1. Infix to postfix

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define MAX 100

char stack[MAX];

char infix[MAX];

char postfix[MAX];

int top=-1;

void push(char);

char pop();

int isEmpty();

void inToPost();

void print();

int precedence(char);

int main()

{

printf("enter infix expression: ");

gets(infix);

inToPost();

print();

return 0;

}

void inToPost()

{

int i,j=0;

char symbol,next;

for(i=0;i<strlen(infix);i++)

{

symbol=infix[i];

switch(symbol)

{

case '(':

push(symbol);

break;

case ')':

while((next=pop())!='(')

postfix[j++]=next;

break;

case '+':

case '-':

case '\*':

case '/':

case '^':

while (!isEmpty() && precedence(stack[top]) >= precedence(symbol))

postfix[j++] = pop();

push(symbol);

break;

default:

postfix[j++] = symbol;

}

}

while (!isEmpty())

postfix[j++] = pop();

postfix[j] = '\0';

}

int precedence(char symbol)

{

switch (symbol)

{

case '^':

return 3;

case '/':

case '\*':

return 2;

case '+':

case '-':

return 1;

default:

return 0;

}

}

void print()

{

int i = 0;

printf("The equivalent postfix expression is: ");

while (postfix[i])

{

printf("%c ", postfix[i++]);

}

printf("\n");

}

void push(char c)

{

if(top==MAX-1)

{

printf("stack overflow");

return;

}

top++;

stack[top]=c;

}

char pop()

{

char c;

if(top==-1)

{

printf("stack underflow");

exit(1);

}

c=stack[top];

top=top-1;

return c;

}

int isEmpty()

{

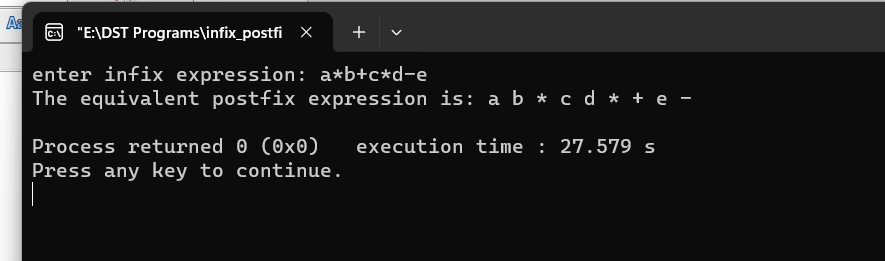
if(top==-1)

return 1;

else

return 0;

}



1. Postfix evaluation

#include<stdio.h>

int stack[20];

int top = -1;

void push(int x)

{

stack[++top] = x;

}

int pop()

{

return stack[top--];

}

int main()

{

char exp[20];

char \*e;

int n1,n2,n3,num;

printf("Enter the expression :: ");

scanf("%s",exp);

e = exp;

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

{

if(isdigit(\*e))

{

num = \*e - 48;

push(num);

}

else

{

n1 = pop();

n2 = pop();

switch(\*e)

{

case '+':

{

n3 = n1 + n2;

break;

}

case '-':

{

n3 = n2 - n1;

break;

}

case '\*':

{

n3 = n1 \* n2;

break;

}

case '/':

{

n3 = n2 / n1;

break;

}

}

push(n3);

}

e++;

}

printf("\nThe result of expression %s = %d\n\n",exp,pop());

return 0;

}

