INFIX TO POSTFIX.cpp

#include "Infix2Postfix.h"

/\*

INFIX TO POSTFIX

Only '+' , '-' , '\*' and '/' are aloud

Operator and operands must be single character.

Input Infix expression must be in a desired format.

\*/

// Reads from Infix to output a postfix expression

string InfixToPostfix(string input)

{

// Declaring stack from standard template

stack<char> Stack;

string postfix = ""; // Initialize postfix

for (int i = 0; i< input.length(); i++) {

// Scan each character

if (input[i] == ' ' || input[i] == ',') continue;

// If operator, pop two elements from stack, perform operation and push the result back.

else if (IsOperator(input[i]))

{

while (!Stack.empty() && Stack.top() != '(' && HasGreaterPriority(Stack.top(), input[i]))

{

postfix += Stack.top();

Stack.pop();

}

Stack.push(input[i]);

}

else if (IsOperand(input[i]))

{

postfix += input[i];

}

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

{

Stack.push(input[i]);

}

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

{

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

postfix += Stack.top();

Stack.pop();

}

Stack.pop();

}

}

while (!Stack.empty()) {

postfix += Stack.top();

Stack.pop();

}

return postfix;

}

// Check if an operator is right value or not.

int IsRightValue(char op)

{

if

(op == '$') return true;

return false;

}

// Operator symbol or not

bool IsOperator(char C)

{

if

(C == '+' || C == '-' || C == '\*' || C == '/')

return true;

return false;

}

// Number or letter?

bool IsOperand(char C)

{

if (C >= '0' && C <= '9') return true;

if (C >= 'a' && C <= 'z') return true;

if (C >= 'A' && C <= 'Z') return true;

return false;

}

int GetOperatorPriority(char op)

{

int Priority = -1;

switch (op)

{

case '+':

case '-':

Priority = 2;

case '\*':

case '/':

Priority = 1;

}

return Priority;

}

int HasGreaterPriority(char op1, char op2)

{

int operator1Priority = GetOperatorPriority(op1);

int operator2Priority = GetOperatorPriority(op2);

if (operator1Priority == operator2Priority)

{

//if ops are equal, return true if they are left value

//if op is left value, left will be given priority.

if (IsRightValue(op1)) return false;

else return true;

}

return operator1Priority < operator2Priority ? true : false;

}