Program no:15

Write a C Program to implement the operator precedence parsing.

Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX 100

char stack[MAX];

int top = -1;

void push(char c) {

if (top < MAX - 1) {

stack[++top] = c;

}

}

char pop() {

if (top >= 0) {

return stack[top--];

}

return '\0';

}

int precedence(char c) {

switch (c) {

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

case '^':

return 3;

default:

return 0;

}

}

void infixToPostfix(char\* infix, char\* postfix) {

int i = 0, j = 0;

char symbol;

push('(');

strcat(infix, ")");

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

if (symbol == '(') {

push(symbol);

} else if (isalnum(symbol)) {

postfix[j++] = symbol;

} else if (symbol == ')') {

while (stack[top] != '(') {

postfix[j++] = pop();

}

pop();

} else {

while (precedence(stack[top]) >= precedence(symbol)) {

postfix[j++] = pop();

}

push(symbol);

}

}

postfix[j] = '\0';

}

int main() {

char infix[MAX], postfix[MAX];

printf("Enter infix expression: ");

scanf("%s", infix);

infixToPostfix(infix, postfix);

printf("Postfix expression: %s\n", postfix);

return 0;

}

Output:

Enter infix expression: a\*b+c

Postfix expression: ab\*c+

Program no:16

#include <stdio.h>

#include <string.h>

int t = 1;

void genTAC(char op, char a, char b)

{

printf("t%d = %c %c %c\n", t++, a, op, b);

}

void process(char expr[])

{

char s[100]; int top = -1;

for (int i = 0; expr[i] != '\0'; i++)

{

if (strchr("+-\*/", expr[i]))

{

genTAC(expr[i], s[top - 1], s[top]);

s[--top] = 't' + t - 2;

} else s[++top] = expr[i];

}

}

int main()

{

char expr[100];

printf("Enter expr: ");

scanf("%s", expr);

process(expr);

return 0;

}