#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#define size 5

struct stack

{

int s[size];

int top;

};

typedef struct stack stack;

int stfull(stack \*st)

{

if ((\*st).top >= size - 1)

return 1;

else

return 0;

}

void push(stack \*st,int item)

{

(\*st).top++;

(\*st).s[(\*st).top] = item;

}

int stempty(stack \*st)

{

if ((\*st).top == -1)

return 1;

else

return 0;

}

int pop(stack \*st)

{

int item;

item = (\*st).s[(\*st).top];

(\*st).top--;

return (item);

}

void display(stack \*st)

{

int i;

if (stempty((st)))

printf("\nStack Is Empty!");

else {

for (i = (\*st).top; i >= 0; i--)

printf("\n%d", (\*st).s[i]);

}

}

void main()

{

int item, choice;

char ans='Y';

stack st;

st.top=-1;

printf("\n\tImplementation Of Stack");

do {

system("cls") ;

printf("0.Exit\n1.Push\n2.Pop\n3.Display\n");

printf("\nEnter Your Choice : ");

scanf("%d", &choice);

switch (choice) {

case 0:

ans='N';

break;

case 1:

printf("\nEnter The item to be pushed : ");

scanf("%d", &item);

if (stfull(&st))

printf("\nStack is Full!");

else

push(&st,item);

break;

case 2:

if (stempty(&st))

printf("\nEmpty stack!Underflow !!");

else {

item = pop(&st);

printf("\nThe popped element is %d", item);

}

break;

case 3:

display(&st);

break;

default:

printf("Wrong Input\nPress Enter to Continue");

fflush(stdin);

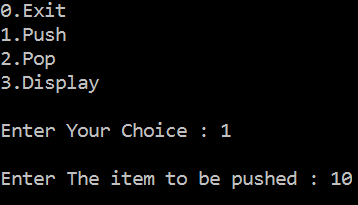
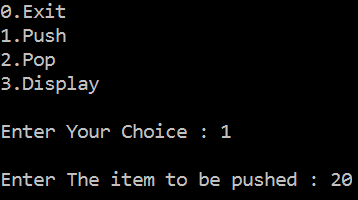
getchar();

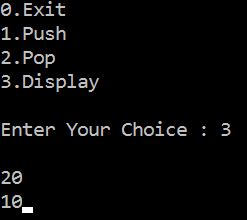
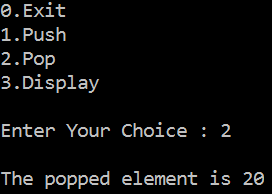
break;

}

} while (ans == 'Y');

}

#include <stdio.h>

#include <stdlib.h>

int count = 0;

struct node

{

int info;

struct node \*ptr;

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

typedef struct node \* stackptr;

/\* Create empty stack \*/

void create(stackptr \* top)

{

\*top = NULL;

}

/\* Count stack elements \*/

void stack\_count(stackptr \* top)

{

printf("\nNo. of elements in stack : %d\n", count);

}

/\* Push data into stack \*/

void push(stackptr \* top,int data)

{

stackptr temp;

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(stackptr \* top)

{

stackptr p = (\*top);

if (p == NULL)

{

printf("Stack is empty\n");

return;

}

while (p != NULL)

{

printf("%d\n", p->info);

p = p->ptr;

}

}

/\* Pop Operation on stack \*/

void pop(stackptr \* top)

{

stackptr top1;

top1 = (\*top);

if (top1 == NULL)

{

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

return;

}

else

top1 = top1->ptr;

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

free(\*top);

\*top = top1;

count--;

}

/\* Return top element \*/

int topelement(stackptr \* top)

{

return((\*top)->info);

}

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

void empty(stackptr \* top)

{

if ((\*top) == NULL)

printf("\nStack is empty");

else

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

}

/\* Destroy entire stack \*/

void destroy(stackptr \* top)

{

stackptr p = (\*top),q;

while (p != NULL)

{

q=p;

p=p->ptr;

free(q);

}

\*top = NULL;

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

count = 0;

}

void main()

{

stackptr top;

int no, ch, e;

create(&top);

while (1)

{

system("cls");

display(&top);

printf("\n1.Push");

printf("\n2.Pop");

printf("\n3.Top");

printf("\n4.Check Empty");

printf("\n5.Exit");

printf("\n6.Dipslay");

printf("\n7.Stack Count");

printf("\n8.Destroy stack\n");

printf("\nEnter choice : ");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("Enter data : ");

scanf("%d", &no);

push(&top,no);

break;

case 2:

pop(&top);

break;

case 3:

if (top == NULL)

printf("No elements in stack");

else

{

e = topelement(&top);

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

}

break;

case 4:

empty(&top);

break;

case 5:

exit(0);

case 6:

display(&top);

break;

case 7:

stack\_count(&top);

break;

case 8:

destroy(&top);

break;

default :

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

break;

}

fflush(stdin);

printf("\nPress Enter to Continue....");

getchar();

}

}

