Prefix, Postfix, Infix Notation

Infix Notation

- To add A, B, we write

  A+B

- To multiply A, B, we write

  A*B

- The operators ('+' and '*') go in between the operands ('A' and 'B')
- This is *"Infix"* notation.

Prefix Notation

- Instead of saying "A plus B", we could say "add A,B " and write

  + A B

- "Multiply A,B" would be written

  * A B

- This is *Prefix* notation.

Postfix Notation

- Another alternative is to put the operators after the operands as in

A B +

and

A B *

- This is *Postfix* notation.

- The terms infix, prefix, and postfix tell us whether the operators go between, before, or after the operands.

Pre A In B Post

Parentheses

- Evaluate 2+3*5.
- + First:

  (2+3)*5 = 5*5 = 25
- * First:

  2+(3*5) = 2+15 = 17
- Infix notation requires Parentheses.

What about Prefix Notation?

- + 2 * 3 5 =

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

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

- * + 2 3 5 =

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

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

- No parentheses needed!

Postfix Notation

- 2 3 5 * + =

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

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

- 2 3 + 5 * =

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

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

- No parentheses needed here either!

Infix to Prefix Conversion

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

$$\begin{tikzcd}[column sep=tiny] (& (& A & + \arrow[lll, bend left] & B) & * & (& C & + & D &) &) \end{tikzcd}$$

Infix to Prefix Conversion

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

$$\begin{tikzcd}[column sep=small] {} & (& + & A & B & * \arrow[lllll, bend left] & (& C & + & D &) &) \end{tikzcd}$$

Infix to Prefix Conversion

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

$$\begin{tikzcd}[column sep=tiny] * & + & A & B & {} & (& C & + & D &) \arrow[from=1-8, to=1-5, bend left] \end{tikzcd}$$

Infix to Prefix Conversion

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

* + A B + C   D

Order of operands does not change!

Infix to Postfix

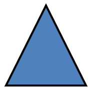
$$\begin{tikzcd}[column sep=tiny]
(& (& (& A & + \arrow[rr, bend right] & B &) & * \arrow[rr, bend right] & C &) & - \arrow[rrrrrrrrrr, bend right] & (& (& D & + \arrow[rr, bend right] & E &) & / \arrow[rr, bend right] & F &) &)
\end{tikzcd}$$

A B + C * D E + F / -

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

Stacks: Infix to Postfix

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



- stack: <empty>
- output: []

Stacks: Infix to Postfix

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

- stack: (
- output: []

Stacks: Infix to Postfix

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

- stack: ((
- output: []

Stacks: Infix to Postfix

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

▲

- stack: (((
- output: []

Stacks: Infix to Postfix

+ B) * (C - E)) / (F + G))

- stack: (((
- output: [A]

Stacks: Infix to Postfix

B) * (C - E)) / (F + G))

- stack: (((+
- output: [A]

Stacks: Infix to Postfix

) * (C - E)) / (F + G))

- stack: (((+
- output: [A B]

Stacks: Infix to Postfix

* (C - E)) / (F + G))

- stack: ((
- output: [A B +]

Stacks: Infix to Postfix

(C - E)) / (F + G))

▲

- stack: ((*
- output: [A B +]

Stacks: Infix to Postfix

C - E)) / (F + G))

- stack: ((* (
- output: [A B +]

Stacks: Infix to Postfix

- E)) / (F + G))

- stack: ((* (
- output: [A B + C]

Stacks: Infix to Postfix

E)) / (F + G))

▲

- stack: ((* (-
- output: [A B + C]

Stacks: Infix to Postfix

)) / (F + G))

- stack: ((* (-
- output: [A B + C E]

Stacks: Infix to Postfix

) / (F + G))

- stack: ((*
- output: [A B + C E -]

Stacks: Infix to Postfix

/ (F + G))

▲

- stack: (
- output: [A B + C E - *]

Stacks: Infix to Postfix

(F + G))

- stack: (/
- output: [A B + C E - *]

Stacks: Infix to Postfix

F + G))

▲

- stack: (/ (
- output: [A B + C E - *]

Stacks: Infix to Postfix

+ G))

▲

- stack: (/ (
- output: [A B + C E - * F]

Stacks: Infix to Postfix

G))

- stack: (/ (+
- output: [A B + C E - * F]

Stacks: Infix to Postfix

))

- stack: (/ (+
- output: [A B + C E - * F G]

Stacks: Infix to Postfix

)

- stack: (/
- output: [A B + C E - * F G +]

Stacks: Infix to Postfix

- stack: <empty>
- output: [A B + C E - * F G + /]