Infix, Postfix and Prefix

Infix, Postfix and Prefix notations are three different but equivalent ways of writing expressions. It is easiest to demonstrate the differences by looking at examples of operators that take two operands.

Infix: X + Y

Operators are written in-between their operands. This is the usual way we write expressions. An expression such as A * (B + C) / D is usually taken to mean something like: "First add B and C together, then multiply the result by A, then divide by D to give the final answer."

Infix notation needs extra information to make the order of evaluation of the operators clear: rules built into the language about operator precedence and associativity, and brackets ( ) to allow users to override these rules.

Postfix: X Y +

Also known as "reverse polish notation”. Operators are written after their operands. The infix expression given above is equivalent to A B C + * D /

The order of evaluation of operators is always left-to-right, and brackets cannot be used to change this order. Because the "+" is to the left of the "*" in the example above, the addition must be performed before the multiplication.

Operators act on values immediately to the left of them. For example, the "+" above uses the "B" and "C". And, the "*" uses the two values immediately preceding: "A", and the result of the addition. Similarly, the "/" uses the result of the multiplication and the "D".

Prefix: + X Y

Also known as "polish notation". Operators are written before their operands. The expressions given above are equivalent to / * A + B C D

Similar to Postfix, operators are evaluated left-to-right and brackets are unnecessary. Operators act on the two nearest values on the right. If these two nearest values themselves involve computations then this changes the order that the operators have to be evaluated in. In the example above, although the division is the first operator on the left, it acts on the result of the multiplication, and so the multiplication has to happen before the division (and similarly the addition has to happen before the multiplication).

Note: In all three versions, the operands occur in the same order, and just the operators have to be moved to keep the meaning correct.

More Examples:

Infix Expression Prefix Expression Postfix Expression A + B + A B A B + A + B * C + A * B C A B

C * + (A + B) * C * + A B C A B + C * A + B * C + D + + A * B C D A B C * + D + (A + B) * (C + D) *

+ A B + C D A B + C D + * A * B + C * D + * A B * C D A B * C D * + A + B + C + D + + + A B C D

A B + C + D +

Algorithm to convert Infix To Postfix

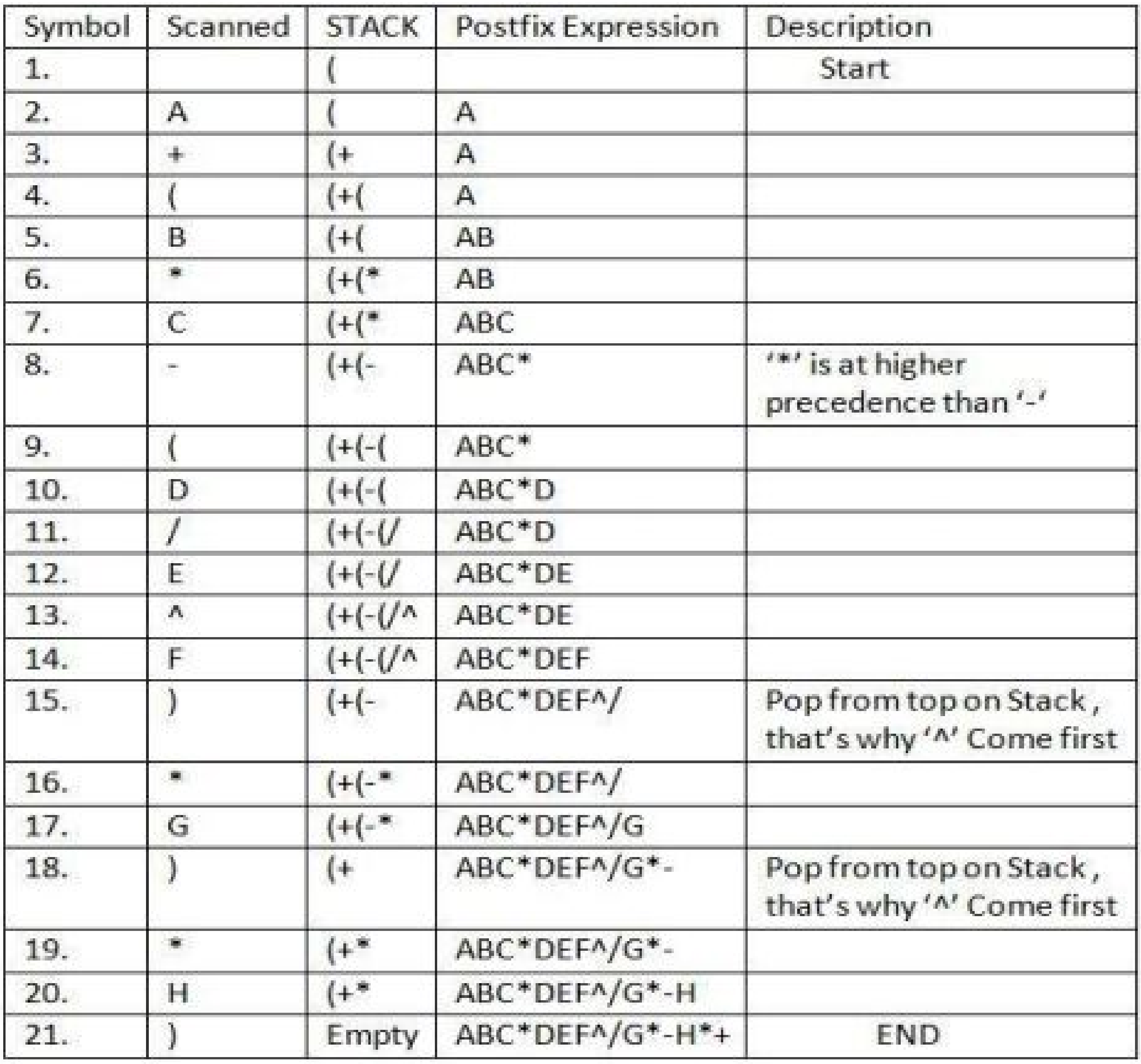
Let, X is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression Y.

1. Push “(“onto Stack, and add “)” to the end of X.
2. Scan X from left to right and repeat Step 3 to 6 for each element of X until the Stack is empty.
3. If an operand is encountered, add it to Y.
4. If a left parenthesis is encountered, push it onto Stack.
5. If an operator is encountered, then:
    1. Repeatedly pop from Stack and add to Y each operator which has the same precedence or higher precedence than operator.
    2. Add operator to Stack.

    [End of If]
6. If a right parenthesis is encountered, then:
    1. Repeatedly pop from Stack and add to Y each operator until a left parenthesis is encountered.
    2. Remove the left Parenthesis.

    [End of If]
7. END.

Given Infix Expression: **(A+(B*C-(D/E^F)*G)*H)**



Symbol	Scanned	STACK	Postfix Expression	Description
1.		(		Start
2.	A	(	A	
3.	+	(+	A	
4.	(	(+(	A	
5.	B	(+(	AB	
6.	*	(+(*	AB	
7.	C	(+(*	ABC	
8.	-	(+(-	ABC*	‘*’ is at higher precedence than ‘-‘
9.	(	(+(-(	ABC*	
10.	D	(+(-(	ABC*D	
11.	/	(+(-(/	ABC*D	
12.	E	(+(-(/	ABC*DE	
13.	^	(+(-(/^	ABC*DE	
14.	F	(+(-(/^	ABC*DEF	
15.	)	(+(-	ABC*DEF^/	Pop from top on Stack , that’s why ‘^’ Come first
16.	*	(+(-*	ABC*DEF^/	
17.	G	(+(-*	ABC*DEF^/G	
18.	)	(+	ABC*DEF^/G*-	Pop from top on Stack , that’s why ‘^’ Come first
19.	*	(+*	ABC*DEF^/G*-	
20.	H	(+*	ABC*DEF^/G*-H	
21.	)	Empty	ABC*DEF^/G*-H*+	END

Resultant Postfix Expression: **ABC*DEFˆ/G*–H*+**

Algorithm to convert Infix To Prefix

1. Reverse the infix expression e.g. A+B*C will become C*B+A. Note while reversing each ‘(‘ will become ‘)’ and each ‘)’ will become ‘(‘.

2. Obtain the postfix expression of the modified expression i.e CB*A+.

3. Reverse the postfix expression. Hence, in our example, prefix will be +A*BC.