Reverse Polish Notation

MARTZEL BASTE

Reverse Polish Notation (RPN)

- was devised as a method of simplifying mathematical expressions.
- It predates modern computers.
- Early program translators converted expressions to RPN for evaluation.

There are different types of expression formats:

Pre A In B Post

Prefix expression	* + a b - c d
Infix expression	(a+b) * (c-d)
Postfix expression	a b + c d - *

Infix Evaluation 2+3*5

PMDAS

2+3*5

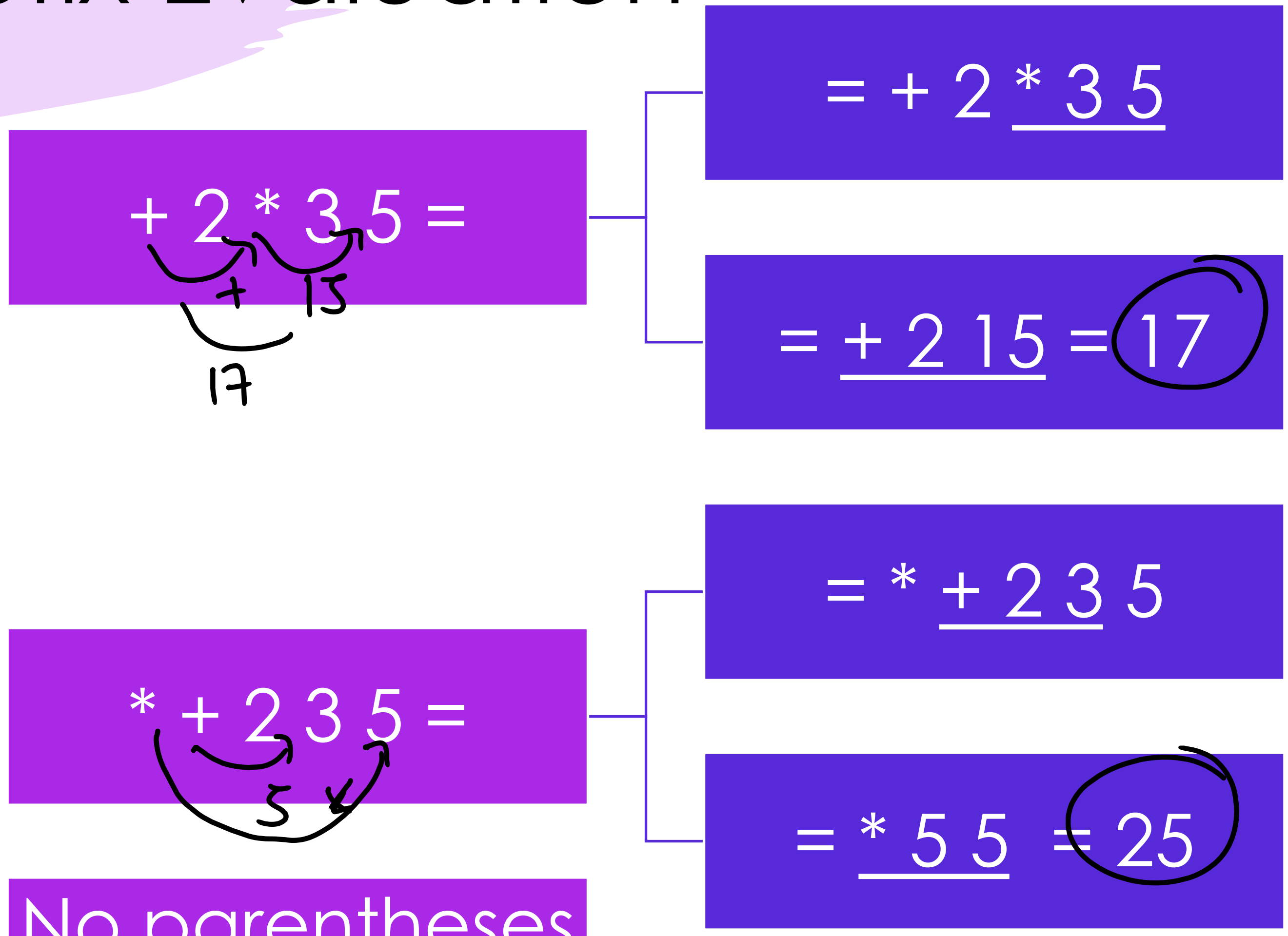
- = 2+15
- = 17

(2+3)*5

- = 5*5
- = 25

Infix notation requires Parentheses.

Prefix Evaluation



+ 2 * 3 5 =

+ 15

17

$= + 2\ \underline{*\ 3\ 5}$

$= \underline{+\ 2\ 15} = 17$

* + 2 3 5 =

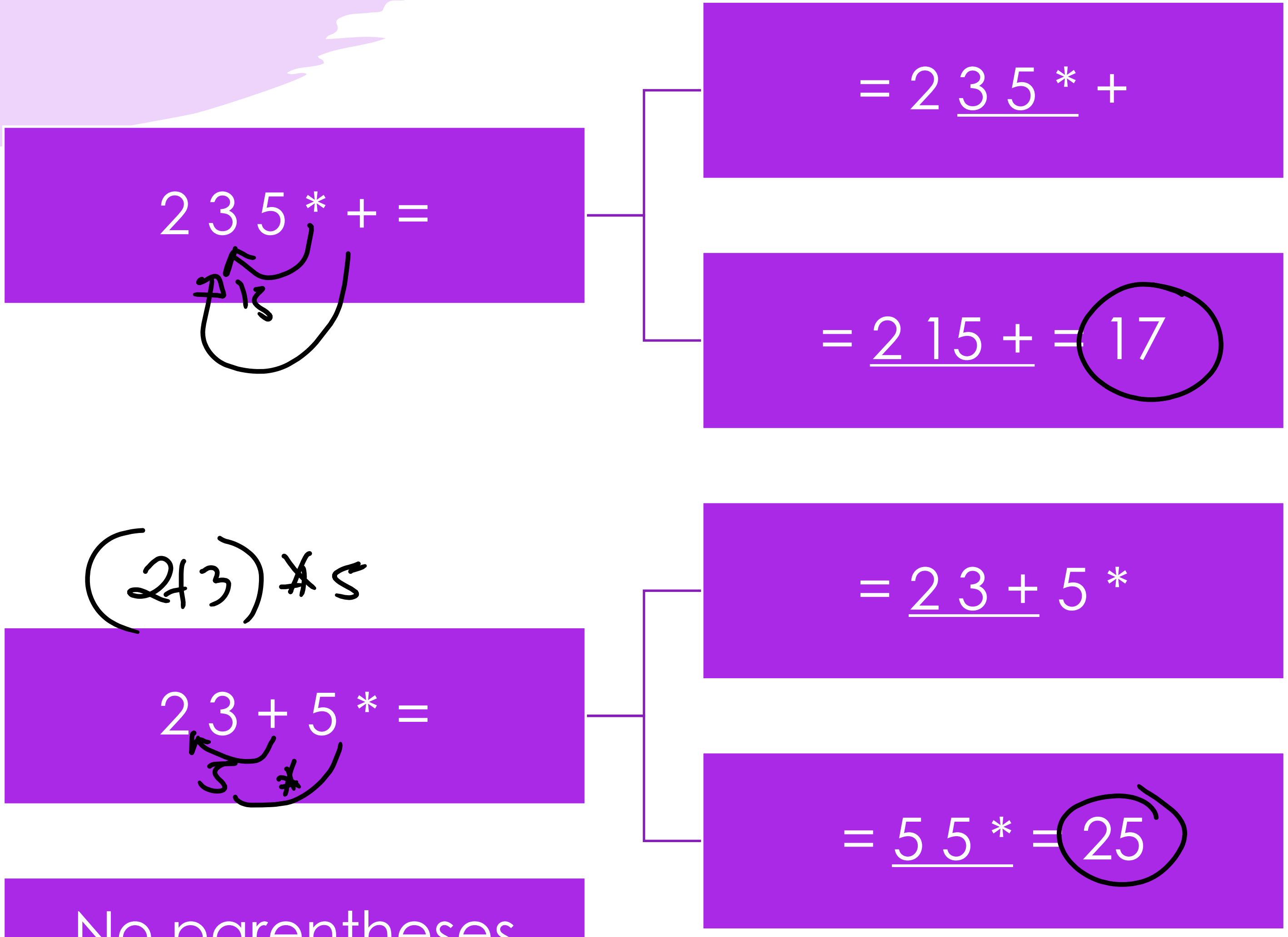
5 x

$= *\ \underline{+\ 2\ 3}\ 5$

$= \underline{*\ 5\ 5} = 25$

No parentheses needed!

Postfix Evaluation



2 3 5 * + =

15

= 2 3 5 * +

= 2 15 + = 17

(2+3) * 5

2 3 + 5 * =

5 *

= 2 3 + 5 *

= 5 5 * = 25

No parentheses needed here either!

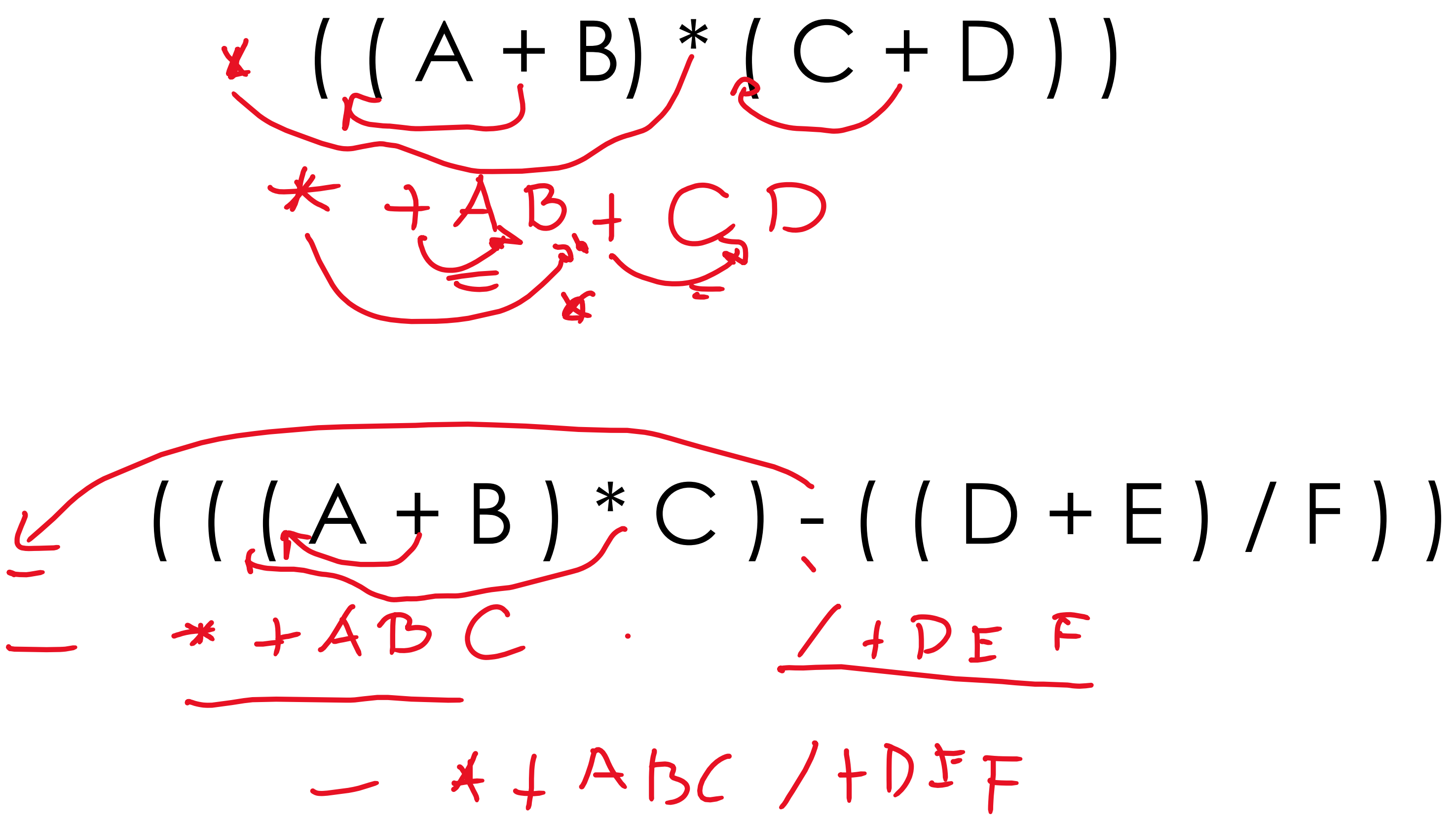
Fully Parenthesized Expression (FPE)

- A FPE has exactly one set of Parentheses enclosing each operator and its operands.
- Which is fully parenthesized?
  - ❑(A + B) * C
  - ❑((A + B) * C) ✓
  - ❑((A + B) * (C))

$\{(2+3) * (3+4)+5\}$

Infix to Prefix Conversion

Move each operator to the left of its operands & remove the parentheses:



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

* +AB+CD

(((A + B) * C) - ((D + E) / F))

- *+ABC /+DEF

- *+ABC /+DEF

Infix to ~~Prefix~~ Postfix Conversion

Move each operator to the right of its operands & remove the parentheses:

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

AB+ CD+ *

(((A + B) * C) - ((D + E) / F)) -

AB+C* DE+F/ -

- ✏Operand order does not change!
- ✏Operators are in order of evaluation!

Prefix to Postfix Conversion (Vice Versa)

Convert to Infix, then to a certain notation.

Infix

Postfix

- + 4 9 8 → 4+9−8 → 4 9 + 8 − = 1

13−8 = 5

13 −8 = 5

* + 2 3 – 4 8 → 2+3*4-8 → 2 3 + 4 8 − *

Infix	Prefix	Postfix
2 + 3 – 4 = 1	- + 2 3 4 = 1 5	2 3 + 4 - = 1 5
2 + (3 - 4) = 1	+ 2 - 3 4	2 3 4 - +
2 + 3 * 4 = 12+2 = 14	+ 2 * 3 4 2 + 12 = 14	2 3 4 * + = 14
(2 + 3) * 4 5 * 4 = 20	* + 2 3 4 = 20 5 *	2 3 + 4 * = 20 5
(2 - 3) * (4 + 5)	* - 2 3 + 4 5	2 3 - 4 5 + *
2 + 8 * 5 / 10 L to R 40/10 = 4 + 2 = 6	+ 2 / * 8 5 10 4 6	2 8 5 * 10 / + 40 /10 = 4 + 2 = 6 ✓

Advantages

- RPN expressions do not need brackets and there is no need for precedence of operators
- RPN is simpler for a machine to evaluate
- There is no need for backtracking in evaluation as the operators appear in the order required for computation and can be evaluated from left to right.

Implementation

- Stack
- Trees – Expression Trees