



Infix to postfix

By: Amjad Saeed



Infix to postfix conversion steps

Scan through an expression, getting one token at a time.

1 Fix a priority level for each operator. For example, from high to low:

3. - (unary negation)

2. * /


1. + - (subtraction)



Thus, high priority corresponds to high number in the table.

2 If the token is an operand, do not stack it. Pass it to the output.

3 If token is an operator or parenthesis, do the following:




-- Pop the stack until you find a symbol of lower priority number than the current one. An incoming left parenthesis will be considered to have higher priority than any other symbol. A left parenthesis on the stack will not be removed unless an incoming right parenthesis is found.

The popped stack elements will be written to output.

--Stack the current symbol.

-- If a right parenthesis is the current symbol, pop the stack down to (and including) the first left parenthesis. Write all the symbols except the left parenthesis to the output (i.e. write the operators to the output).

-- After the last token is read, pop the remainder of the stack and write any symbol (except left parenthesis) to output.



Example: Convert $A * (B + C) * D$ to postfix notation.

Move	Current Token	Stack	Output
1	A	empty	A
2	*	*	A
3	((*	A
4	B	(*	AB
5	+	+(*	AB
6	C	+(*	ABC
7)	*	ABC+
8	*	*	ABC+*
9	D	*	ABC+*D
10		empty	



Notes:

In this table, the stack grows toward the left. Thus, the top of the stack is the leftmost symbol.

In move 3, the left paren was pushed without popping the `*` because `*` had a lower priority than `"("`.

In move 5, `"+"` was pushed without popping `"("` because you never pop a left parenthesis until you get an incoming right parenthesis. In other words, there is a distinction between incoming priority, which is very high for a `"("`, and instack priority, which is lower than any operator.

In move 7, the incoming right parenthesis caused the `"+"` and `"("` to be popped but only the `"+"` as written out.

In move 8, the current `"*"` had equal priority to the `"*"` on the stack. So the `"*"` on the stack was popped, and the incoming `"*"` was pushed onto the stack

Thank You.

