Lecture :- 23

Arithmetic Expression

Assume the following three levels of precedence for the usual five binary operations:→

Highest : Exponentiation (↑)

Next Highest : Multiplication (*) and division(/)

Lowest : Addition (+) and subtraction (−)

- ~~Le~~ Let Q be an arithmetic expression
- It involves constants , operand , operations.
- Assume that Q contains no unary operation
- Operations on the same level are performed from left to right (ie in any paranthesis free expr-ession)

↳ This part is not standard ∵ in some languages exponentiations are perform from Right to left.

Infix Notation :—

When the operator symbol is placed between its two operands

eg ① A + B here A & B are operands
+ → operator

eg ② C − D

$\boxed{eg3}$ $(A+B) * C$

<u>Polish Notation</u> or <u>Prefix Notation</u> :-

In which operator symbol is placed before its two operands.

<u>eg</u> $+AB$ here $\begin{bmatrix} A \& B \text{ are operands} \\ + \rightarrow \text{operator} \end{bmatrix}$

<u>eg</u> $*EF$

say we have infix expression such as

$(A+B) * C$

<u>Now</u> its prefix notation is

$* + ABC$

<u>Another Example</u>

$(A+B) / (C-D) \Rightarrow [+AB] / [-CD]$

$\Rightarrow /+AB-CD$

Here we had ~~[illegible]~~ used brackets [ ] to indicate the partial translation.

But one never needs paranthesis when writing expression in Polish Notation.

Reverse Polish Notation :- or Postfix Expression
Postfix Notation.

The notation in which the operator symbol is placed after its two operands.

eg AB+

Note :-

The computer usually evaluate an arithmetic expression written in infix notation in two steps:-

① First, it convert the expression to postfix notation

② Second, it evaluates the postfix expression.

In each step, stack is the main tool.

Transforming Infix Expression into Postfix Expression

- The algorithm will transform infix expression Q into its equivalent postfix expression P
- Algorithm uses a stack to temporarily hold operators and left paranthesis.
- The postfix expression P will be constructed from L to R using the operands from Q and operators which are removed from stack.

```
Polish ( Q, P)
{
1. Push "(" onto stack and ")" to the end of Q

2. Scan 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

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 than ⊗
   (b) Add ⊗ to stack.

6. If a right paranthesis is encountered, then:
   (a) Repeatedly pop from Stack and add to P
       each operator until a left paranthesis is
       encountered.
   (b) Remove the left paranthesis // Do not add
                                      it to P
}

7. Exit

}
```

Example :-

Let $Q = A+(B*C-(D/E\uparrow F)*G)*H)$ ← Sentinel added.

- Add) to Q at the end
- Push (to Stack

Symbol Scanned	Stack	Expression P
① A	(	A
② +	(+	A
③ (	(+ (	A
④ B	(+ (	A B
⑤ *	(+ (*	A B
⑥ C	(+ (* No change	A B C
⑦ −	(+ (− Multiply is pop	A B C *
⑧ (	(+ (− (	A B C *
⑨ D	Stack remain same	A B C * D
⑩ /	(+ (− (/	A B C * D
⑪ E	No change on stack	A B C * D E
⑫ ↑	(+ (− (/ ↑	A B C * D E // No pop from stack as / has low priority than ↑
⑬ F	No change on stack	A B C * D E F
⑭)	(+ (− Pop till we find (	A B C * D E F ↑ /
⑮ *	(+ (− * No pop as − has low priority	A B C D * D E F ↑ /
⑯ G	No change on stack	A B C D * D E F ↑ / G
⑰)	(+ Pop till we find (	A B C D * D E F ↑ / G * −
⑱ *	(+ * No pop as + has low priority	A B C D * D E F ↑ / G * −
⑲ H	No change on stack	A B C D * D E F ↑ / G * − H
⑳)	Pop till we find (& stack is empty so stop	A B C D * D E F ↑ / G * − H * +

Evaluation of a Postfix Expression

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

① Add ")" at the end of P. // to act as sentinel

② Scan P from left to Right and repeat step 3 & 4 for each element of P until the sentinel ") " is encountered.

③ If an operand is encountered, put it on Stack.

④ If an operator ⊗ is encountered then:

 ⓐ Remove the two top elements of stack where A is the top element and B is the next-to-top element

 ⓑ Evaluate B ⊗ A

 ⓒ Place the result of ⓑ back on stack

⑤ Set Value equal to the top element on stack

⑥ Exit.

Example

Consider the following postfix expression P

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

The equivalent infix ~~@~~ expression for above Postfix is

Q : 5 * (6+2) - 12/4

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

add sentinel

	Symbol Scanned	Stack
1	5	5
2	6	5 | 6
3	2	5 | $\cancel{6}$ | $\cancel{2}$
4	+	$\cancel{5}$ | $\cancel{8}$
5	*	40
6	12	40 | 12
7	4	40 | $\cancel{12}$ | $\cancel{4}$
8	/	$\cancel{40}$ | $\cancel{3}$
9	-	37
10	)	

A = 2 , B = 6
B + A ⇒ 8.

A = 8 , B = 5
B * A ⇒ 40

A = 4 , B = 12
B / A ⇒ 3

A = 3 , B = 40
B - A ⇒ 37

37 assign to value.

Self Assessment Question

Q1 Consider the following arithmetic Expression (Infix)

$Q = A * (B+D)E - F * (G + H/K)$

Convert the above infix expression to equivalent postfix expression.

Q2 Consider the following postfix notation P:

P : 12, 7, 3, -, /, 2, 1, 5, +, *, +

~~Ea~~ Evaluate the above postfix expression.