Polish Notation

Infix Notation

- Infix notation:
  - for most common arithmetic operations, the operator symbol is place between its two operands
  - For example,

    A + B,   C – D,   E * F,   G / H
  - This is known as infix notation.

Infix Notation

- The order of the operators and operands in an arithmetic does not uniquely determine the order in which operations are to be performed.
  - For example:
  - **A + B * C**

Infix Notation

- In infix notation we use parentheses to get the order of operations to be performed.
  - For example:
    - **(A + B) * C**
    - **A + (B * C)**

Polish Notation

- Polish notation: operators symbol is placed before its two operands.
  - For example:
    - **+AB, -CD, *EF, /GH**
  - This notation is also known as prefix notation

Polish Notation

- Infix to Polish notation example

INFIX		INTERMEDIATE		POLISH
(A+B)*C	→	[+AB] *C	→	*+ABC
A + (B * C)	→	A +[*BC]	→	+A*BC
(A+B)/(C-D)	→	[+AB]/[-CD]	→	/+AB-CD

Polish Notation

- The fundamental property of polish notation is that the order in which the operations are to be performed is completely determined by the positions of the operators and operands in the expression.

Reverse Polish Notation

- Reverse polish notation is also know as postfix.
- In postfix notation operator symbols are placed after its two operands
  - **A + B $\rightarrow$ AB+**

Postfix Notation

- Computer evaluates an arithmetic notation in two steps.
  1. It converts the expression to postfix.
  2. It evaluates the postfix expression.

Operators Precedence

- The three levels of precedence for the usual five binary operators :
- **^, *, /, +, -**
- **Highest:** **(^)Exponentiation**
- **Next highest:** **(*) Multiplication and (/) division**
- **Lowest:** **(+) Addition and (-) subtraction**

Operators Associativity

- **Right to left:** **(^)Exponentiation**
- **Left to right:** **(*) Multiplication and (/) division**
- **Left to right :** **(+) Addition and (-) subtraction**

Infix to Postfix Algorithm

POLISH(Q, P)

Suppose Q is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression P.

Infix to Postfix Algorithm

1. Push “(“ onto STACK, and add “)” to the end of Q.
2. Read Q from left to right and repeat step 3 to 6 for each element of Q until the STACK is empty.
3. If an operand is encountered add it to P.
4. If a left parenthesis is encountered, push it onto STACK.

Infix to Postfix Algorithm

5. If an operator “×” is encountered, then:
    a. Repeatedly pop from STACK and add to P each operator which has the same precedence or higher precedence the operator “×”.
    b. Add operator “×” to STACK.
6. If a right parenthesis is encountered, then:
    a. Repeatedly pop from STACK and add to P each operator until a left parenthesis is encountered.
    b. Remove the left parenthesis form STACK.
7. Exit

Source: Theory and Problems of Data Structure – Seymour Lipschutz

Example – infix to postfix

Q: A + (B * C – (D / E ^ F) * G) * H

Symbol scanned	STACK	Expression P
-	(	-
(1) A	(	A
(2) +	(+	A
(3) (	(+ (	A
(4) B	(+ (	A B
(5) *	(+ (*	A B
(6) C	(+ (*	A B C

Status of STACK

Symbol scanned	STACK	Expression P
(6) C	(+ (*	A B C
(7) -	(+ (-	A B C*
(8) (	(+ (- (	A B C*
(9) D	(+ (- (	A B C * D
(10) /	(+ (- (/	A B C * D
(11) E	(+ (- (/	A B C * D E
(12) ^	(+ (- (/ ^	A B C * D E
(13) F	(+ (- (/ ^	A B C * D E F

Status of STACK

Symbol scanned	STACK	Expression P
(13) F	(+ (- (/ ^	A B C * D E F
(14))	(+ (-	A B C * D E F ^ /
(15) *	(+ (- *	A B C * D E F ^ /
(16) G	(+ (- *	A B C * D E F ^ / G
(17))	(+	A B C * D E F ^ / G * -
(!8) *	(+ *	A B C * D E F ^ / G * -
(19) H	(+ *	A B C * D E F ^ / G * - H
(20))		A B C * D E F ^ / G * - H * +

Postfix expression evaluation

POSTFIXEVAL(P)

This algorithm finds the VALUE of an arithmetic expression P written in postfix notation

Postfix expression evaluation

1. Add a right parenthesis “) ” at the end of P.
2. Scan P from left to right and repeat steps 3 and 4 for each element of P until right parenthesis “)” is encountered.
3. If an operand is encountered, put it on STACK

Postfix expression evaluation

4. If an operator “×” is encountered, then
   a) Remove two top elements of STACK, where A is top element and B is the next-to-top element.
   b) Evaluate B “×” A (B operator A).
   c) Place the result of “step b” back on STACK.
5. Set VALUE equal to the top element of STACK.
6. Exit.

Source: Theory and Problems of Data Structure – Seymour Lipschutz

Example postfix evaluation

Consider the following expression P written in postfix notation:

P: 5, 6, 2, +, *, 12, 4, /, -

Postfix expression: 5, 6, 2, +, *, 12, 4, /, -

Symbol scanned	STACK
(1) 5	5
(2) 6	5, 6
(3) 2	5,6,2
(4) +	5, 8
(5)*	40
(6) 12	40, 12
(7) 4	40, 12, 4
(8) /	40, 3
(9) -	37
(10) “)”	

Infix to Postfix

- Translate, by inspection and hand, each infix expression into its equivalent postfix expression
  a) (A-B)*(D/E)
  b) (A+B^D)/(E-F)+G
  c) A*(B+D)/E-F*(G+H/K)

Answers

a) (A-B)*(D/E) → AB-DE/*

b) (A+B^D)/(E-F)+G → ABD^+EF-/G+

c) A*(B+D)/E-F*(G+H/K) → ABD+*E/FGHK/+*-

Thank you