Converting expressions from infix to postfix

The Shunting Yard Algorithm

The idea that expressions in postfix notation can be evaluated much more quickly than those in infix notation suggests that a fast way to evaluate an infix expression might be to convert it first from infix into postfix notation, then evaluate the postfix version.

Dijkstra's shunting-yard algorithm is an algorithm for converting infix expressions to postfix expressions. The algorithm reads tokens and outputs tokens from left-to-right: the next character to be read/written immediately follows the previous one. The infix expression consists of operands, operators and brackets. The algorithm uses a stack to convert the expression from infix to postfix. Operators and brackets are stored on the stack during the conversion while the operands are output in postfix notation in exactly the same order as they occur in the infix expression.

The most basic form of the shunting-yard algorithm is as follows:

For each token in turn in the input infix expression:

- If the token is an operand, append it to the postfix output.
- If the token is an operator A then:
    - While there is an operator B of higher or equal precedence than A at the top of the stack, pop B off the stack and append it to the output.
    - Push A onto the stack.
- If the token is an opening bracket, then push it onto the stack.
- If the token is a closing bracket:
    - Pop operators off the stack and append them to the output, until the operator at the top of the stack is a opening bracket.
    - Pop the opening bracket off the stack.

When all the tokens have been read:

- While there are still operator tokens in the stack:
    - Pop the operator on the top of the stack, and append it to the output.

Example

Consider the infix expression: 5 + ((1 + 2) *4) - 3

Ignoring the spaces used for clarity this consists of 13 characters (symbols).

Applying a simplified version of Dijkstra's shunting yard algorithm yields:

Step num	Input symbol	Stack contents	Postfix Output
0	5		5
1	+	+	5
2	(	+(	5
3	(	+((	5
4	1	+((	51
5	+	+((+	51
6	2	+((+	512
7	)	+(	512+
8	*	+(*	512+
9	4	+(*	512+4
10	)	+	512+4*+
11	-	-	512+4*+
12	3	-	512+4*+3
13	(none)		512+4*+3-

The final output is the postfix expression 512+4*+3-