**TITLE 46**

Write a C program to implement the stack using linked list

**OBJECTIVE:**

By the end of this problem we will be able to implement STACK using linked list

**PROBLEM STATEMENT:**

In this program using linked list we implement STACK.

Once the program is free of errors the output is printed.

**ALGORITHM:**

START

Define variables: info, \*ptr, count, no, ch, e

INPUT: Read from the user

COMPUTATION: Computing the stack using linked list

DISPLAY: Displaying the output according the users choice

STOP

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int info;

struct node \*ptr;

}\*top,\*top1,\*temp;

int topelement();

void push(int data);

void pop();

void empty();

void display();

void destroy();

void stack\_count();

void create();

int count = 0;

void main()

{

int no, ch, e;

printf("\n 1 - Push");

printf("\n 2 - Pop");

printf("\n 3 - Top");

printf("\n 4 - Empty");

printf("\n 5 - Exit");

printf("\n 6 - Dipslay");

printf("\n 7 - Stack Count");

printf("\n 8 - Destroy stack");

create();

while (1)

{

printf("\n Enter choice : ");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("Enter data : ");

scanf("%d", &no);

push(no);

break;

case 2:

pop();

break;

case 3:

if (top == NULL)

printf("No elements in stack");

else

{

e = topelement();

printf("\n Top element : %d", e);

}

break;

case 4:

empty();

break;

case 5:

exit(0);

case 6:

display();

break;

case 7:

stack\_count();

break;

case 8:

destroy();

break;

default :

printf(" Wrong choice, Please enter correct choice ");

break;

}

}

}

/\* Create empty stack \*/

void create()

{

top = NULL;

}

/\* Count stack elements \*/

void stack\_count()

{

printf("\n No. of elements in stack : %d", count);

}

/\* Push data into stack \*/

void push(int data)

{

if (top == NULL)

{

top =(struct node \*)malloc(1\*sizeof(struct node));

top->ptr = NULL;

top->info = data;

}

else

{

temp =(struct node \*)malloc(1\*sizeof(struct node));

temp->ptr = top;

temp->info = data;

top = temp;

}

count++;

}

/\* Display stack elements \*/

void display()

{

top1 = top;

if (top1 == NULL)

{

printf("Stack is empty");

return;

}

while (top1 != NULL)

{

printf("%d ", top1->info);

top1 = top1->ptr;

}

}

/\* Pop Operation on stack \*/

void pop()

{

top1 = top;

if (top1 == NULL)

{

printf("\n Error : Trying to pop from empty stack");

return;

}

else

top1 = top1->ptr;

printf("\n Popped value : %d", top->info);

free(top);

top = top1;

count--;

}

/\* Return top element \*/

int topelement()

{

return(top->info);

}

/\* Check if stack is empty or not \*/

void empty()

{

if (top == NULL)

printf("\n Stack is empty");

else

printf("\n Stack is not empty with %d elements", count);

}

/\* Destroy entire stack \*/

void destroy()

{

top1 = top;

while (top1 != NULL)

{

top1 = top->ptr;

free(top);

top = top1;

top1 = top1->ptr;

}

free(top1);

top = NULL;

printf("\n All stack elements destroyed");

count = 0;

}

**CONCLUSION:**

The simulation of the above C program helped me undertsang how STACK is implemented using linked list.

**OUTPUT:**

1 – Push

2 – Pop

3 – Top

4 – Empty

5 – Exit

6 – Display

7 – Stack Count

8 – Destroy stack

Enter choice : 1

Enter data : 4

Enter choice : 1

Enter data : 8

Enter choice : 1

Enter data : 12

Enter choice : 1

Enter data : 16

Enter choice : 7

No. of elements in stack : 4

Enter choice : 6

16 12 8 4

Enter choice : 5