Data Structures: Stacks (with C Program source code)

To go through the C program / source-code, scroll this page

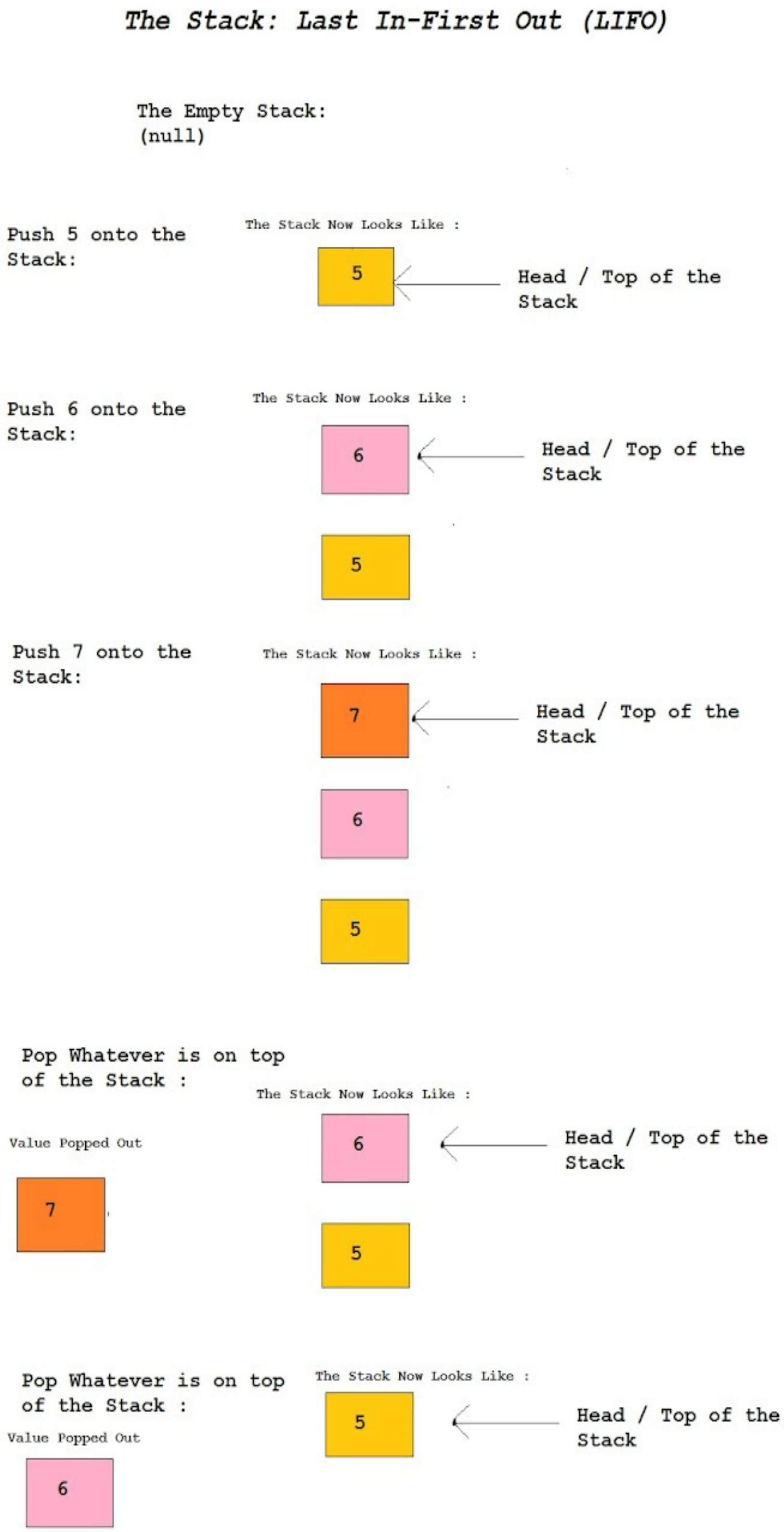
Stack

Stack is a specialized data storage structure (Abstract data type). Unlike, arrays access of elements in a stack is restricted. It has two main functions push and pop. Insertion in a stack is done using push function and removal from a stack is done using pop function. Stack allows access to only the last element inserted hence, an item can be inserted or removed from the stack from one end called the top of the stack. It is therefore, also called Last-In-First-Out (LIFO) list. Stack has three properties: capacity stands for the maximum number of elements stack can hold, size stands for the current size of the stack and elements is the array of elements.

Mathematics
Mathematics - Linear Algebra
Mathematics - Geometry
Mathematics - Single Variable Calculus
Mathematics - Game Theory
Mathematics - Operations Research
Physics
Physics - Basic Mechanics
Physics - Electrostatics
Sitemap
Recent site activity

These are Quick Links to sections of the Main Page

0



Algorithm:

Stack structure is defined with fields capacity, size and *elements (pointer to the array of elements).

Functions – (explained in greater detail in the document)

1. **createStack function**– This function takes the maximum number of elements (maxElements) the stack can hold as an argument, creates a stack according to it and returns a pointer to the stack. It initializes Stack S using malloc function and its properties.
2. **push function** - This function takes the pointer to the top of the stack S and the item (element) to be inserted as arguments. Check for the emptiness of stack
3. **pop function** - This function takes the pointer to the top of the stack S as an argument.
4. **top function** – This function takes the pointer to the top of the stack S as an argument and returns the topmost element of the stack S.

Properties of stacks:

1. Each function runs in O(1) time.
2. It has two basic implementations
Array-based implementation – It is simple and efficient but the maximum size of the stack is fixed.
Singly Linked List-based implementation – It's complicated but there is no limit on the stack size, it is subjected to the available memory.

Complete tutorial with examples :

Stack

Stack is a specialized data storage structure (Abstract data type). Unlike, arrays access of elements in a stack is restricted. It has two main functions **push** and **pop**. Insertion in a stack is done using **push** function and removal from a stack is done using **pop** function. Stack allows access to only the last element inserted hence, an item can be inserted or removed from the stack from one end called the **top** of the stack. It is therefore, also called Last-In-First-Out (LIFO) list. Stack has three properties: **capacity** stands for the maximum number of elements stack can hold, **size** stands for the current size of the stack and **elements** is the array of elements.

Algorithm:

Stack structure is defined with fields capacity, size and *elements (pointer to the array of elements).

Functions –

1. createStack function– This function takes the maximum number of elements (maxElements) the stack can hold as an argument, creates a stack according to it and returns a pointer to the stack. It initializes Stack S using malloc function and its properties.
   elements = (int *)malloc(sizeof(int)*maxElements).
   S->size = 0, current size of the stack S.
   S->capacity = maxElements, maximum number of elements stack S can hold.
2. push function - This function takes the pointer to the top of the stack S and the item (element) to be inserted as arguments. Check for the emptiness of stack
   If S->size is equal to S->capacity, we cannot push an element into S as there is no

Stacks - C Program source code

```
#include<stdio.h>
#include<stdlib.h>
/* Stack has three properties. capacity stands for the maximum number of elements stack can hold.
   Size stands for the current size of the stack and elements is the array of elements */
typedef struct Stack
{
        int capacity;
        int size;
```

```
        int *elements;
}Stack;
/* crateStack function takes argument the maximum number of elements the stack can hold, creates
   a stack according to it and returns a pointer to the stack. */
Stack * createStack(int maxElements)
{
        /* Create a Stack */
        Stack *S;
        S = (Stack *)malloc(sizeof(Stack));
        /* Initialise its properties */
        S->elements = (int *)malloc(sizeof(int)*maxElements);
        S->size = 0;
        S->capacity = maxElements;
        /* Return the pointer */
        return S;
}
void pop(Stack *S)
{
        /* If stack size is zero then it is empty. So we cannot pop */
        if(S->size==0)
        {
                printf("Stack is Empty\n");
                return;
        }
        /* Removing an element is equivalent to reducing its size by one */
        else
        {
                S->size--;
        }
        return;
}
int top(Stack *S)
{
        if(S->size==0)
        {
                printf("Stack is Empty\n");
                exit(0);
        }
        /* Return the topmost element */
        return S->elements[S->size-1];
}
void push(Stack *S,int element)
{
        /* If the stack is full, we cannot push an element into it as there is no space for it.*/
        if(S->size == S->capacity)
        {
                printf("Stack is Full\n");
        }
        else
        {
                /* Push an element on the top of it and increase its size by one*/
                S->elements[S->size++] = element;
        }
        return;
}
int main()
{
        Stack *S = createStack(5);
        push(S,7);
        push(S,5);
        push(S,21);
        push(S,-1);
        printf("Top element is %d\n",top(S));
        pop(S);
        printf("Top element is %d\n",top(S));
        pop(S);
        printf("Top element is %d\n",top(S));
        pop(S);
        printf("Top element is %d\n",top(S));

}
```

Related Tutorials :

Stacks	Last In First Out data structures (LIFO). Like a stack of cards from which you pick up the one on the top (which is the last one to be placed on top of the stack). Documentation of the various operations and the stages a stack passes through when elements are inserted or deleted. C program to help you get an idea of how a stack is implemented in code.	Queues	First in First Out data structure (FIFO). Like people waiting to buy tickets in a queue - the first one to stand in the queue, gets the ticket first and gets to leave the queue first. Documentation of the various operations and the stages a queue passes through as elements are inserted or deleted. C Program source code to help you get an idea of how a queue is implemented in code.

Some Important Data Structures and Algorithms, at a glance:

Arrays : Popular Sorting and Searching Algorithms			
Bubble Sort	Insertion Sort	Selection Sort	Shell Sort
Merge Sort	Quick Sort	Heap Sort	Binary Search Algorithm
Basic Data Structures and Operations on them			
Stacks	Queues	Single Linked List	Double Linked List
Circular Linked List	1.		

Tree Data Structures			
Binary Search Trees	Heaps	Height Balanced Trees	
Graphs and Graph Algorithms			
Depth First Search	Breadth First Search	Minimum Spanning Trees: Kruskal Algorithm	Minumum Spanning Trees: Prim's Algorithm
Dijkstra Algorithm for Shortest Paths	Floyd Warshall Algorithm for Shortest Paths	Bellman Ford Algorithm	
Popular Algorithms in Dynamic Programming			
Dynamic Programming	Integer Knapsack problem	Matrix Chain Multiplication	Longest Common Subsequence
Greedy Algorithms			

Elementary cases : Fractional Knapsack Problem, Task Scheduling	Data Compression using Huffman Trees		

5 comments

Add a comment...

Mahmoud El Akrami · Indonesian Computer University
It's good code.
Reply · Like · October 17 at 4:07am

Anjali Shukla · Kanpur, Uttar Pradesh
nice code
Reply · Like · September 29 at 12:31pm

Vikas Acharya · Banswara
nice, but very short.
Reply · Like · September 24 at 12:48am

Prashant Bhattacharji · IIT Kharagpur
The embedded document in the middle had blanked ou
computer-science/data-structures-stacks--with-c-p
Reply · Like · September 26 at 6:27am

Ansari Tahir · GTU
thanks..it is very useful..
Reply · Like · September 17 at 4:30am

View 1 mor