TITLE 36

Write a C program to convert the given infix expression to post-fix expression using STACK

OBJECTIVE:

To convert infix to postfix

PROBLEM STATEMENT:

In this problem with the help of STACK we will convert an infix expression to post-fix expression.

ALGORITHM:

START

Define variables: top

INPUT: Read from the keyboard

COMPUTATION: Computing the infix to post-fix expression

DISPLAY: Displaying the converted post-fix expression

STOP

PROGRAM:

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define max 1000

typedef struct Stack{

int top;

int arr[max];

}stack;

char pop(stack \*s){

if (s->top != -1)

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

return '#';

}

void push(stack \*s, char op){

s->arr[++s->top] = op;

}

int Prec(char ch){

switch (ch) {

case '+': return 1;

case '-': return 1;

case '\*': return 2;

case '/': return 2;

case '^': return 3;

}

return -1;

}

void in2post(char\* exp) {

int i, k;

stack s; s.top = -1;

for (i = 0, k = -1; exp[i]; ++i) {

if ((exp[i] >= 'a' && exp[i] <= 'z') || (exp[i] >= 'A' && exp[i] <= 'Z'))

exp[++k] = exp[i];

else if (exp[i] == '(')

push(&s, exp[i]);

else if (exp[i] == ')'){

while (s.top != -1 && s.arr[s.top] != '(')

exp[++k] = pop(&s);

if (s.top == -1 && s.arr[s.top] != '(')

printf("Invalid expression\n");

else

pop(&s);

}

//if is a operation

else {

while (s.top != -1 && Prec(exp[i]) <= Prec(s.arr[s.top]))

exp[++k] = pop(&s);

push(&s, exp[i]);

}

}

exp[++k] = pop(&s);

exp[++k] = '\0';

printf( "%s", exp );

}

int main() {

char exp[] = "a+b\*(c^d)+(e-f/g)\*c+d";

printf( "Infix- expression: %s\n", exp );

printf("%s","postfix expression:" );

in2post(exp);

return 0;

}

CONCLUSION:

This program helps us understand how to convert an infix expression to post-fix expression with the usage of STACK.

OUTPUT:

abcd^e-fgh\*+^\*+i-