(1).
We will extend to 3 stacks in an array in a separate post.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recent Comments

Aman Hi, Why arent we checking for conditions...
Write a C program to Delete a Tree. · 15 minutes ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 19 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 44 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...
Count trailing zeroes in factorial of a number · 45 minutes ago

newCoder3006 If the array contains negative numbers also. We...
Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...
Find subarray with given sum · 1 hour ago

Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

Writing code in comment? Please use **ideone.com** and share the link here.

22 Comments GeeksforGeeks

Sort by Newest

Arulmozhi • 4 months ago
how about storing elements alternately. This will implement 2 stacks in a array
• Reply • Share ›

仮面 の男 → Arulmozhi • 4 months ago
With the additional restriction that neither stack should overflow unless idea wouldn't work, but if not, then it is okay.
• Reply • Share ›

Arifa Khan • a year ago

that&#039s not good.

Reply • Share ›

abhishek08aug • a year ago

```
#include<stdio.h>

struct two_stacks {
  int first_stack_top;
  int second_stack_top;
  int size;
  int * array;
};

void push(struct two_stacks * ts, int insert_value, int stack_num) {
  if(stack_num==1) {
    if(ts->first_stack_top+1==ts->second_stack_top) {
      printf("ERROR: two stack array full! can't have any more elemen
      return;
    } else {
        ts->first_stack_top++;
        ts->array[ts->first_stack_top]=insert_value;
    }
```

see more

1 • Reply • Share ›

Ganesh • a year ago

You can find java code here:

[sourcecode language="JAVA"]

/**

* Create a data structure twoStacks that represents two stacks. Implementati

```
array;
* i.e., both stacks should use the same array for storing elements.
*
* @author GAPIITD
*
*/
public class TwoStack {
private int stack[];
private int top1, top2;

TwoStack() {
this(10);
}
```

see more

1 • Reply • Share ›

abhay • a year ago

using an array to store the even and odd numbers...

```
 #include<stdio.h>
#include<stdlib.h>



struct node
{
    int data;
    struct node* next;
};
```

```
void insertbeg(int value,struct node **head)
{
struct node *temp=malloc(sizeof(struct node));
temp->data=value;
```

see more

• Reply • Share ›

Geek • a year ago

Even Better Memory optimized solution..

1) push everything from start
2) Create link lists inside array..
say input order:
which stack it should go
stack 1 : 10
stack 2 : 12
stack 2 : 16
stack 1 : 1
stack 2 : 23
answer is:

```
        _______________
       |      |        |
10     12     16   1  23
|_________________1
```

Always Maintain 4 pointers
stack1Head, stack1Tail
stack1Head stack2Tail

obvs you can get idea about popping.

/* Paste your code here (You may delete these lines if not writing code) */

Geek ➚ Geek • a year ago

/*

____

| | |

10 12 16 1 23

|_____1

*/

Geek ➚ Geek • a year ago

Not able to fix this indentation...

lists will be

12->16->23

16->1

Thirunavukkarasu • 2 years ago

```
 #include<stdio.h>
int arr[10];
int top1=-1,top2=sizeof(arr)/sizeof(arr[0]);
void push1(int data)
{
        if(top1+1==top2)
        { printf("Stack is full "); return;}
        else
```

```
        {
                top1++;
                arr[top1]=data;
        }
}

void push2(int data)
{
        if(top1==top2-1)
        {
```

see more

Reply • Share ›

Seema • 2 years ago

Great work! Keep it on!

Reply • Share ›

Krupa • 2 years ago

Making 2-stacks in the method(2) presented above is nothing but how OS can
growing in reverse direction. Its simple solution.

Reply • Share ›

test • 2 years ago

test

Reply • Share ›

Inderpreet Singh • 2 years ago

Best Stuff. It's very easy to understand. I tried to understand this concept man
i could not. After reading this post, i understood it . Thanks Man and Thanks G

Reply • Share ›

In Second method while we do pop operation. Their is a case that while we po
first item because we are not differentiating between items of two stack. Corre

```
/* Paste your code here (You may delete these lines if not writing c
```

• Reply • Share ›

kartik ➚ suhas meena • 2 years ago

That is not possible because there are two different top variables and c

• Reply • Share ›

kunal • 2 years ago

you people provide the best stuff to crack interview

keep it on :)

• Reply • Share ›

laxmi • 2 years ago

like your explanation

• Reply • Share ›

Other Neo • 2 years ago

The code look amateur "C" code written at the very best....

1. push1 and push2 do not return the error/exception to the caller
2. Big Blunder in terms of EXIT(0)
3. Duplicate code with respect to push1 & push2 and pop1 and pop2. Ideally p
parameter about the stack on which to operate

• Reply • Share ›

kartik ➚ Other Neo • 2 years ago

@Other Neo: Thanks for your inputs. The coding is done this way to ke

provide more details or reference about the problem with exit(0).

Also, all code in push1() and push2() not duplicatae, it's just the overflc operations does top1++ and push2() does top1--. Similarly pop1() and

Other Neo ➚ kartik • 2 years ago

@Kartik "exit(0)" are old rudimentary ways for error handling dc code just shows the arrogance of the code / programmer. It bre like goto) and closes the program without any possibility of recc

In your code the instead of EXIT(0) you could have simply writte the opportunity to the caller of the function (main) to deal with th

PUSH functions are exact duplicates of each other if we ignore increment/decrement; and the same goes with POP. For illustr change from a static array to std::vector, you would need to cha PUSH.

If you are still not convinced let me know and I will rewrite the w

lalor ➚ Other Neo • 2 years ago

I hope you give the whole code.

```
/* Paste your code here (You may delete these li
```