Data Structure & Algorithms

Nilesh Ghule

Stack / Queue in Java collections

- class java.util.Stack<E>
  - E push(E);
  - E pop();
  - E peek();
  - boolean isEmpty();

- interface java.util.Queue<E>
  - boolean offer(E e);
  - E poll();
  - E peek();
  - boolean isEmpty();

Math Expressions

operands

operator

$A + B$

$A \underset{\sim}{+} B \rightarrow$ $\underset{\sim}{\text{in}}\text{fix}$ notation $\rightarrow$ human readable

$$\left.\begin{array}{l} A \quad B \quad \underset{\sim}{+} \rightarrow \text{postfix notation} \\ \\ \underset{\sim}{+} \quad A \quad B \rightarrow \text{prefix notation} \end{array}\right\} \text{to develop algo which can be solved by Computer}$$

① input infix expr from user.

② Convert expr into postfix or prefix expr.

③ solve that postfix/prefix expr.

5 + 9 – 4 * (8 – 6 / 2) + 1 $ (7 – 3)

infix to postfix

A + B

A B +

high ( )

priority $ .... power

* / %

low + –

```
  6     7     5       2     1   8   4     3
  5  +  9  -  4  *  ( 8  -  6 / 2 ) + 1 $ ( 7 - 3 )
  5  +  9  -  4  *  ( 8  -  6 2 / ) + 1 $ ( 7 - 3 )
  5  +  9  -  4  *  8 6 2 / -       + 1 $ ( 7 - 3 )
  5  +  9  -  4  *  8 6 2 / -       + 1 $ 7 3 -
  5  +  9  -  4  *  8 6 2 / -       + 1 7 3 - $
  5  +  9  -  4 8 6 2 / - *         + 1 7 3 - $
  5  9  +  -  4 8 6 2 / - *         + 1 7 3 - $
  5  9  +  4 8 6 2 / - * -          + 1 7 3 - $
  5  9  +  4 8 6 2 / - * -          1 7 3 - $ +
```

5 + 9 – 4 * (8 – 6 / 2) + 1 $ (7 – 3)

infix to prefix

A + B

+ A B

$$5 \overset{⑥}{+} 9 \overset{⑦}{-} 4 \overset{⑤}{*} (8 \overset{②}{-} 6 \overset{①}{/} 2) \overset{⑧}{+} 1 \overset{④}{\$} (7 \overset{③}{-} 3)$$

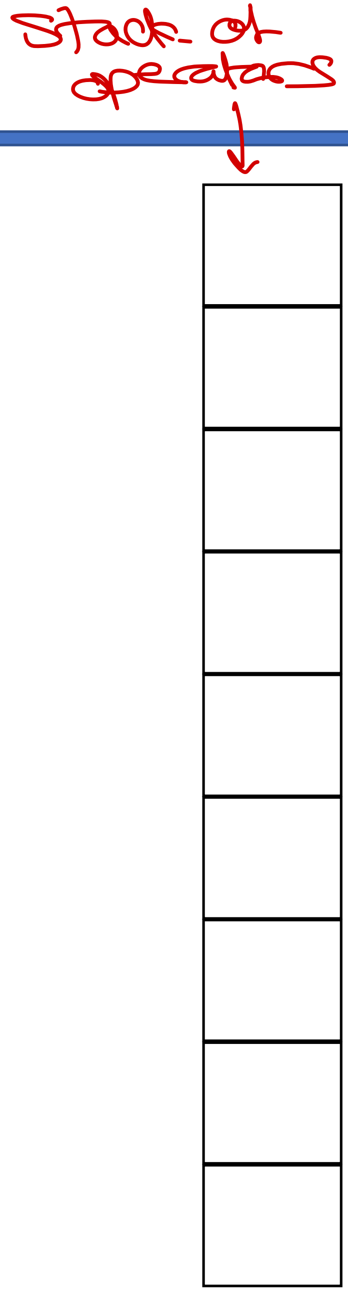
$$\underline{+ \; - \; + \; 5 \; 9 \; * \; 4 \; - \; 8 \; / \; 6 \; 2 \; \$ \; 1 \; - \; 7 \; 3}$$

Infix to Postfix

- 5 + 9 – 4 * (8 – 6 / 2) + 1 $ (7 – 3)

= 5 9 + 4 8 6 2 / – * – 1 7 3 – $ +

1. traverse infix expr from left to right.
2. if operand, append to postfix string.
3. if operator, push on stack.
4. if prio of topmost op on stack is greater or equal to prio of current op, then pop it & append to postfix.
5. if infix expr is completed, pop all op from stack one by one and append to postfix.
6. if current operator is '(', then push on stack.
7. If current operator is ')', then pop ops one by one from stack and append to postfix until you get '(' on stack. Also discard '('.



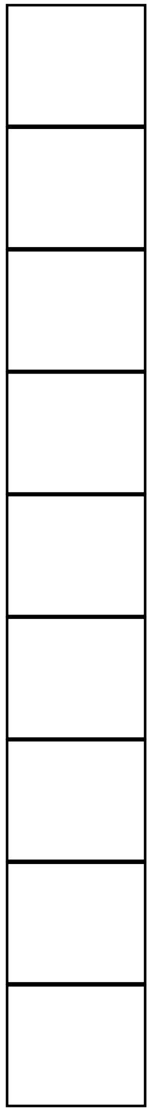
Infix to Prefix

- 5 + 9 – 4 * (8 – 6 / 2) + 1 $ (7 – 3)

3 7 - 1 $ 2 6 / 8 - 4 * 9 5 + - +

$\xleftarrow{\qquad\qquad\qquad\qquad\qquad\qquad}$

+ - + 5 9 * 4 - 8 / 6 2 $ 1 - 7 3



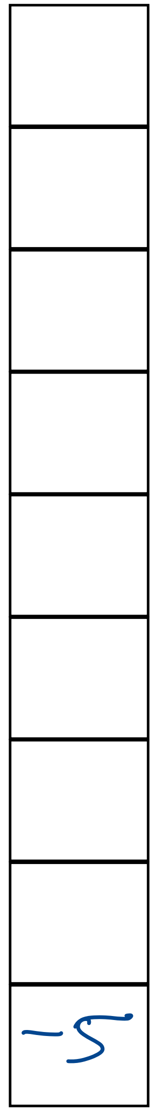
Postfix Evaluation

'5' ≠ 5 '0' ≠ 0

$53 - 48 \rightarrow 5$ 48

- 5 9 + 4 8 6 2 / - * - 1 7 3 - $ +

stack of operands



① traverse postfix from left to right.

② if operand, push on stack.

③ if operator, pop two operands from stack, calculate result & push it on stack.

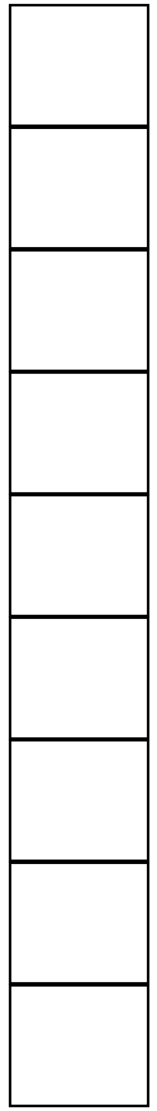
~~result~~ = pop2 $\bigcirc$op pop1

④ repeat until postfix string is completed.

⑤ pop final result from stack.

Prefix Evaluation

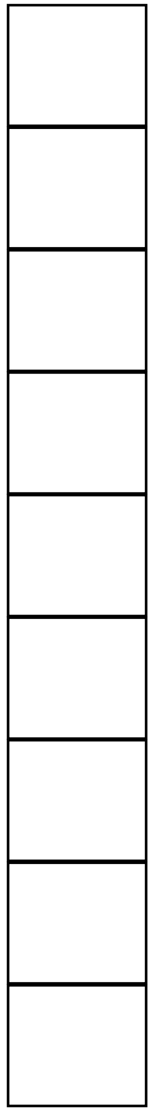
- + - + 5 9 * 4 – 8 / 6 2 $ 1 – 7 3



Postfix to Infix

- While there are input symbol left
- Read the next symbol from input.
- If the symbol is an operand , Push it onto the stack.
- Otherwise, the symbol is an operator.
- If there are fewer than 2 values on the stack
- Show Error
- Else
- Pop the top 2 values from the stack.
- Put the operator, with the values as arguments and form a string.
- Encapsulate the resulted string with parenthesis.
- Push the resulted string back to stack.
- If there is only one value in the stack
- That value in the stack is the desired infix string.
- If there are more values in the stack
- Show Error

- a b c - + d e – f g – h + / *



Prefix to Postfix

- Read the Prefix expression in reverse order (from right to left)
- If the symbol is an operand, then push it onto the Stack
- If the symbol is an operator, then pop two operands from the Stack
- Create a string by concatenating the two operands and the operator after them.
- string = operand1 + operand2 + operator
- And push the resultant string back to Stack
- Repeat the above steps until end of Prefix expression.

- * + A B – C D

Parenthesis Balancing

- 5 + ([9 – 4] * (8 – { 6 / 2 }])) ↑

0 1 2

open = ([{

close =)] }

] × (

① ⓪

1 +) * 3

↓

) ⟶ pop open from stack, but stack is empty.

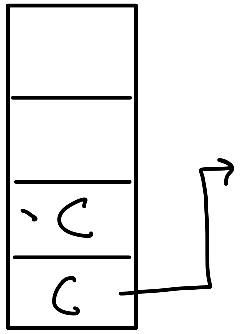
means – extra parenthsis

((1 + 2) * 3 ⊗

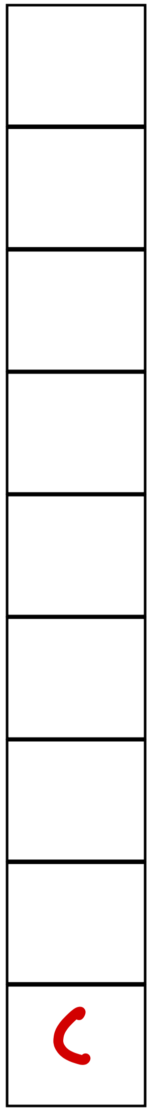
✓ ✓

↓

) = (←-----



extra ele on stack, means closing is missing,



SUNBEAM

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>