Prefix/Infix/Postfix Notation

One commonly writes mathematical expressions, such as $A - \frac{B}{C+D}$ in *infix* notation: *A*-*B*/(*C*+*D*). In this example, one must first evaluate *C*+*D* (call the result *X*), then *B*/*X* (call the result *Y*), and finally *A*-*Y*. The order of evaluation is not simply "go from left to right, evaluating on the fly." Rather, evaluation is dependent on the precedence of the operators and the location of parentheses. Two alternative formats, *prefix* and *postfix*, have been developed to make evaluation more mechanical (and hence, solvable by a simple computer program).

Prefix notation places each operator before its operands, and postfix places each operator after its operands. The example above becomes –*A*/*B*+*CD* in prefix notation, and *ABCD*+/- in postfix notation. In all notations, the relative order of the operands is the same.

A simple algorithm for converting from infix to prefix (postfix) is as follows: (i) Fully parenthesize the infix expression. It should now consist solely of "terms": a binary operator sandwiched between two operands. (ii) Write down the operands in the same order that they appear in the infix expression. (iii) Look at each term in the infix expression in the order that one would evaluate them, i.e., inner most parenthesis to outer most and left to right among terms of the same depth. (iv) For each term, write down the operand before (after) the operators. The following sequence of steps illustrates converting *X*=(*A***B*-*C*/*D*)↑*E* from infix to postfix:

(*X*=(((*A***B*)-(*C*/*D*))↑*E*))
XABCDE
*XAB***CDE*
*XAB***CD*/*E*
*XAB***CD*/-*E*
*XAB***CD*/-*E*↑
*XAB***CD*/-*E*↑=

This equation has a prefix value of =*X*↑-**AB*/*CDE*. A quick check for determining whether a conversion is correct is to convert the result back into the original format.

The area of computer science that uses prefix and postfix notation (also known as "Polish" and "Reverse Polish" notation for the Polish logician Jan Lukasiewicz) most frequently is compiler design. Typically, expressions (as well as statements) are translated into an intermediate representation, known as a "syntax tree," which is equivalent to either prefix or postfix notation. The intermediate representation can then be easily executed.

References

Tenenbaum, Aaron M. and Moshe J. Augenstein. *Data Structures Using Pascal*, Prentice-Hall (1981), pp.76-95.

Tremblay, Jean-Paul and Richard Bunt. *Introduction to Computer Science*, McGraw-Hill (1979), pp. 456-478.

Sample Problems

Translate the following infix expression into postfix. $$\frac{(A-\frac{B}{C}+D)^{\frac{1}{2}}}{A+B}$$	The expression converts as follows: $(A\text{-}BC/)+D)^{1/2}AB+/$ $(ABC/\text{-}D+)^{1/2}AB+/$ $ABC/\text{-}D+12/\uparrow AB+/$ It is wrong to simplify the 1/2 to .5 or to "sqrt".
Given A=4, B=14 and C=2, evaluate the following prefix expression: * / - + *A B C * A C B*	Convert to infix: $\rightarrow$* / - + *A B C * A C B* $\rightarrow$* / -(*A*+*B*) *C*(*A***C*) *B* $\rightarrow$* /((*A*+*B*) – *C*)(*A***C*) *B* $\rightarrow$*(((*A*+*B*) – *C*) / (*A***C*)) *B* $\rightarrow$((((*A*+*B*) – *C*) / *A***C*)) * *B* $\rightarrow (\frac{A+B-C}{A*C}) * B$ Substitute and simplify: $(\frac{4+14-2}{4*2}) * 14 = 28$

Evaluate the following prefix expression, when A=10, B=2, C=12 and D=2. + / ↑ - *A B* 2 ↑ / - *C D* / *A B* 3 / + *AC B*	Take the original prefix expression, + / ↑ - *A B* 2 ↑ / - *C D* / *A B* 3 / + *AC B* and make passes, each time putting as many binary operators as possible into place: + / ↑(*A*-*B*)2 ↑ /(*C*-*D*)(*A*/*B*) 3 /(*A*+*C*) *B* + /(*A*-*B*)^2 ↑((*C*-*D*)/(*A*/*B*)) 3((*A*+*C*)/*B*) +((*A*-*B*)^2/((*C*-*D*)/(*A*/*B*))^3)((*A*+*C*)/*B*) $\frac{(A-B)^2}{\left(\frac{C-D}{A/B}\right)^3}+\frac{A+C}{B}$ This evaluates to 19.