INFIX TO POSTFIX

#include<iostream>

#include<cstring>

#include<stack>

using namespace std;

// get weight of operators as per precedence

// higher weight given to operators with higher precedence

// for non operators, return 0

int getWeight(char ch) {

switch (ch) {

case '/':

case '\*': return 2;

case '+':

case '-': return 1;

default : return 0;

}

}

// convert infix expression to postfix using a stack

void infix2postfix(char infix[], char postfix[], int size) {

stack<char> s;

int weight;

int z = 0;

int j = 0;

char ch;

// iterate over the infix expression

while (z < size) {

ch = infix[z];

if (ch == '(') {

// simply push the opening parenthesis

s.push(ch);

z++;

continue;

}

if (ch == ')') {

// if we see a closing parenthesis,

// pop of all the elements and append it to

// the postfix expression till we encounter

// a opening parenthesis

while (!s.empty() && s.top() != '(') {

postfix[j++] = s.top();

s.pop();

}

// pop off the opening parenthesis also

if (!s.empty()) {

s.pop();

}

z++;

continue;

}

weight = getWeight(ch);

if (weight == 0) {

// we saw an operand

// simply append it to postfix expression

postfix[j++] = ch;

}

else {

// we saw an operator

if (s.empty()) {

// simply push the operator onto stack if

// stack is empty

s.push(ch);

}

else {

// pop of all the operators from the stack and

// append it to the postfix expression till we

// see an operator with a lower precedence that

// the current operator

while (!s.empty() && s.top() != '(' &&

weight <= getWeight(s.top())) {

postfix[j++] = s.top();

s.pop();

}

// push the current operator onto stack

s.push(ch);

}

}

z++;

}

// pop of the remaining operators present in the stack

// and append it to postfix expression

while (!s.empty()) {

postfix[j++] = s.top();

s.pop();

}

postfix[j] = 0; // null terminate the postfix expression

}

// main

int main() {

char infix[100];//"A\*(B+C)/D";

cout<<"\nENter Infix Operation:";

cin>>infix;

int size = strlen(infix);

char postfix[size];

infix2postfix(infix,postfix,size);

cout<<"\nInfix Expression :: "<<infix;

cout<<"\nPostfix Expression :: "<<postfix;

cout<<endl;

return 0;

}