**Godavari Collage Of Engineering.**

**Subject Name:** Data Structure. **Teacher Name:** Prof. S.S.Shete

**Practical No. :**  10 **Date:**

**Class:** S.E **Roll No:**

**Title:**  Write a program to implement a stack using a linked list such that the push and pop operations of stack still take O(1) time.

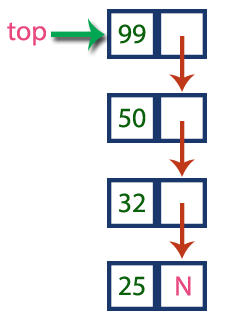
**Aim:** To implement a stack using a linked list such that the push and pop operations of stack still take O(1) time.

**Theory:**

A stack can be easily implemented through the linked list. In stack Implementation , a stack contains a top pointer. Which is “head” of the stack where pushing and popping items happens at the head of the list. first node have null in link field and second node link have first node address in link field and so on and last node address in “top” pointer.

The main advantage of using linked list over an arrays is that it is possible to implements a stack that can shrink or grow as much as needed. In using array will put a restriction to the maximum capacity of the array which can lead to stack overflow. Here each new node will be dynamically allocate. so overflow is not possible.

In linked list implementation of a stack, every new element is inserted as 'top' element. That means every newly inserted element is pointed by 'top'. Whenever we 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 must be always NULL.



**Stack Operations using Linked List**

**1) Push(value) - Inserting an element into the Stack**

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

* Step 1 - Create a newNode with given 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.

**2) Pop() - Deleting an Element from a 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)).

**3) Display() - Displaying stack of elements**

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'.

**Program:**

#include<stdio.h>

#include<malloc.h>

struct Node

{

int data;

struct Node \*next;

}\*top = NULL;

void push(int);

void pop();

void display();

int main()

{

int choice, value;

printf("\n:: Stack using Linked List ::\n");

printf("\n\*\*\*\*\*\* MENU \*\*\*\*\*\*\n");

printf("1. Push\n2. Pop\n3. Display\n4. Exit\n");

REP:

printf("\n\n======================================================");

printf("\n\t 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:

printf("\n\n\t Thanks For VIsiting Program.");

printf("\n\t NOw You Are Exiting From PROGRAM.");

break;

default:

printf("\nWrong selection!!! Please try again!!!\n");

break;

}

if(choice!=4)

{

goto REP;

}

}

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("\n Stack NOde is Inserted.!!!\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()

{

int count=0;

if(top == NULL)

printf("\nStack is Empty!!!\n");

else

{

struct Node \*temp = top;

while(temp->next != NULL)

{

printf("\n\t %dth Stack Node :- %d",count,temp->data);

temp = temp -> next;

count++;

}

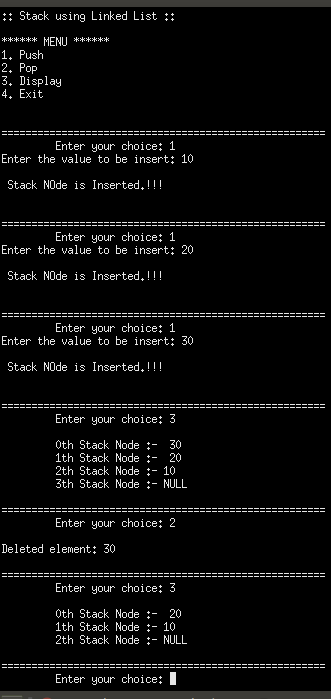
printf("\n\t %dth Stack Node :- %d ",count,temp->data);

count++;

printf("\n\t %dth Stack Node :- NULL ",count);

}

}

**Output**:

**Conclusion:-**