AIM:-Program to convert the infix to the postfix.

Algorithm

Step 1: Scan the infix expression from left to right.

Step 2: If the second character is an operand, append it with final infix to the postfix.

Step 3: Else, If the precedence order of the scanned operator is greater then the precedence order of the operator in the stack.

Step 4: Else, pop all the operators from the stack which are greater then or equal to in precedence than that of the scanned operator. After doing that push the scanned operator to the stack.

Step 5: If the scanned character is a ‘(‘or’)’, push it to the stack.

Step 6: Repeat the steps until infix expression is scanned.

Step 7: Print the output.

Step 8: Pop and output from the stack until it is not empty.

#include<stdio.h>//standard input output headerfile//

#include<ctype.h>

char stack[20];//declaring the stack size//

int top=-1;//declaring initial value as top is -1//

void push(char x)//push function//

{

stack[++top]=x;

}

char pop()//pop function//

{

if(top==-10';

return -1;

else

return stack[top--1];

}

int priority(char X)//priority function//

{

if(x=='(')//if the value of x is (then return to 0)//

return 0;

if(x=='+'||x=='-')//if the value of x is+ or - then return to 1//

return 1;

if(x=='\*'||x=='/')// if the value of x is \* or / then return to 2;

retrun 2;

}

int main()//main function//

{

char \*e,x;

char exp{20];

printf("enter the expression \n");//displaying the expression//

scanf("%s",exp);

e=exp;// initiliazing the value of e to the exp//

while(\*e!+'\0')//check the condition//

{

if(isallnum(\*e)

printf("%C",\*E);

else if(\*e=='(')

push(\*e);//check the condition of else if and pushes the element//

else if(\*e==')')

{

while((x=pop())!+'(')

printf("%c",x);

}

else

{

while(priority(stack[top])=priority(\*e))//check the condition//

printf("%c" ,pop());

push(\*e);

}

prinft("\n");

return 0;//terminating program//

}

Screenshot:

