**Q. wap to convert infix to postfix notation, and evaluate it if applicable**

**Source Code :**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

#define MAXSIZE 50

int tos = -1, i = 0;

char in[100], post[100], stack[100];

int isEmpty()

{

int status;

if (tos == -1){

status = 1; //true

}

else{

status = 0; //false

}

return status;

}

int isFull()

{

int status;

if (tos == MAXSIZE - 1){

status = 1; //true

}

else{

status = 0; //false

}

return status;

}

void push(char n)

{

if (isFull()){

printf("\nSTACK FULL !");

}

else{

tos ++;

stack[tos] = n;

}

}

char pop()

{

char character;

if (isEmpty()){

printf("\nSTACK EMPTY !");

}

else{

character = stack[tos];

tos--;

}

return character;

}

char peek()

{

char top;

if (isEmpty()){

printf("\nSTACK EMPTY !");

}

else{

top = stack[tos];

}

return top;

}

int isLowerPrecedence(char scanned)

{

char top = peek();

switch(scanned)

{

case '/':

case '\*':

switch(top)

{

case '/':

case '\*':

return 1;

break;

default: return 0;

}

break;

default: return 1;

}

}

float Evaluate(char \*ptr)

{

float n, n1, n2;

char c;

while(\*ptr != '\0')

{

switch(\*ptr)

{

case '+':

case '-':

case '\*':

case '/':

n1 = pop() - 48.0;

n2 = pop() - 48.0;

switch(\*ptr)

{

case '+':

n = n2 + n1;

break;

case '-':

n = n2 - n1;

break;

case '\*':

n = n2 \* n1;

break;

case '/':

n = n2 / n1;

break;

}

c = n + 48.0;

push(c);

break;

case '0':

case '1':

case '2':

case '3':

case '4':

case '5':

case '6':

case '7':

case '8':

case '9':

push(\*ptr);

break;

default : printf("\nUnable to Evaluate\n");

exit(1);

}

ptr++;

}

return (pop() - 48);

}

void writePost(char c)

{

post[i] = c;

//printf("\n%c\n", c);

post[i+1] = '\0';

i++;

}

int main()

{

char \*pc, c;

int loop = 0, ascii = 0;

float result = 0;

printf("\nEnter Infix Expression : ");

scanf("%[^\n]s", in);

printf("\nThe input Infix Expression is : ");

puts(in);

pc = in;

//printf("\n%c\n", \*pc);

while(\*pc != '\0')

{

//printf("\n%c\n", \*pc);

switch(\*pc)

{

case '(':

//printf("\n%c\n", \*pc);

push(\*pc);

break;

case ')':

//printf("\n%c\n", \*pc);

loop = 1;

while (loop == 1)

{

c = pop();

if (c == '('){ loop = 0; }

else {

writePost(c);

}

}

break;

case '/':

case '\*':

case '+':

case '-':

//printf("\n%c\n", \*pc);

if( isEmpty() == 1 || peek() == '(' ){

push(\*pc);

}

else{

if (isLowerPrecedence(\*pc)){

c = pop();

writePost(c);

}

push(\*pc);

}

break;

default:

ascii = \*pc; //implicit typecast

if ((65 <= ascii && ascii <= 90) || (97 <= ascii && ascii <= 122) || (48 <= ascii && ascii <= 57))

{

//printf("\n%c %d\n", \*pc, ascii);

writePost(\*pc);

}

}

pc++;

}

while(isEmpty() == 0)

{

c = pop();

writePost(c);

}

printf("\nThe Postflix Expression is : %s", post);

getch();

result = Evaluate(post);

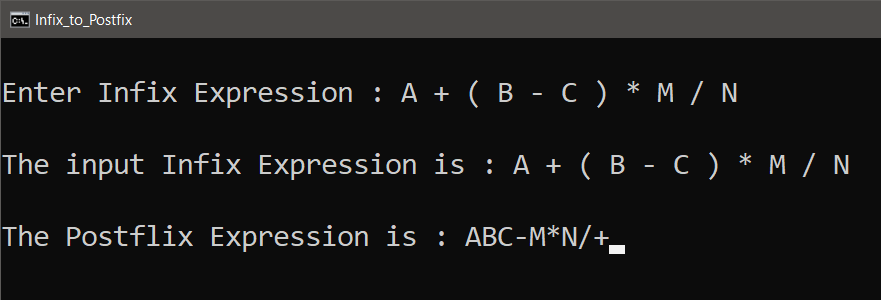
printf("\nAns : %f", result);

return 0;

}

**Output:**

Conversion of Generic Infix to Postfix Notation.



Conversion of Infix to Postfix Notation and evaluation of the postfix expression.

