DATA STRUCTURES AND ALGORITHMS

SLIDE 9

Infix, Postfix and Prefix

- Infix, Postfix and Prefix notations are three different but equivalent ways of writing expressions.

Infix Notation:

- *The traditional method of our writing of mathematical expressions is called as the infix expressions.*
- *It is of the form <operand><operator><operand>.*
- *As the name suggests, here the operator is fixed inside between the operands. e.g. A+B here the plus operator is placed inside between the two operators, (A*B)/Q.*

- ***Prefix**: An expression is called the prefix expression if the operator appears in the expression before the operands. Simply of the form (operator operand1 operand2).*

*Example : *+AB-CD (Infix : (A+B) * (C-D))*

- ***Postfix**: An expression is called the postfix expression if the operator appears in the expression after the operands. Simply of the form (operand1 operand2 operator).*

Example : AB+CD- (Infix : (A+B * (C-D))*

- During processing, a computer uses prefix or postfix expression to execute data.

REASON

- In the infix expressions, it is difficult to keep track of the operator precedence whereas here the postfix expression itself determines the precedence of operators (which is done by the placement of operators)i.e the operator which occurs first operates on the operand.
- Infix expression are easy to understand and evaluate for human beings. However computer finds it difficult to parse - Information is needed about operator precedence and associativity rules, and brackets which override these rules.
- Hence we have postfix and prefix notations which make the computer take less effort to solve the problem.

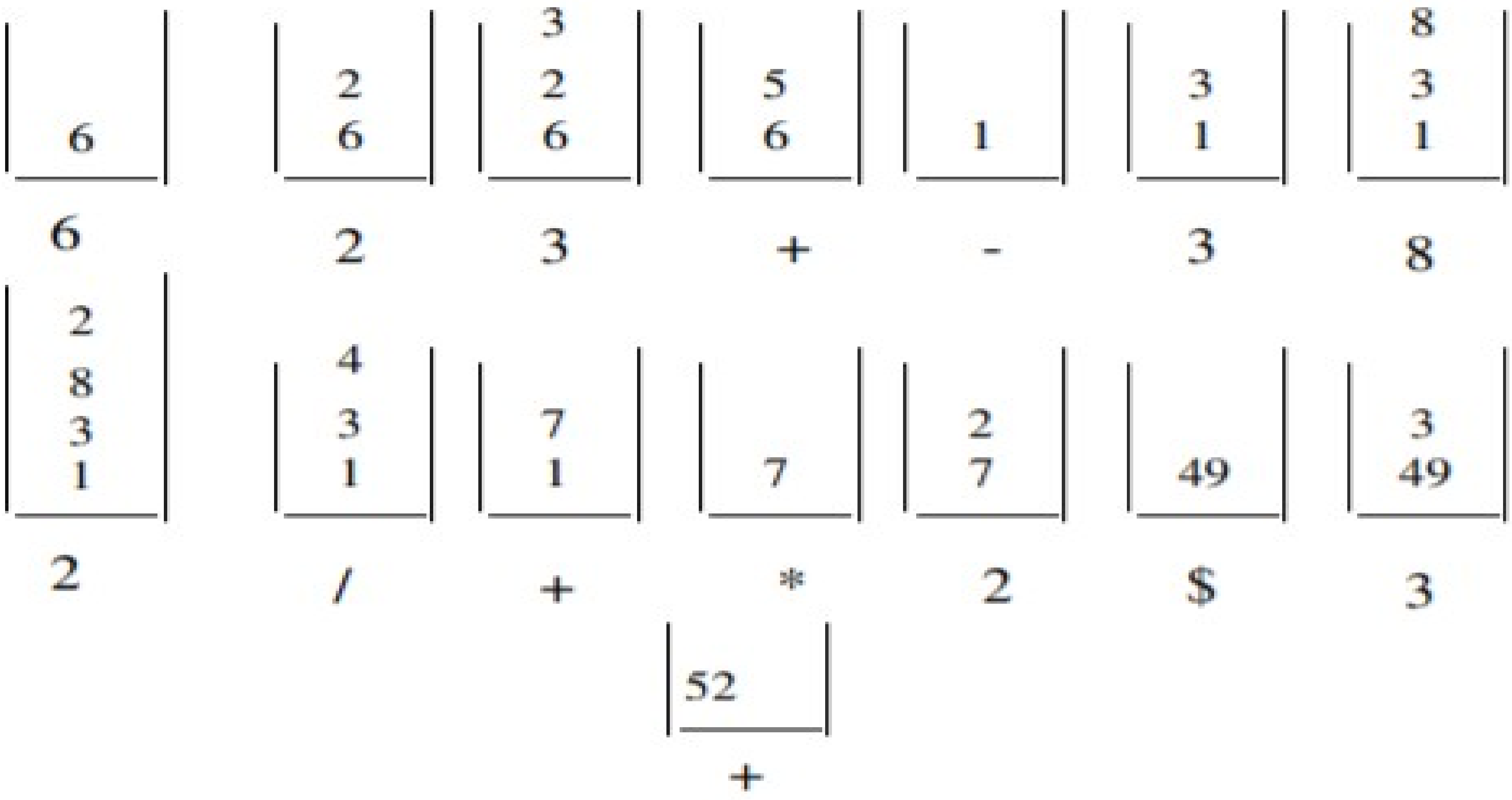
Sr.No.	Infix Notation	Prefix Notation	Postfix Notation
1	a + b	+ a b	a b +
2	(a + b) ∗ c	∗ + a b c	a b + c ∗
3	a ∗ (b + c)	∗ a + b c	a b c + ∗
4	a / b + c / d	+ / a b / c d	a b / c d / +
5	(a + b) ∗ (c + d)	∗ + a b + c d	a b + c d + ∗
6	((a + b) ∗ c) - d	- ∗ + a b c d	a b + c ∗ d -

Postfix Evaluation Algorithm

- *We shall now look at the algorithm on how to evaluate postfix notation –*
- *Step 1 – scan the expression from left to right Step 2 – if it is an operand push it to stack*
- *Step 3 – if it is an operator pull operand from stack and perform operation*
- *Step 4 – store the output of step 3, back to stack Step 5 – scan the expression until all operands are consumed*
- *Step 6 – pop the stack and perform operation*

Use the algorithm to evaluate: 6 2 3 + - 3 8 2 / + * 2 $ 3 +

Evaluating Postfix Expressions (continued)



- *A prefix expression works entirely in same manner as the postfix expression.*
- *While evaluating a prefix expression, the operators are applied to the operands immediately on the right of the operator.*

Evaluation of Prefix Expression

Step 1 – scan the expression from left to right

Step 2 – if it is an operand push it to stack

Step 3 – for any TWO consecutive operands pull operator before them from stack and perform operation

Step 4 – store the output of step 3, back to stack

Step 5 – scan the expression until all operands are consumed

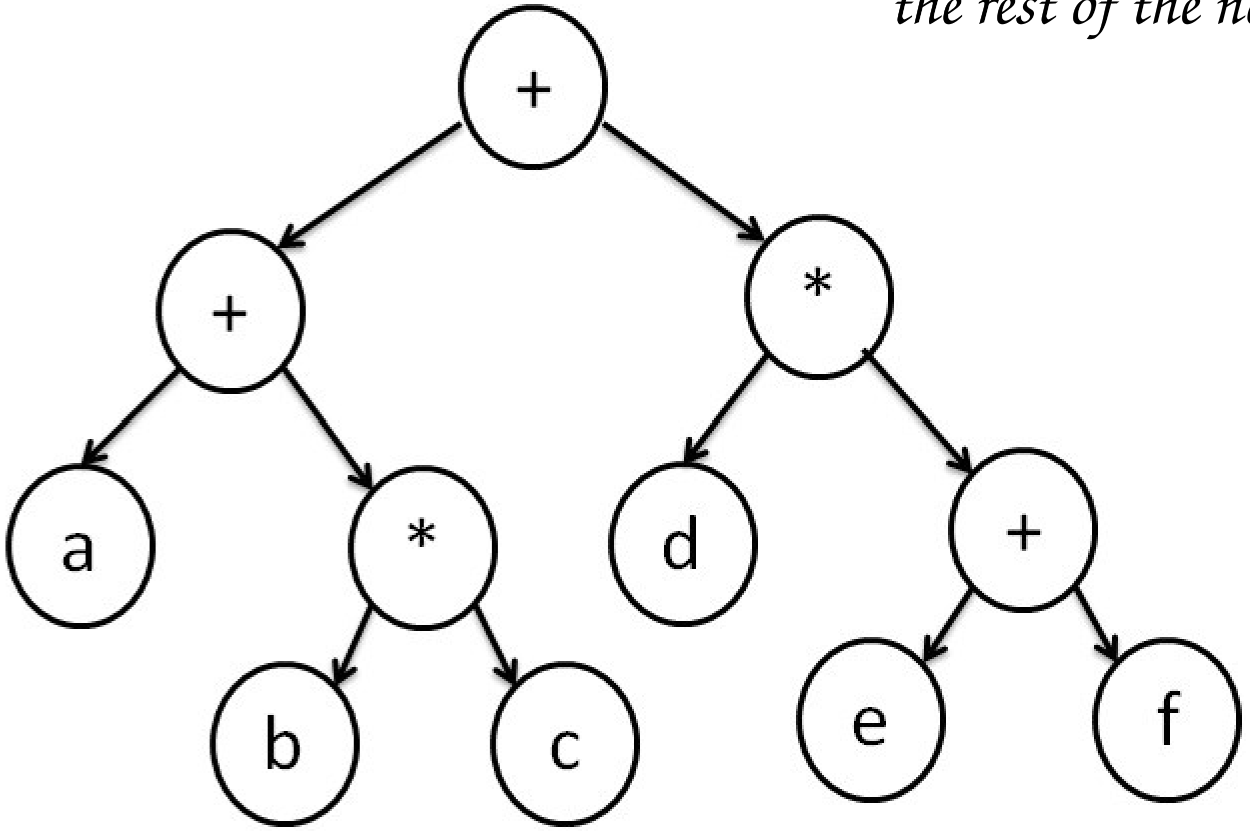
Step 6 – pop the stack and perform operation

BINARY EXPRESSION TREE

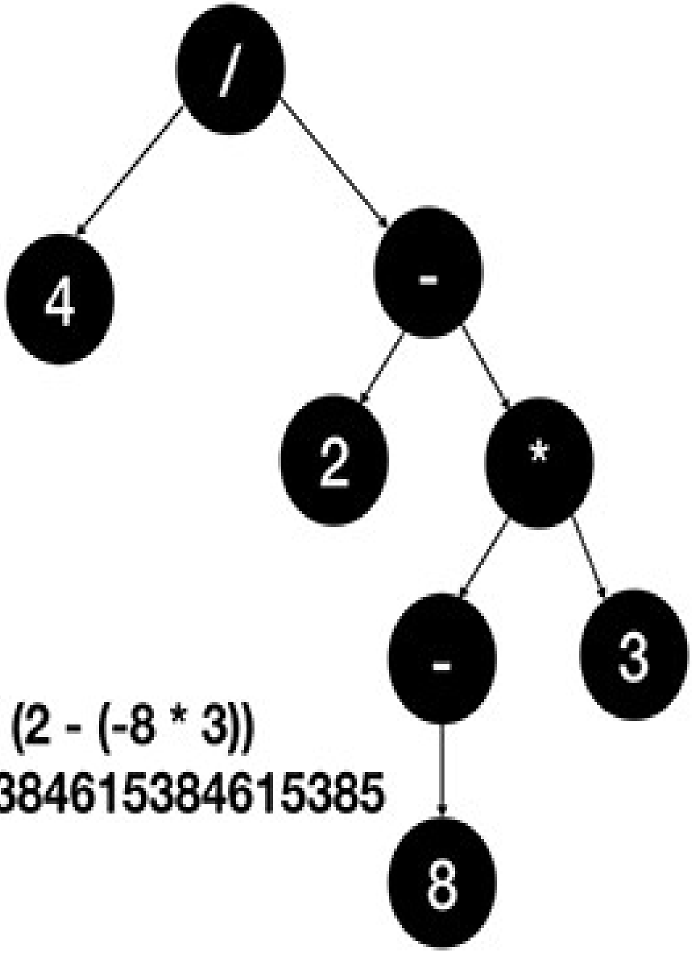
- *A binary expression tree is a specific kind of a binary tree used to represent expressions.*
- *Two common types of expressions that a binary expression tree can represent are algebraic and boolean*

a + (b * c) + d * (e + f)

NB: In a binary expression tree, all the leafs are operands, whilst the rest of the nodes are operators

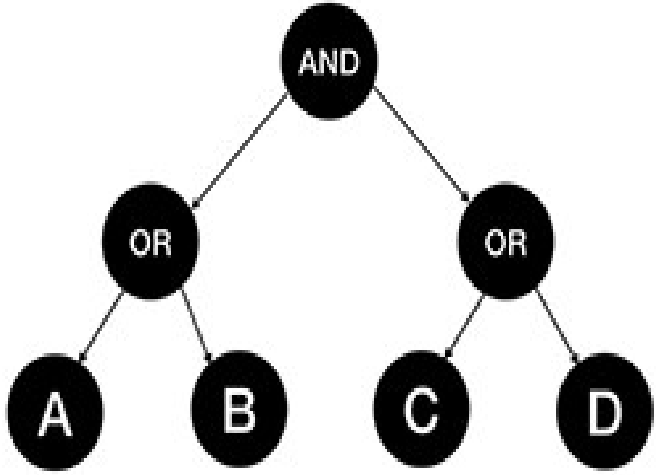


4 / (2 - (-8 * 3))



```
> 4 / (2 - (-8 * 3))
0.15384615384615385
```

(A OR B) AND (C OR D)



(A & C), (B & C),
(A & D), (B & D)

Constructing a Binary Expression Tree

- Apply BODMAS
- Consider the placement of operands and operators in the expression

Traversing a Binary Expression Tree

- *An algebraic expression can be produced from a binary expression tree by various traversal methods.*
- *An infix expression is produced by the inorder traversal, a postfix expression is produced by the post-order traversal, and a prefix expression is produced by the pre-order traversal*

END.

WAIRAGU G.R.