Stack Application: Infix, Postfix, and Prefix

Basic Definitions and Examples

In **infix** notation the operator is between two operands, in **prefix** notation the operator precedes the two operands, and in **postfix** notation the operator follows the two operands, For example,

A+B infix
+AB prefix
AB+ postfix

If more than one operation appears in the same expression, we use precedence rule. The operations with highest precedence are converted first. For example,

A+B*C infix
+A*BC prefix
ABC*+ postfix

Here, we consider five binary operations: addition (+), subtraction (-), multiplication (*), division (/), and exponentiation ($ or ^). The order of precedence (highest to lower) is: exponentiation, multiplication/division, and addition/subtraction.

When unparenthesized operators of the same precedence are scanned, the order is assumed to be left to right except in the case of exponentiation, where the order is assumed to be from right to left.

Some Examples:

Infix	Prefix	Postfix
A+B	+AB	AB+
A+B–C	–+ABC	AB+C–
(A+B)*(C–D)	*+AB–CD	AB+CD–*
A$B*C–D+E/F/(G+H)	+–*$ABCD//EF+GH	AB$C*D–EF/GH+/+
((A+B)*C–(D–E))$(F+G)	$–*+ABC–DE+FG	AB+C*DE– –FG+$
A–B/(C*D$E)	–A/B*C$DE	ABCDE$*/-
A+B$C–D$E	–+ABCDE	ABC$+DE$–

Converting an Infix expression to Postfix (Stack Application)

Algorithm: To convert infix expression to postfix:

1. Make *postfix-string* empty.
2. Make *operator-stack* empty.
3. For each symbol *symb* in the infix expression (scanning from left to right), repeat:
   - 3.1. If *symb* is an operand:
     - 3.1.1. Add *symb* to the *postfix-string*.
   - 3.2. Otherwise (if *symb* is not an operand):
     - 3.2.1. While *operator-stack* is not empty and the topmost element of *operator-stack* has precedence over *symb*, repeat:

3.2.1.1. Remove the topmost element from the *operator-stack* and add it to the *postfix-string*.

3.2.2. If *operator-stack* is empty or *symb* is not equal to ')':

3.2.2.1. Add *symb* to the top of the *operator-stack.*

3.2.3. Otherwise:

3.2.3.1. Remove the topmost element from the *operator-stack.*

4. While *operator-stack* is not empty, repeat:

4.1. Remove the topmost element from the *operator-stack* and add it to the *postfix-string*.

5. Terminate.

Precedence Rules: Precedence rules among operators other than parenthesis are obvious. In all cases below, first operator is the topmost operator in operator-stack and the second operator is the operator symbol from infix expression.

- Precedence of '(' is less than 'op' for any operator 'op'.
- Precedence of 'op' is less than '(' for any operator 'op' other than ')'.
- Precedence of 'op' is greater than ')' for any operator 'op' other than '('.
- Precedence of ')' is undefined with 'op' for any operator 'op'

Examples:

A+B*C (infix expression)

Symb	***postfix-string***	***operator-stack***
A	A	
+	A	+
B	AB	+
*	AB	+*
C	ABC	+*
	ABC*	+
	ABC*+	

(A+B)*C (infix expression)

Symb	***postfix-string***	***operator-stack***
(		(
A	A	(
+	A	(+
B	AB	(+
)	AB+	
*	AB+	*
C	AB+C	*
	AB+C*	

Evaluating a Postfix Expression (Stack Application)

Algorithm: To evaluate a postfix expression:

1. Make *operand-stack* empty.
2. For each symbol *symb* in the postfix expression (scanning from left to right), repeat:
   - 2.1. If *symb* is an operand:
     - 2.1.1. Add *symb* to at the top of the *operand-stack*.
   - 2.2. Otherwise (if *symb* is an operator):
     - 2.2.1. Remove the topmost element from the *operand-stack* and place it in *opnd2*.
     - 2.2.2. Remove the topmost element from the *operand-stack* and place it in *opnd1*.
     - 2.2.3. Store the result of applying *symb* (operator) to *opnd1* and *opnd2* in *value*.
     - 2.2.4. Insert value to the top of the *operand-stack*.
3. Remove and return topmost element from the *operand-stack* .

Examples:

623+-382/+*2$3+

Symb	***opnd1***	***opnd2***	***value***	***operand-stack***
6				6
2				6, 2
3				6, 2, 3
+	2	3	5	6, 5
-	6	5	1	1
3	6	5	1	1, 3
8	6	5	1	1, 3, 8
2	6	5	1	1, 3, 8, 2
/	8	2	4	1, 3, 4
+	3	4	7	1, 7
*	1	7	7	7
2	1	7	7	7, 2
$	7	2	49	49
3	7	2	49	49, 3
+	49	3	52	52