1. **Stack using Structure.**

#include <stdio.h>

#include <stdlib.h>

typedef struct STACK{

int \*data;

int top;

int stack\_length;

}Stack;

Stack\* init(int size){

Stack \*sp = (Stack\*)malloc(sizeof(Stack));

if(sp == NULL){

printf("Error unable to allocate memory !!");

exit(1);

}

sp->top = -1;

sp->stack\_length = size;

sp->data = (int\*) malloc(sizeof(int) \* size);

return sp;

}

void push(Stack \*s,int value){

if(s->top != s->stack\_length-1){

s->data[++s->top] = value;

}

else{

printf("Overflow condition !!");

}}

void pop(Stack \*s){

if(s->top != -1){

printf("Popped element is %d\n",s->data[s->top]);

s->top -= 1;

}

else{

printf("Stack is Empty !!");

}}

void display(Stack \*s){

int i;

if (s->top != -1){

printf("Elements in stack are: \n");

for(i = s->top; i >= 0; i --){

printf(" %d \n",s->data[i]);

}}}

void peep(Stack \*a){

if(a->top != -1){

printf("Topmost element of stack is: %d\n",a->data[a->top]);

}}

void main(){

Stack \*a;

a = init(4);

push(a,43);

push(a,44);

push(a,45);

push(a,46);

display(a);

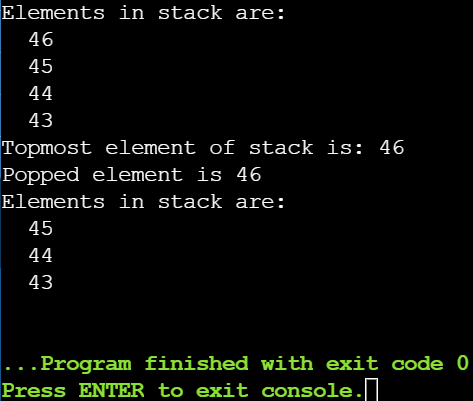
peep(a);

pop(a);

display(a);

}

**OUTPUT:**



1. **Infix Expression to postfix.**

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <string.h>

typedef struct {

char \*data;

int top;

int size;

}Stack;

Stack\* initialize(int size){

Stack \*s = (Stack\*)malloc(sizeof(Stack));

s->data = (char\*)malloc(sizeof(char) \* size);

s->top = -1;

s->size = size;

return s;

}

char pop(Stack \*s){

if(s->top != -1){

char element = s->data[s->top];

s->top --;

return element;

}}

char push(Stack \*s,char ele){

if(s->top < s->size-1){

s->data[++s->top] = ele;

}}

void display(Stack \*s){

int i;

for(i = 0; i <= s->top; i ++){

printf("%c",s->data[i]);

}}

char peep(Stack \*s){

if(s->top != -1){

return s->data[s->top];

}}

int precedence(char a){

switch(a){

case '^':

return 5;

break;

case '/':

return 4;

break;

case '\*':

return 3;

break;

case '+':

return 2;

break;

case '-':

return 1;

break;

default:

return 0;

break;

}}

int main(){

Stack \*op\_stack, \*out\_stack;

op\_stack = initialize(15);

out\_stack = initialize(15);

char infix\_expression[20];

char character,popped;

int i;

printf("Enter infix expression:");

scanf("%s",infix\_expression);

printf("Infix Expression is: %s\n",infix\_expression);

while(infix\_expression[i] != '\0'){

character = infix\_expression[i];

if(character == '('){

push(op\_stack,character);

}

else if(character == ')'){

while(peep(op\_stack) != '('){

popped = pop(op\_stack);

push(out\_stack,popped);

}

pop(op\_stack);

}

else if(character == '^'||character == '/'||character == '\*'||character == '+'||character == '-'){

while(precedence(peep(op\_stack)) > precedence(character) && op\_stack->top != -1){

popped = pop(op\_stack);

push(out\_stack,popped);

}

push(op\_stack,character);

}

else{

push(out\_stack,character);

}

i++;

}

if(op\_stack->top != -1){

while(op\_stack->top != -1){

popped = pop(op\_stack);

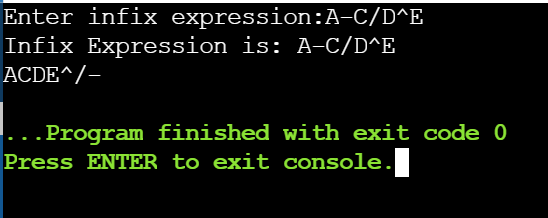
push(out\_stack,popped);

}}

display(out\_stack);

return 0;

}

**OUTPUT:**

1. **Infix Expression to Prefix Expression.**

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <string.h>

typedef struct {

char \*data;

int top;

int size;

}Stack;

Stack\* initialize(int size){

Stack \*s = (Stack\*)malloc(sizeof(Stack));

s->data = (char\*)malloc(sizeof(char) \* size);

s->top = -1;

s->size = size;

return s;

}

char pop(Stack \*s){

if(s->top != -1){

char element = s->data[s->top];

s->top --;

return element;

}}

char push(Stack \*s,char ele){

if(s->top < s->size-1){

s->data[++s->top] = ele;

}}

void display(Stack \*s, int j){

if(j == 1){

int i;

for(i = 0; i <= s->top; i ++){

printf("%c",s->data[i]);

}}

else if(j == 0){

int i;

for(i = s->top; i >= 0; i --){

printf("%c",s->data[i]);

}}}

char peep(Stack \*s){

if(s->top != -1){

return s->data[s->top];

}}

int precedence(char a){

switch(a){

case '^':

return 5;

break;

case '/':

return 4;

break;

case '\*':

return 3;

break;

case '+':

return 2;

break;

case '-':

return 1;

break;

default:

return 0;

break;

}}

int main(){

Stack \*op\_stack, \*out\_stack;

op\_stack = initialize(15);

out\_stack = initialize(15);

char infix\_expression[20];

char character,popped;

int i;

printf("Enter infix expression:");

scanf("%s",infix\_expression);

printf("Infix Expression is: %s\n",infix\_expression);

i = strlen(infix\_expression);

while(i >= 0){

character = infix\_expression[i];

if(character == ')'){

push(op\_stack,character);

}

else if(character == '('){

while(peep(op\_stack) != ')'){

popped = pop(op\_stack);

push(out\_stack,popped);

}

pop(op\_stack);

}

else if(character == '^'||character == '/'||character == '\*'||character == '+'||character == '-'){

while(precedence(peep(op\_stack)) > precedence(character) && op\_stack->top != -1){

popped = pop(op\_stack);

push(out\_stack,popped);

}

push(op\_stack,character);

}

else{

push(out\_stack,character);

}

i--; }

if(op\_stack->top != -1){

while(op\_stack->top != -1){

popped = pop(op\_stack);

push(out\_stack,popped);

}

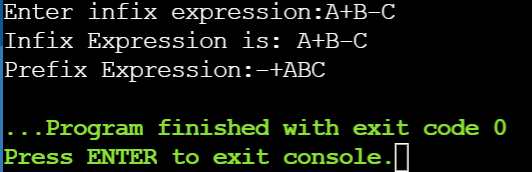
i--; }

printf(“Prefix Expression: “);

display(out\_stack,0);

return 0;

}

**OUTPUT:**

1. **Evaluate prefix expression**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <math.h>

typedef struct{

int \*data;

int top;

int size;

}Stack;

Stack\* init(int size){

Stack \*s = (Stack\*)malloc(sizeof(Stack));

s->data = (int\*)malloc(sizeof(int) \* size);

s->top = -1;

s->size = size;

return s;

}

void push(Stack \*sp,int element){

if(sp->top > sp->size - 1){

printf("Stack is full");

}

else{

sp->top ++;

sp->data[sp->top] = element;

}

}

int pop(Stack \*sp){

if(sp->top != -1){

int ele;

ele = sp->data[sp->top];

sp->top --;

return ele;

}

}

void display(Stack \*s){

int i;

printf("\nStack is");

for(i = s->top; i >= 0; i --){

printf(" %d \n",s->data[i]);

}}

int main(){

Stack \*s;

s = init(10);

char a[10];

printf("Enter prefix expression: ");

scanf("%s",a);

int i = 0,ans,c,b;

char ele;

i = strlen(a) - 1;

while(i >= 0){

ele = a[i];

if(a[i] >= 48 && a[i] < 58){

push(s,ele - '0');

}

else{

b= pop(s);

c = pop(s);

switch(ele){

case '^':

ans = pow(b,c);

break;

case '+':

ans = b + c;

break;

case '-':

ans = b - c; break;

case '\*':

ans = b \* c;

break;

case '/':

ans = b / c;

break;

default:

break;

}

push(s,ans);

}

display(s);

printf("\n");

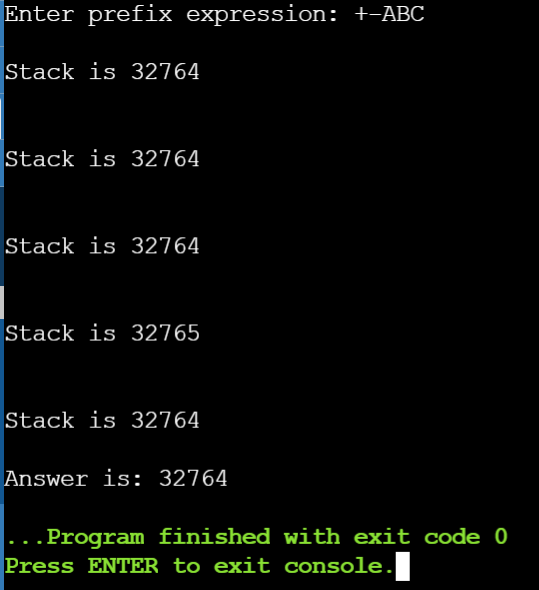
i --;

}

printf("Answer is: %d",pop(s));

return 0;

}

**OUTPUT:**

1. **Evaluate postfix expression**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <math.h>

typedef struct{

int \*data;

int top;

int size;

}Stack;

Stack\* init(int size){

Stack \*s = (Stack\*)malloc(sizeof(Stack));

s->data = (int\*)malloc(sizeof(int) \* size);

s->top = -1;

s->size = size;

return s;

}

void push(Stack \*sp,int element){

if(sp->top > sp->size - 1){

printf("Stack is full");

}

else{

sp->top ++;

sp->data[sp->top] = element;

}

}

int pop(Stack \*sp){

if(sp->top != -1){

int ele;

ele = sp->data[sp->top];

sp->top --;

return ele;

}

}

void display(Stack \*s){

int i;

printf("\nStack is: ");

for(i = s->top; i >= 0; i --){

printf(" %d \n",s->data[i]);

}

}

int main(){

Stack \*s;

s = init(10);

char a[10];

printf("Enter postfix expression: ");

scanf("%s",a);

int i = 0,ans,c,b;

char ele;

while(i < strlen(a)){

ele = a[i];

if(a[i] >= 48 && a[i] < 58){

push(s,ele - '0');

}

else{

b= pop(s);

c = pop(s);

switch(ele){

case '^':

ans = pow(b,c);

break;

case '+':

ans = b + c;

break;

case '-':

ans = b - c;

break;

case '\*':

ans = b \* c;

break;

case '/':

ans = b / c;

break;

default:

break;

}

push(s,ans);

}

display(s);

printf("\n");

i ++;

}

printf("Answer is: %d",pop(s));

return 0;

}

**OUTPUT:**