Data Structures and Algorithms

MSc. KhangVQH

Faculty Of Information Technology

Fall - 2022

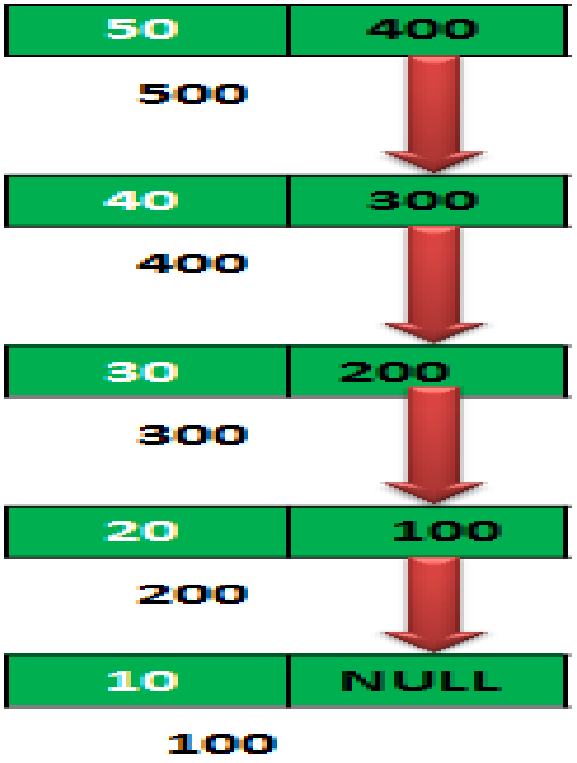
Stack Linked List

Terminology: Top, Node, Data, Next, NULL

Operations: push( ), pop( ), Display()

Stack Linked List

Top--->



content

- Introduction
- What is Stack Linked List?
- Stack Operations using Linked List
- Algorithm for Push()
- Algorithm for Pop()
- Algorithm for Display()
- Coding
- Output

Introduction

- The major problem with the stack implemented using an array is, it works only for a fixed number of data values.
- That means the amount of data must be specified at the beginning of the implementation itself.
- Stack implemented using an array is not suitable, when we don't know the size of data which we are going to use.

What is Stack Linked List?

- A stack data structure can be implemented by using a linked list data structure.
- The stack implemented using linked list can work for an unlimited number of values.
- In stack implemented using linked list, there is no need to fix the size at the beginning of the implementation.

What is Stack Linked List?

- In linked list implementation of a stack, every new element is inserted at **'top'** position.
- Here every newly inserted element is pointed by **'top‘** variable.
- If you want to remove an element from the stack, simply remove the node which is pointed **by 'top'** by moving **'top'** to its previous node in the list.
- The **next** field of the first element(node) should be always **NULL.**

Stack Operations using Linked List

- Push (To insert an element on to the stack)
- Pop (To delete an element from the stack)
- Display (To display elements of the stack)

Structure of Node

- To implement a stack using a linked list, we need to define the structure for node
- **Step 1 -** Define a **'Node'** structure with two members **data** and **next**.
- **Step 2 -** Define a **Node** pointer **'top'** and set it to **NULL**.

Node Structure

```
Structure Node
{
Int data;
Structure Node *Next;
}*top=NULL;
```

Node

100

Algorithm for Push Operation

We can use the following steps to insert a new node into the stack...

Step 1 - Create a **newNode** with given value.

set **newNode → data = Value ;**

Step 2 - Check whether stack is **Empty (top == NULL)**

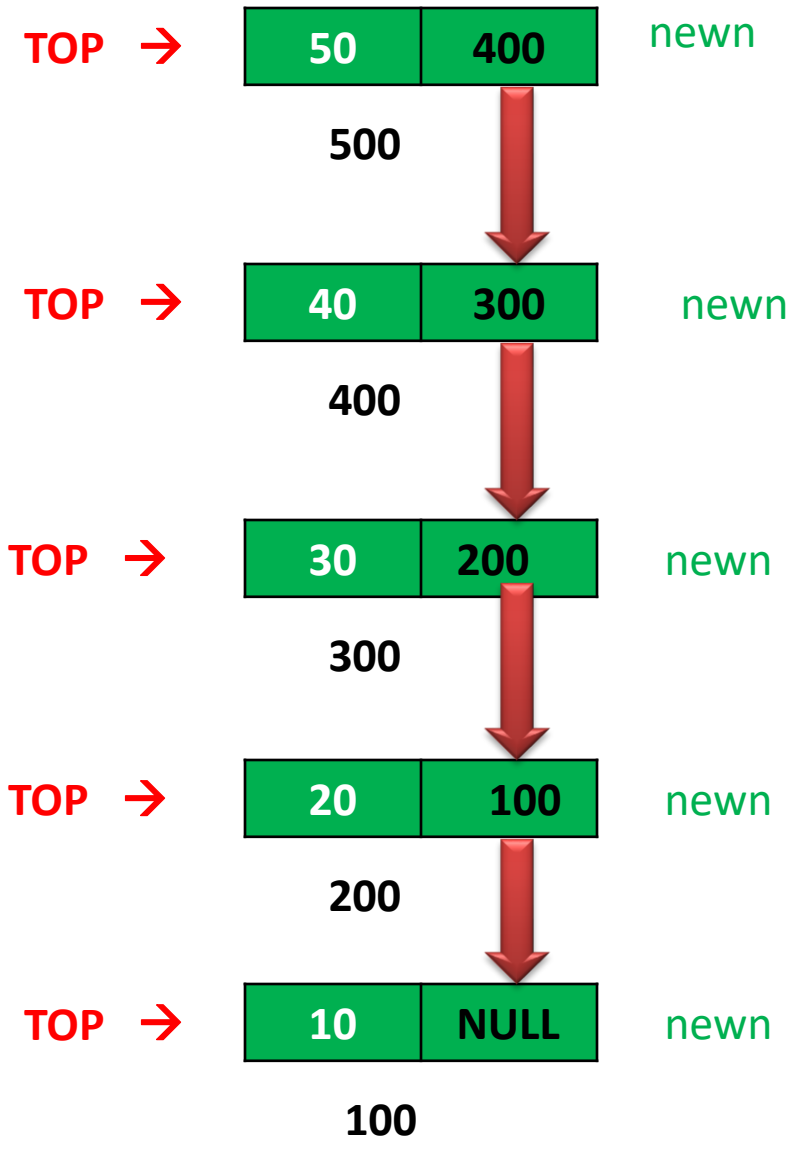
Step 3 - If it is **Empty**, then

set **newNode → next = NULL;.**

Step 4 - If it is **Not Empty**, then

set **newNode → next = top.**

Step 5 - Finally, set **top = newNode**.



Step 1 - Create a newNode with given value.

set newNode → data = Value ;

Step 2 - Check whether stack is Empty (top == NULL)

Step 3 - If it is Empty, then

set newNode → next = NULL;.

Step 4 - If it is Not Empty, then

set newNode → next = top.

Step 5 - Finally, set top = newNode.

Pop (To delete an element from the stack)

We can use the following steps to delete a node from the stack...

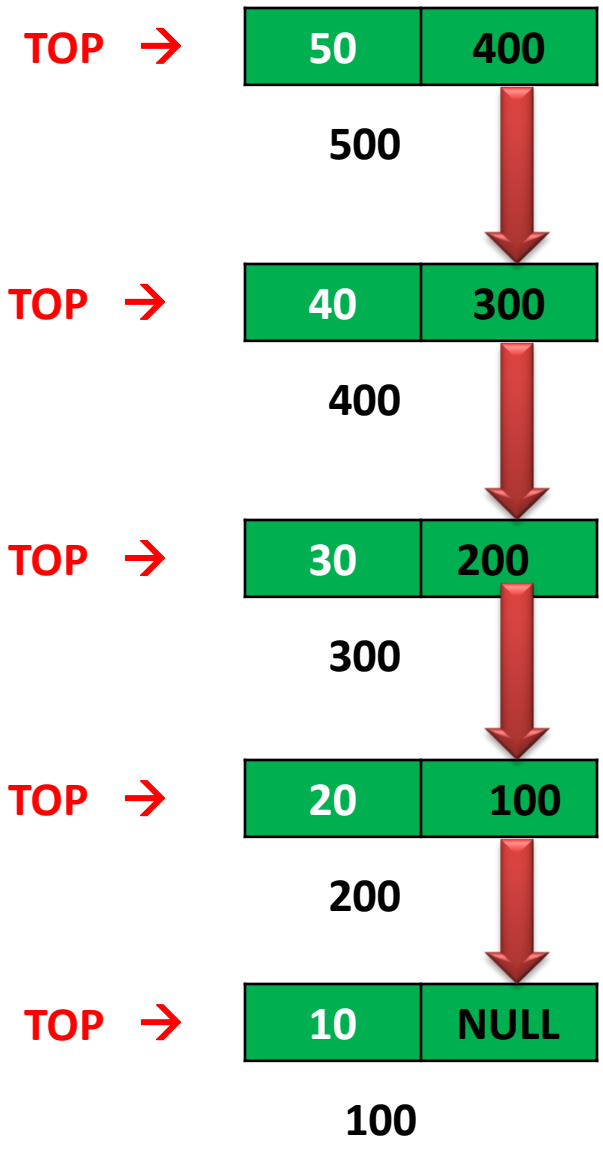
Step 1 - Check whether **stack** is **Empty (top == NULL).**

Step 2 - If it is **Empty**, then display **"Stack is Empty!!! Deletion is not possible!!!"** and terminate the function

Step 3 - If it is **Not Empty**, then define a **Node** pointer **'temp'** and set it to **'top'**.

Step 4 - Then set **'top = top → next'.**

Step 5 - Finally, delete **'temp'. (free(temp)).**



TOP=NULL

Stack has no Elements.
List is empty

Display (To display elements of the stack)

- We can use the following steps to display the elements (nodes) of a stack...

Step 1 - Check whether stack is **Empty (top == NULL).**

Step 2 - If it is **Empty,** then display **'Stack is Empty!!!'** and terminate the function.

Step 3 - If it is **Not Empty**, then define a Node pointer **'temp'** and initialize with **top**.

Step 4 - Display **'temp → data --->'** and move it to the next node. Repeat the same until **temp** reaches to the first node in the stack. **(temp → next != NULL).**

Step 5 - Finally! Display **'temp → data ---> NULL'.**

Code for Stack SLL

Implementation of Stack using Linked List | C Programming

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

struct Node
{
   int data;
   struct Node *next;
}*top = NULL;

void push(int);
void pop();
void display();

void main()
{
   int choice, value;
   clrscr();
   printf("\n:: Stack using Linked List ::\n");
   while(1){
      printf("\n****** MENU ******\n");
      printf("1. Push\n2. Pop\n3. Display\n4. Exit\n");
      printf("Enter your choice: ");
      scanf("%d",&choice);
      switch(choice){
         case 1: printf("Enter the value to be insert: ");
                 scanf("%d", &value);
                 push(value);
                 break;
         case 2: pop(); break;
         case 3: display(); break;
         case 4: exit(0);
         default: printf("\nWrong selection!!! Please try again!!!\n");
      }
   }
}
```

```
}
void push(int value)
{
   struct Node *newNode;
   newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   if(top == NULL)
      newNode->next = NULL;
   else
      newNode->next = top;
   top = newNode;
   printf("\nInsertion is Success!!!\n");
}
void pop()
{
   if(top == NULL)
      printf("\nStack is Empty!!!\n");
   else{
      struct Node *temp = top;
      printf("\nDeleted element: %d", temp->data);
      top = temp->next;
      free(temp);
   }
}
void display()
{
   if(top == NULL)
      printf("\nStack is Empty!!!\n");
   else{
      struct Node *temp = top;
      while(temp->next != NULL){
         printf("%d--->",temp->data);
         temp = temp -> next;
      }
      printf("%d--->NULL",temp->data);
   }
}
```

OUTPUT

```
C:\WINDOWS\system32\cmd.exe - tc

:: Stack using Linked List ::

****** MENU ******
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 10

Insertion is Success!!!

****** MENU ******
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 20_
```

C:\WINDOWS\system32\cmd.exe - tc

```
4. Exit
Enter your choice: 2

Deleted element: 20
****** MENU ******
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
10--->NULL
****** MENU ******
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2

Deleted element: 10
****** MENU ******
1. Push
2. Pop
3. Display
4. Exit
Enter your choice:
```

Thank You