**STACK OPERATIONS USING ARRAYS:**

#include<stdio.h>

#define size 5

int top=-1,s[5];

void push(int);

void pop();

void traverse();

void main()

{

int i,item,ch;

while(1)

{

printf("\n Enter your choice 1.push 2.pop 3.traverse");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("\n Enter the item ");

scanf("%d",&item);

push(item);

break;

case 2: pop();

break;

case 3: traverse();

break;

default: exit(0);

}

}

}

void push(int item)

{

if(top>=size-1)

{

printf("\n stack is full ");

}

else

{

top=top+1;

s[top]=item;

}

}

void pop()

{

if(top < 0)

{

printf("\n Stack is empty");

}

else

{

printf("Poped item is %d",s[top]);

top=top-1;

}

}

void traverse()

{

int i;

if(top < 0)

{

printf("\n Stack is empty");

}

else

{

printf("\n Items of stack are ");

for(i=0;i<=top;i++)

{

printf("%d\t",s[i]);

}

}

}

**OUTPUT:**

Enter your choice 1.push 2.pop 3.traverse3

Stack is empty

Enter your choice 1.push 2.pop 3.traverse1

Enter the item 10

Enter your choice 1.push 2.pop 3.traverse1

Enter the item 20

Enter your choice 1.push 2.pop 3.traverse3

Items of stack are 10 20

Enter your choice 1.push 2.pop 3.traverse1

Enter the item 30

Enter your choice 1.push 2.pop 3.traverse1

Enter the item 40

Enter your choice 1.push 2.pop 3.traverse1

Enter the item 50

Enter your choice 1.push 2.pop 3.traverse1

Enter the item 60

stack is full

Enter your choice 1.push 2.pop 3.traverse3

Items of stack are 10 20 30 40 50

Enter your choice 1.push 2.pop 3.traverse2

Poped item is 50

Enter your choice 1.push 2.pop 3.traverse2

Poped item is 40

Enter your choice 1.push 2.pop 3.traverse2

Poped item is 30

Enter your choice 1.push 2.pop 3.traverse3

Items of stack are 10 20

Enter your choice 1.push 2.pop 3.traverse2

Poped item is 20

Enter your choice 1.push 2.pop 3.traverse2

Poped item is 10

Enter your choice 1.push 2.pop 3.traverse2

Stack is empty

Enter your choice 1.push 2.pop 3.traverse

**STACK OPERATIONS USING LINKED LIST:**

**Program:**

#include<stdio.h>

void push();

void pop();

void traverse();

struct node

{

int data;

struct node\*link;

}\*header,\*new,\*ptr,\*top;

int item,ch;

void main()

{

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

header->link=NULL;

while(1)

{

printf("\n1.push\t2.pop\t3.Traverse\n");

printf("enter the choice :");

scanf("%d",&ch);

switch(ch)

{

case 1:push();

break;

case 2:pop();

break;

case 3:traverse();

break;

default :

exit(0);

}

}

getch();

}

void push()

{

top=header->link;

printf("\n Enter data to push :");

scanf("%d",&item);

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

header->link=new;

new->link=top;

top=new;

new->data=item;

}

void pop()

{

top=header->link;

if(header->link==NULL)

{

printf("\n Stac is empty");

}

else

{

ptr=top->link;

item=top->data;

header->link=ptr;

top=ptr;

printf("\n Deleted item is %d",item);

}

}

void traverse()

{

ptr=header;

if(header->link==NULL)

{

printf("\n Stack is empty");

}

else

{

printf("\n Items present in stack are ");

while(ptr->link!=NULL)

{

ptr=ptr->link;

printf("%d \t",ptr->data);

}

}

}

**OUT PUT:**

1.push 2.pop 3.Traverse

enter the choice :3

Stack is empty

1.push 2.pop 3.Traverse

enter the choice :1

Enter data to push :10

1.push 2.pop 3.Traverse

enter the choice :1

Enter data to push :20

1.push 2.pop 3.Traverse

enter the choice :3

Items present in stack are 20 10

1.push 2.pop 3.Traverse

enter the choice :2

Deleted item is 20

1.push 2.pop 3.Traverse

enter the choice :2

Deleted item is 10

1.push 2.pop 3.Traverse

enter the choice :2

Stac is empty

1.push 2.pop 3.Traverse

enter the choice :3

Stack is empty

1.push 2.pop 3.Traverse

enter the choice :

**EVALUATION OF POSTFIX EXPRESSION:**

#include<stdio.h>

#include<ctype.h>

#include<string.h>

#include<math.h>

char s[20];

int top=0;

void main()

{

char postfix[20],symb;

int i=0,l,op1,op2,res;

printf("\n Enter postfix expression");

scanf("%s",postfix);

l=strlen(postfix);

for(i=0;i<l;i++)

{

if(isdigit(postfix[i]))

{

top=top+1;

s[top]=postfix[i]-48;

}

else

{

op2=s[top];

top=top-1;

op1=s[top];

top=top-1;

switch(postfix[i])

{

case '+':

res=op1+op2;

break;

case '-':

res=op1-op2;

break;

case '\*':

res=op1\*op2;

break;

case '/':

res=op1/op2;

break;

case '%':

res=op1%op2;

break;

case '$':

res=pow(op1,op2);

break;

}

top=top+1;

s[top]=res;

}

}

printf("\nresult postfix expression is %d",s[top]);

}

**OUTPUT:1**

Enter infix expression24+

result postfix expression is 6

**OUTPUT:2**

Enter infix expression4-2

result postfix expression is 2

**OUTPUT:3**

Enter infix expression234+\*

result postfix expression is 14