**AIM:** To write a c program to perform infix to postfix conversion using stack.

**ALGORITHM:**

**Step 1:** Start.

**Step 2:** Make an empty stack.

**Step 3:** Read the infix expression one character at a time until it encounters end of expression.

**Step 4:** If the character is an operand, place it onto the output.

**Step 5:** If the character is an operator, push it onto the stack. If the stack operator has a

higher or equal priority than input operator, then pop that operator from the stack

and place it onto the output.

**Step 6:**  If the character is a left parenthesis, push it onto the stack.

**Step 7:** If the character is a right parenthesis, pop all the operators from the stack till it

encounters left parenthesis, discard both the parenthesis in the output.

**Step 8:**  Stop.

**PROGRAM:**

#include <stdio.h>

#include <string.h>

#define MAX 20

int Stack[MAX], top = -1;

char expr[MAX], post[MAX];

void Push(char sym);

char Pop();

char Top();

int Priority(char sym);

int main()

{

int i;

printf("Enter the infix expression : ");

scanf(“%d”,&i);

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

{

if(expr[i] >= 'a' && expr[i] <= 'z')

printf("%c", expr[i]);

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

Push(expr[i]);

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

{

while(Top() != '(')

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

Pop();

}

else

{

while(Priority(expr[i])<=Priority(Top()) && top!=-1)

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

Push(expr[i]);

}

}

for(i = top; i >= 0; i--)

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

return 0;

}

void Push(char sym)

{

top = top + 1;

Stack[top] = sym;

}

char Pop()

{

char e;

e = Stack[top];

top = top - 1;

return e;

}

char Top()

{

return Stack[top];;

}

int Priority(char sym)

{

int p = 0;

switch(sym)

{

case '(':

p = 0;

break;

case '+':

case '-':

p = 1;

break;

case '\*':

case '/':

case '%':

p = 2;

break;

case '^':

p = 3;

break;

}

return p;

}

**OUTPUT:**

Enter the infix expression : a/b^c+d\*e-f\*g

abc^/de\*+fg\*-

**RESULT:**

Hence the program to convert infix to postfix expression using stack is implemented.