m/1/22

Stack Using Linked List

Aim

Implement a stack using linked list with the operations. 1. Push elements to the queue. 2. pop elements from the queue 3. Display the queue after each operation.

Algorithm

Step 1: Start

Step 2: Create a structure node with data & link.

Step 3: Create a top pointer an intialize it to null.

Step 4: Declare push(data)

i) temp = top.

ii) Create a newnode newnode->data = data.
newnode->link = NULL.

iii) if (top == NULL) top = newnode.

iv) else new->link = top. top = newnode;

Step 5: Declare pop(). with int return type.

(i) if (top == NULL) print 'Stack empty return -1

else int val = top->data
top = top->link
return val.

Step 6: Declare a display() function.

(i) Create temp=top. until temp != NULL repeat below steps.

(ii) print(temp->data)

(iii) temp = temp->link

Step 7: Declare the main function.

// ~~Repeat steps to be~~ write a menu driven program to call these functions

Step 8: Stop

Result

The program is executed.

Code

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


struct node
{
    int data;
    struct node* link;
};


struct node* top=NULL;


void push(int data)
{
    struct node* temp=top;
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=data;
    newnode->link=NULL;
    if(top==NULL)
    {
        top=newnode;
    }else
    {
        newnode->link=top;
        top=newnode;
    }


}

int pop()
{
    if(top==NULL)
    {
        printf("Stack is empty");
        return -1;
    }else
    {
        int val=top->data;
        top=top->link;
        return val;
    }
}
```

```c
void display()
{
    printf("STACK : ");
    struct node* temp=top;
    while(temp!=NULL)
    {
        printf("%d ",temp->data);
        temp=temp->link;
    }
    printf("\n");
}









int main()
{



    while(1)
    {



        int choice;
        printf("1.Push\n2.Pop\n3.Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {


            case 1:
            {
                int temp;
                printf("Enter the value to push");
                scanf("%d",&temp);
                // add_end(temp);
                push(temp);
                display();
                break;
```

```
            }
            case 2:
            {
                int temp=pop();
                if(temp!=-1)
                {
                    printf("%d Popped\n",temp);
                }
                display();
                break;
            }
            case 3:
            {
                return 0;
            }
        }
    }
    return 0;
}
```

OUTPUT

1.Push

2.Pop

3.Exit

1

Enter the value to push12

STACK : 12

1.Push

2.Pop

3.Exit

1

Enter the value to push13

STACK : 13 12

1.Push

2.Pop

3.Exit

1

Enter the value to push14

STACK : 14 13 12

1.Push

2.Pop

3.Exit

1

Enter the value to push15

STACK : 15 14 13 12

1.Push

2.Pop

3.Exit

2

15 Popped

STACK : 14 13 12

1.Push

2.Pop

3.Exit

2

14 Popped

STACK : 13 12

1.Push

2.Pop

3.Exit

2

13 Popped

STACK : 12

1.Push

2.Pop

3.Exit

2

12 Popped

STACK :

1.Push

2.Pop

3.Exit

2

Stack is emptySTACK :

1.Push

2.Pop

3.Exit

2

Stack is emptySTACK :

1.Push

2.Pop

3.Exit

3