Project 3. Implementation of the Expression Type

[Problem Description]

There is a correspondence between the expression and the binary tree. Your task is to write a program that realizes the functions of the arithmetic expression based on the binary tree.

[Requirement]

Supposing that elements of an expression can be variables (a~z), constants (0~9) and binary operators (+, -, *, /, ^). Please implement the following functions.

(1) ReadExpr(E) -- Input character sequence as prefix expression E.

(2) WriteExpr(E) -- Output E as infix expression.

(3) Assign(V, c) -- Assign value c to variable V, which has a default value of 0.

(4) Value(E) -- Calculate the arithmetic result of expression E.

(5) CompoundExpr(P, E1, E2) -- Return a compound expression (E1)P(E2). P is an operator.

(6) Diff(E, V) -- Find the partial derivative of function E with respect to variable V.

(7) Add functions which allow trigonometric functions to be included in the expressions (optional)

(8) MergeConst(E) -- Merge all the constants within the expression E. For instance,
E = (2+3-a)*(b+3*4) would return E = (5-a)*(b+12).

[Test Cases]

(1) Input 0; a; -91; +a*bc; +*5^x2*8x; +++*3^ *3*2^ x2x6 and output to test function ReadExpr and WriteExpr.

(2) Input the expression, assign values to variables, and calculate them using Value(E) to test the functions.

[Hints]

(1) Identify the operators and operands (integers) while reading the character sequences of the expressions.

(2) Turn character form into integer form while identifying the operands.

(3) Calculate the expression following the post-order traversal.

(4) Use brackets while outputting the expression using in-order traversal.

[Grading]

Implementation: 50% Interface: 30% Coding Style: 20%

Notice: This project will be checked on the experimental lesson in the 13rd week (2016.11.22).