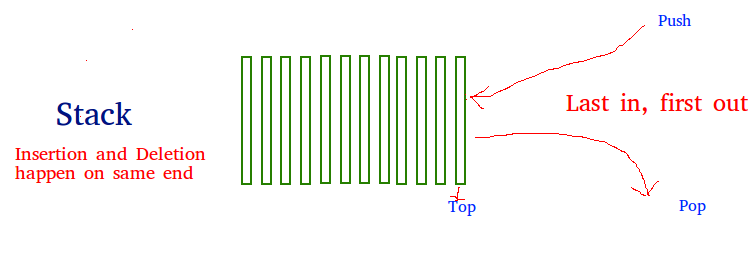
**Aim:-** Design and implement a stack (Linked list implementation) and demonstrate its working with necessary inputs. Display the appropriate messages in case of exceptions.

[GitHub-Link](https://github.com/logan-14/Stack-Operation-Using-Array)

**Theory: -**

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).



There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO(Last In First Out)/FILO(First In Last Out) order.

**Basic Operations**

Stack operations may involve initializing the stack, using it and then de-initializing it. Apart from these basic stuffs, a stack is used for the following two primary operations −

* **push()** − Pushing (storing) an element on the stack.
* **pop()** − Removing (accessing) an element from the stack.

When data is PUSHed onto stack.

To use a stack efficiently, we need to check the status of stack as well. For the same purpose, the following functionality is added to stacks −

* **peek()** − get the top data element of the stack, without removing it.
* **isFull()** − check if stack is full.
* **isEmpty()** − check if stack is empty.

At all times, we maintain a pointer to the last PUSHed data on the stack. As this pointer always represents the top of the stack, hence named **top**. The **top** pointer provides top value of the stack without actually removing it.

ALGORITHM:-

1)For push function:-

* Step 1 - Check whether **stack** is **FULL**. (**top == SIZE-1**)
* Step 2 - If it is **FULL**, then display **"Stack is FULL!!! Insertion is not possible!!!"** and terminate the function.
* Step 3 - If it is **NOT FULL**, then increment **top** value by one (**top++**) and set stack[top] to value (**stack[top] = value**).

2)For pop function:-

* Step 1 - Check whether **stack** is **EMPTY**. (**top == -1**)
* 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 delete **stack[top]** and decrement **top** value by one (**top--**).

3)For display function:-

* Step 1 - Check whether **stack** is **EMPTY**. (**top == -1**)
* 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 variable '**i**' and initialize with top. Display **arr[i]** value and decrement **i** value by one (**i--**).

Step 3 - Repeat above step until **i** value becomes '0'.

**Program:-**

#include <stdio.h>

#include <stdlib.h>

void push();

void pop();

void display();

struct node

{

    int val;

    struct node \*next;

};

struct node \*head;

void main()

{

    int choice = 0;

    printf("\n\*\*\*\*\*\*\*\*\*Stack operations using linked list\*\*\*\*\*\*\*\*\*\n");

    printf("\n----------------------------------------------\n");

    while (choice != 4)

    {

        printf("\n\nChose one from the below options...\n");

        printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");

        printf("\n Enter your choice \n");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

        {

            push();

            break;

        }

        case 2:

        {

            pop();

            break;

        }

        case 3:

        {

            display();

            break;

        }

        case 4:

        {

            printf("Exiting....");

            break;

        }

        default:

        {

            printf("Please Enter valid choice ");

        }

        };

    }

}

void push()

{

    int val;

    struct node \*ptr = (struct node \*)malloc(sizeof(struct node));

    if (ptr == NULL)

    {

        printf("not able to push the element");

    }

    else

    {

        printf("Enter the value");

        scanf("%d", &val);

        if (head == NULL)

        {

            ptr->val = val;

            ptr->next = NULL;

            head = ptr;

        }

        else

        {

            ptr->val = val;

            ptr->next = head;

            head = ptr;

        }

        printf("Item pushed");

    }

}

void pop()

{

    int item;

    struct node \*ptr;

    if (head == NULL)

    {

        printf("Underflow");

    }

    else

    {

        item = head->val;

        ptr = head;

        head = head->next;

        free(ptr);

        printf("Item popped");

    }

}

void display()

{

    int i;

    struct node \*ptr;

    ptr = head;

    if (ptr == NULL)

    {

        printf("Stack is empty\n");

    }

    else

    {

        printf("Printing Stack elements \n");

        while (ptr != NULL)

        {

            printf("%d\n", ptr->val);

            ptr = ptr->next;

        }

    }

}

**Output:-**

\*\*\*\*\*\*\*\*\*Stack operations using linked list\*\*\*\*\*\*\*\*\*

----------------------------------------------

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

 Enter your choice

1

Enter the value10

Item pushed

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

 Enter your choice

1

Enter the value20

Item pushed

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

 Enter your choice

130

Please Enter valid choice

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

 Enter your choice

3

Printing Stack elements

20

10

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

 Enter your choice

1

Enter the value30

Item pushed

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

 Enter your choice

3

Printing Stack elements

30

20

10

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

 Enter your choice

2

Item popped

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

 Enter your choice

2

Item popped

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

 Enter your choice

3

Printing Stack elements

10

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

 Enter your choice

4

Exiting....