

Infix -> expression tree

Zdroj:

http://en.wikipedia.org/wiki/Shunting-yard_algorithm

Níže přeložený a upravený postup neobsahuje práci s funkcemi – jestli je to podporováno (jakože asi ano), mohlo by se časem doplnit

Vytvořil: Michal Kozubík, xkozub03, FIT VUTBR

Příprava

- První token už je načtený (o to se postarala funkce `get_token()` v hlavním těle parseru),
- Algoritmus tedy prvně proběhne a pak až načte další token
- ZOP – zásobník uzlů s operátory
- ZUZ – zásobník uzlů s operandy

Shunting yard algorithm

- Pokud je token operand, vytvoř z něj uzel a vlož jej na zásobník ZUZ
- Pokud je token operátor (OP1):
 - **Dokud** je na zásobníku ZOP operand se stejnou nebo vyšší prioritou než OP1, vyjmi ho ze zásobníku a vlož mu potomky ze zásobníku ZUZ (prvně pravý, potom levý), a tento uzel opět vlož na vrchol zásobníku ZUZ
 - Vytvoř z OP1 uzel a vlož jej na zásobník ZOP
- Pokud je token **levá** závorka, vytvoř z ní uzel a vlož ji na zásobník ZOP
- Pokud je token **pravá** závorka:
 - **Dokud** na vrcholu zásobníku ZOP nebude **levá** závorka, vyjmi ze ZOP uzel, vlož mu potomky ze ZUZ (pravý, levý) a tento nový uzel vlož na vrchol ZUZ
 - Vyjmi **levou** závorku ze ZOP, nikam ji nekládej
 - Pokud si nenašel **levou** závorku a ZOP je prázdný, je vstup chybně uzávorkován
- Pokud si narazil na jakýkoli jiný token (už pravděpodobně nesouvisí s výrazem):
 - **Dokud** není ZOP prázdný:
 - Pokud je na vrcholu závorka, je výraz špatně uzávorkován
 - Vyjmi uzel ze ZOP, dej mu potomky ze ZUZ (pravý, levý) a tento uzel vlož zpět na ZUZ
 - Ukonči načítání
- Načti token

$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:
- ZOP:
- ZUZ:
- Exptree:

$$\frac{(A+B)}{(C+D)} + \frac{(E-F-G)}{(H+J)}$$

- VSTUP: (
- ZOP: (
- ZUZ:
- Exptree:

$$(\underline{A}+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP: A
- ZOP: (
- ZUZ: A
- Exptree:

$$(A \pm B) / (C + D) + (E - F - G) / (H + J)$$

- VSTUP: +
- ZOP: (+
- ZUZ: A
- Exptree:

$$(A+\underline{B})/(C+D)+(E-F-G)/(H+J)$$

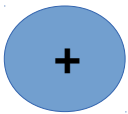
- VSTUP: B
- ZOP: (+
- ZUZ: A B
- Exptree:

$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP: (+
- ZUZ: A B
- Exptree:

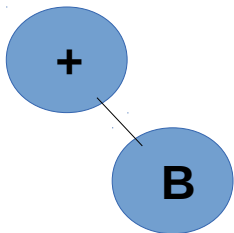
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP: (
- ZUZ: A B
- Exptree:



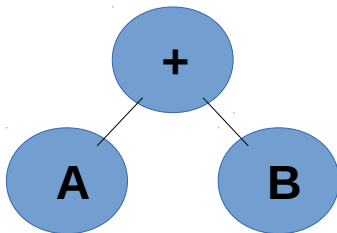
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP: (
- ZUZ: A
- Exptree:



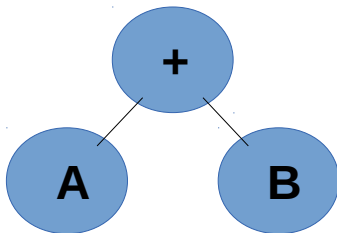
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP: (
- ZUZ:
- Exptree:



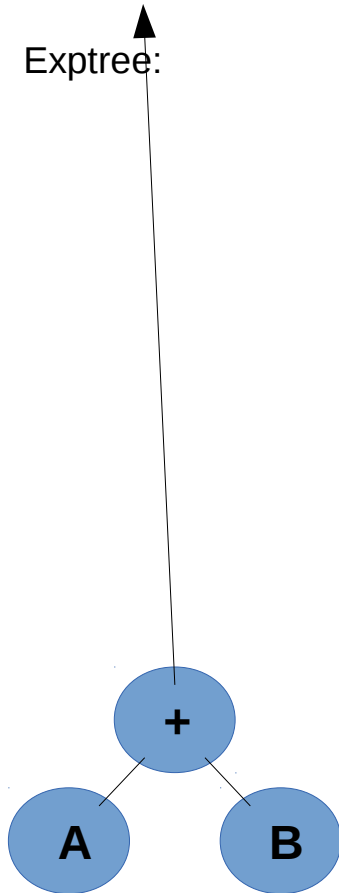
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP:
- ZUZ:
- Exptree:



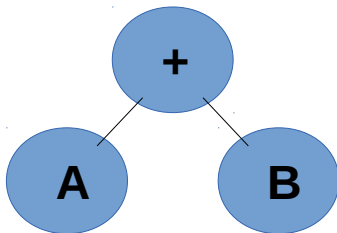
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP:
- ZUZ: +
- Exptree:



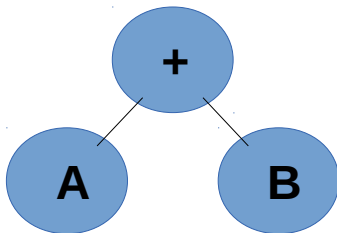
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP: /
- ZOP: /
- ZUZ: +
- Exptree:



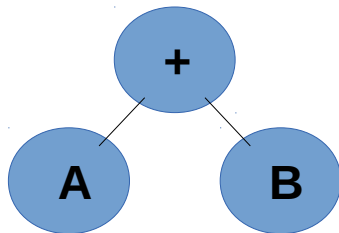
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP: (
- ZOP: / (
- ZUZ: +
- Exptree:



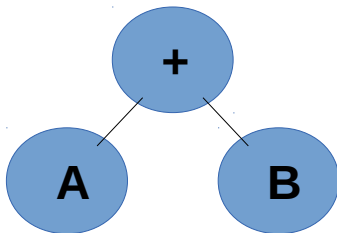
$$(A+B)/(\underline{C}+D)+(E-F-G)/(H+J)$$

- VSTUP: C
- ZOP: / (
- ZUZ: + C
- Exptree:



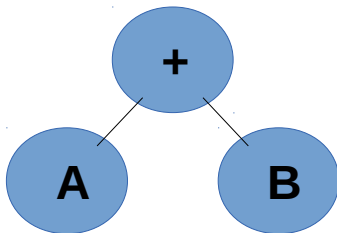
$$(A+B)/(C\pm D)+(E-F-G)/(H+J)$$

- VSTUP: +
- ZOP: / (+
- ZUZ: + C
- Exptree:



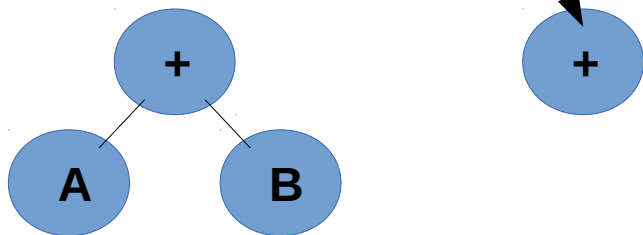
$$(A+B)/(C+\underline{D})+(E-F-G)/(H+J)$$

- VSTUP: D
- ZOP: / (+
- ZUZ: + C D
- Exptree:



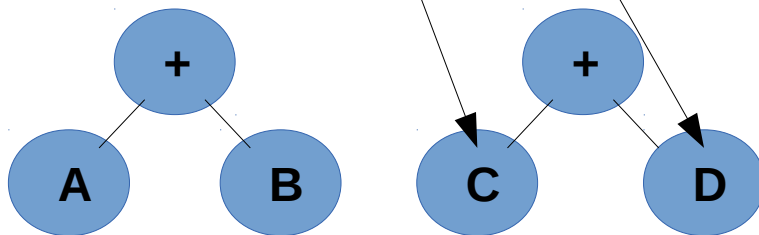
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP: / (+
- ZUZ: + C D
- Exptree:



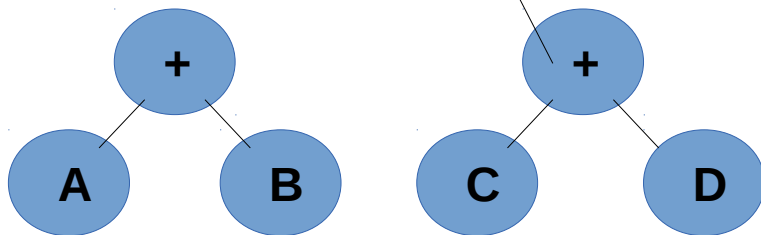
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP: / (
- ZUZ: + C D
- Exptree:



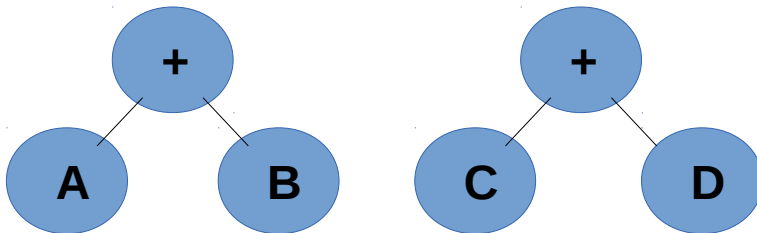
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP: / (
- ZUZ: + +
- Exptree:



$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

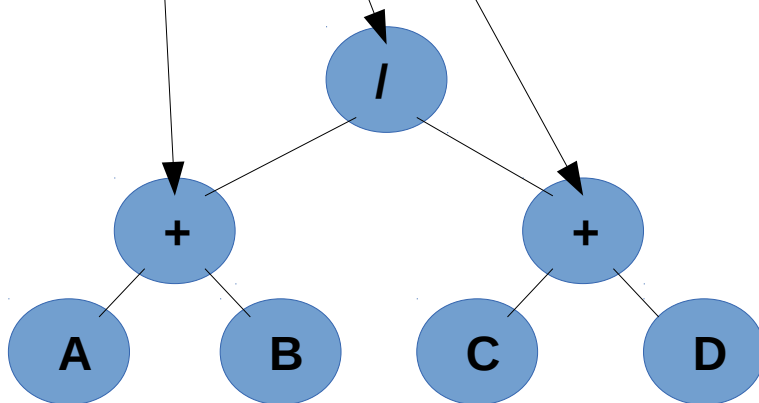
- VSTUP:)
- ZOP: /
- ZUZ: + +
- Exptree:



$$(A+B)/(C+D)\pm(E-F-G)/(H+J)$$

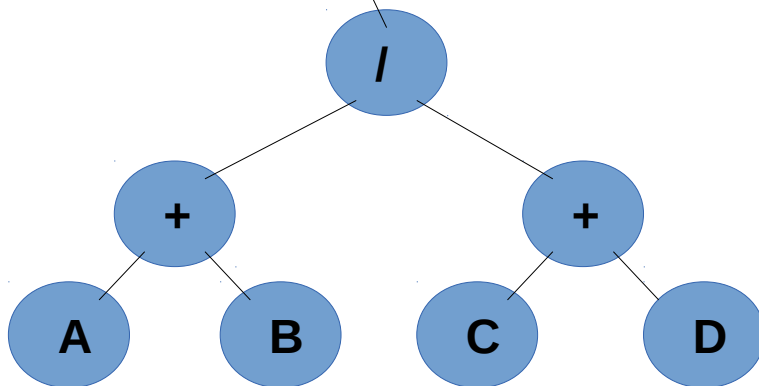
- VSTUP: +
- ZOP: /
- ZUZ: + +
- Exptree:

+ <= /




$$(A+B)/(C+D)\pm(E-F-G)/(H+J)$$

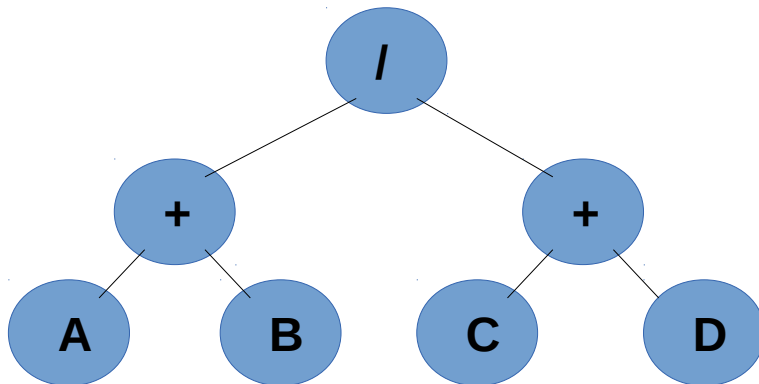
- VSTUP: +
- ZOP:
- ZUZ: /
- Exptree:



$$(A+B)/(C+D)\pm(E-F-G)/(H+J)$$

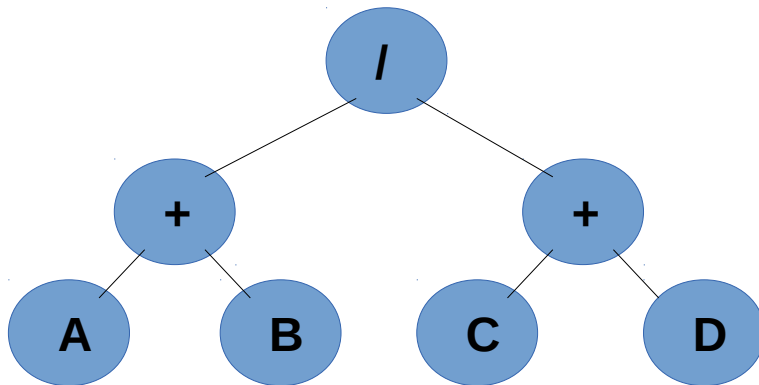
- VSTUP: +
- ZOP: + 
- ZUZ: /
- Exptree:

Stack empty



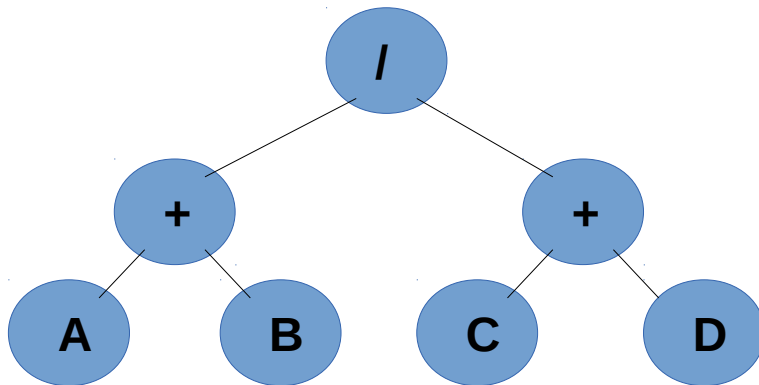
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP: (
- ZOP: + (
- ZUZ: /
- Exptree:



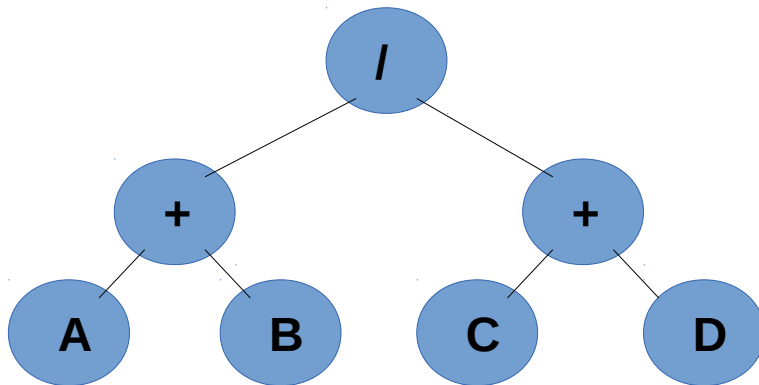
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP: E
- ZOP: + (
- ZUZ: / E
- Exptree:



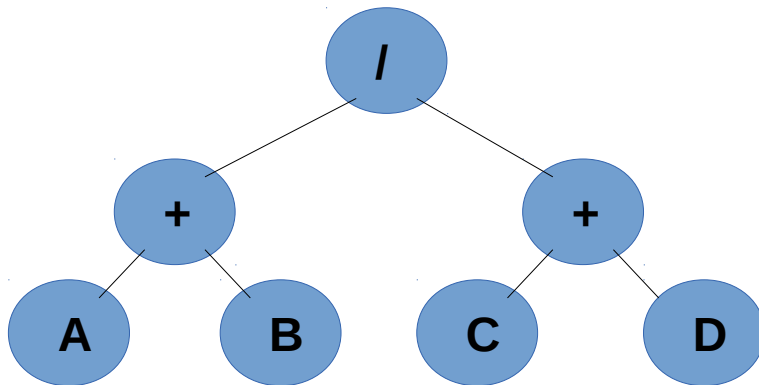
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP: -
- ZOP: + (-
- ZUZ: / E
- Exptree:



$$(A+B)/(C+D)+(E-\underline{F}-G)/(H+J)$$

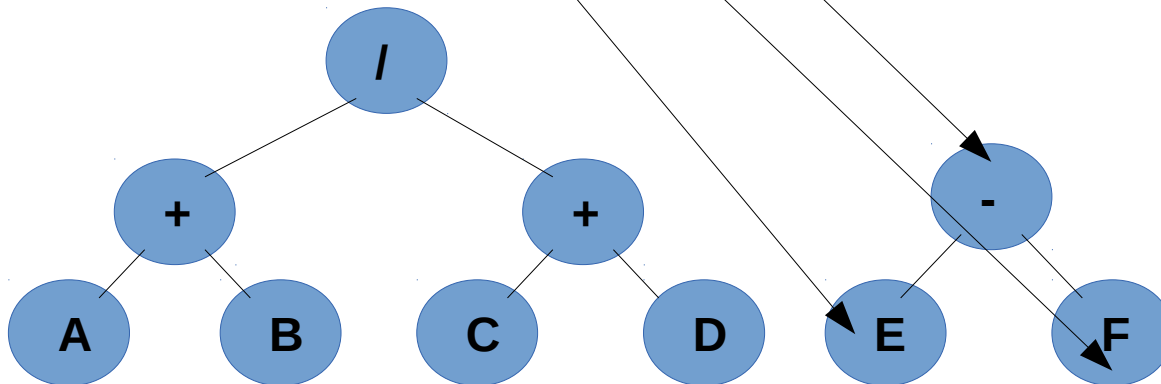
- VSTUP: F
- ZOP: + (-
- ZUZ: / E F
- Exptree:



$$(A+B)/(C+D)+(E-F-\underline{G})/(H+J)$$

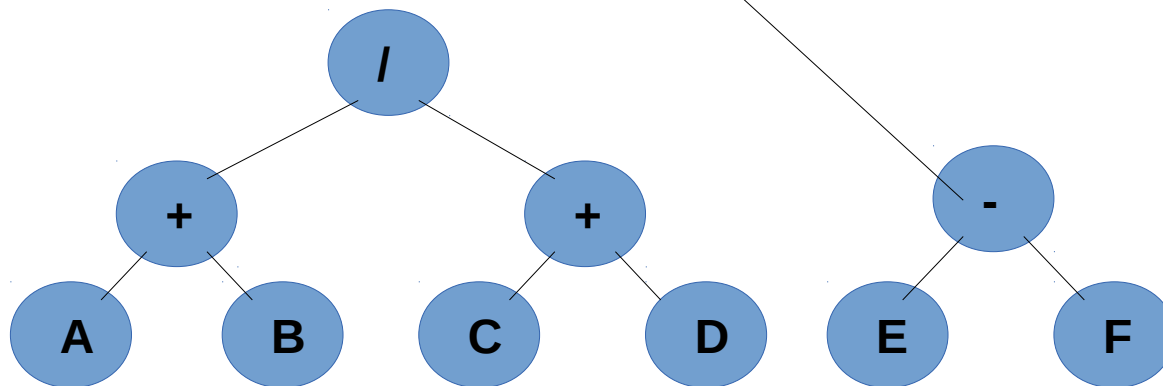
- VSTUP: -
- ZOP: + (-
- ZUZ: / E F
- Exptree:

- <= -



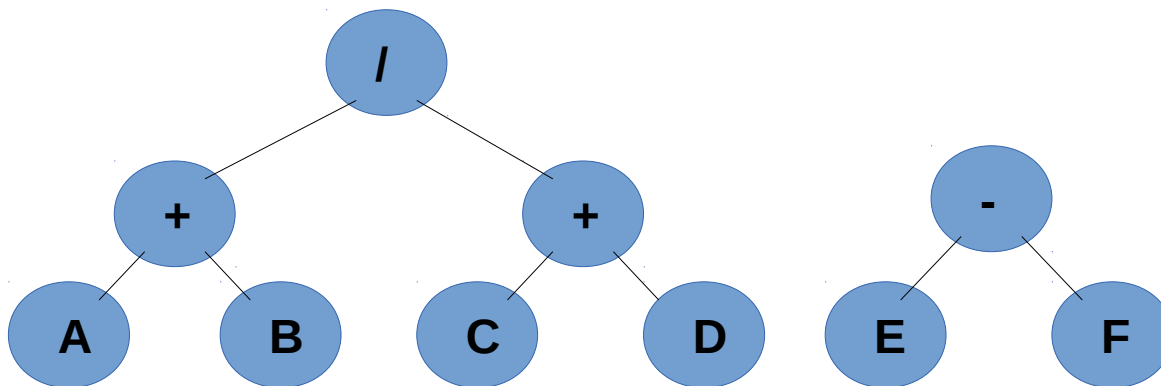
$$(A+B)/(C+D)+(E-F-\underline{G})/(H+J)$$

- VSTUP: -
- ZOP: + (
- ZUZ: / -
- Exptree:



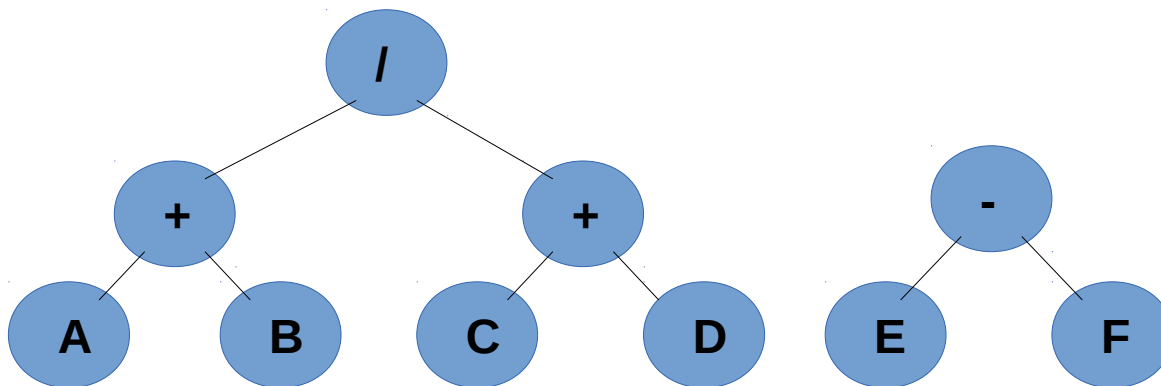
$$(A+B)/(C+D)+(E-F-\underline{G})/(H+J)$$

- VSTUP: -
- ZOP: + (-
- ZUZ: / -
- Exptree:



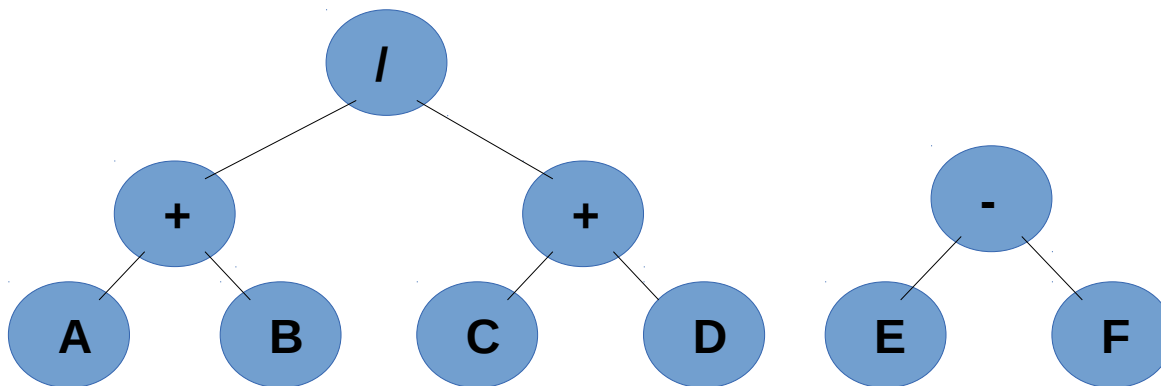
$$(A+B)/(C+D)+(E-F-\underline{G})/(H+J)$$

- VSTUP: G
- ZOP: + (-
- ZUZ: / - G
- Exptree:



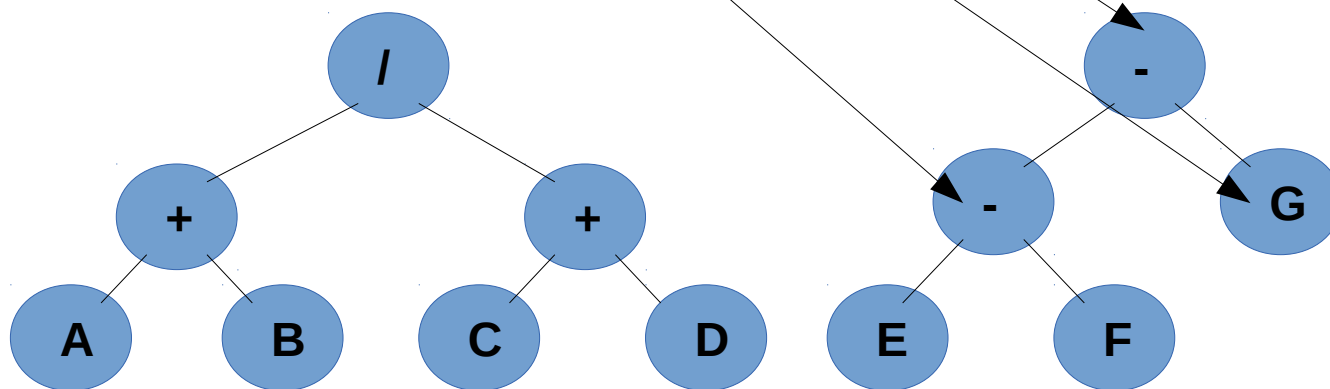
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP: + (-
- ZUZ: / - G
- Exptree:



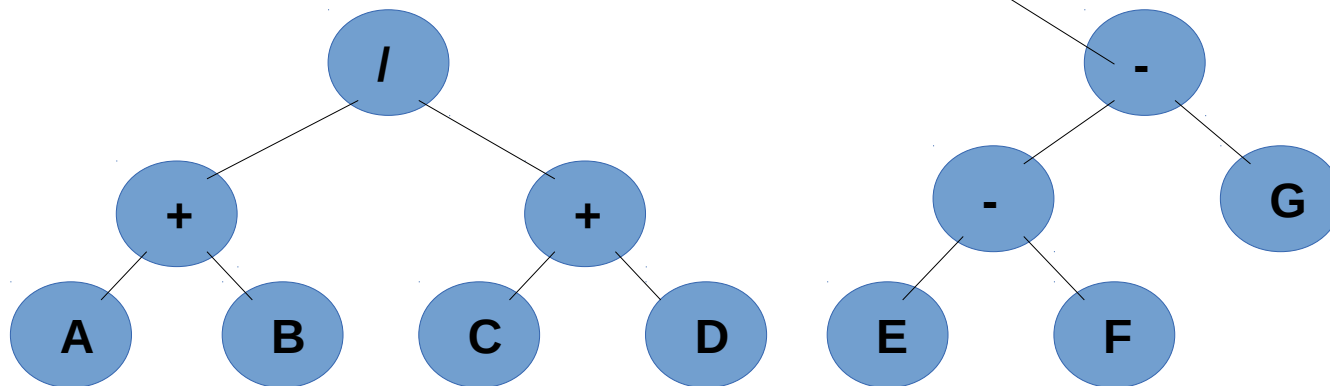
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP: + (-
- ZUZ: / - G
- Exptree:



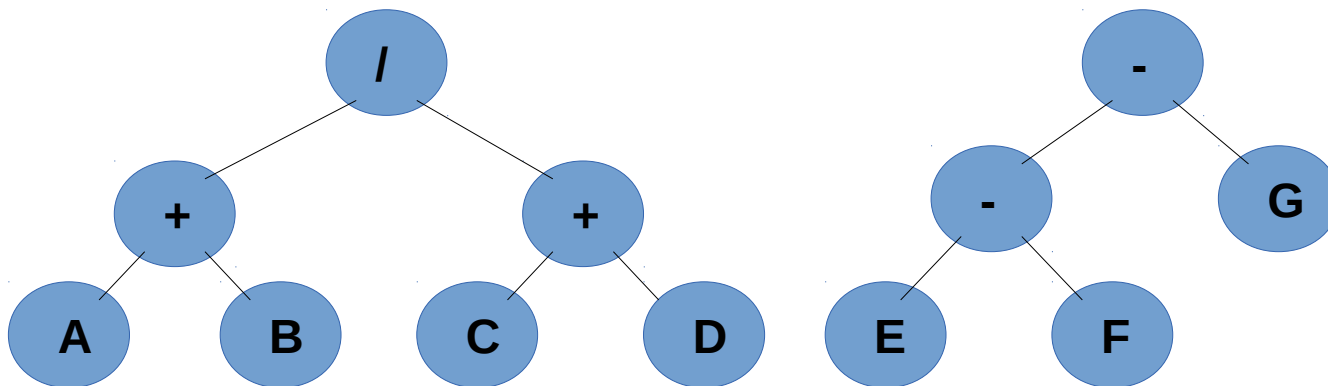
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP: + (
- ZUZ: / -
- Exptree:




$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

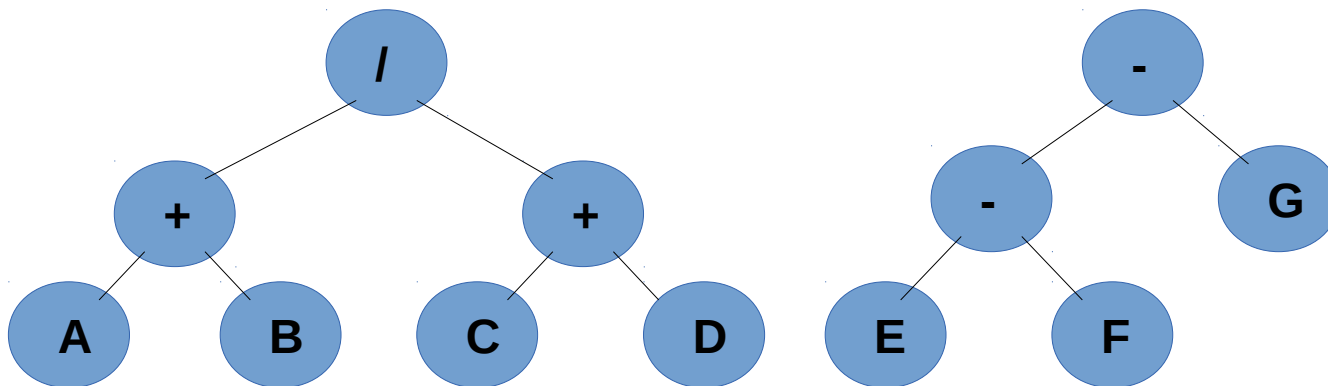
- VSTUP:)
- ZOP: +
- ZUZ: / -
- Exptree:



$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

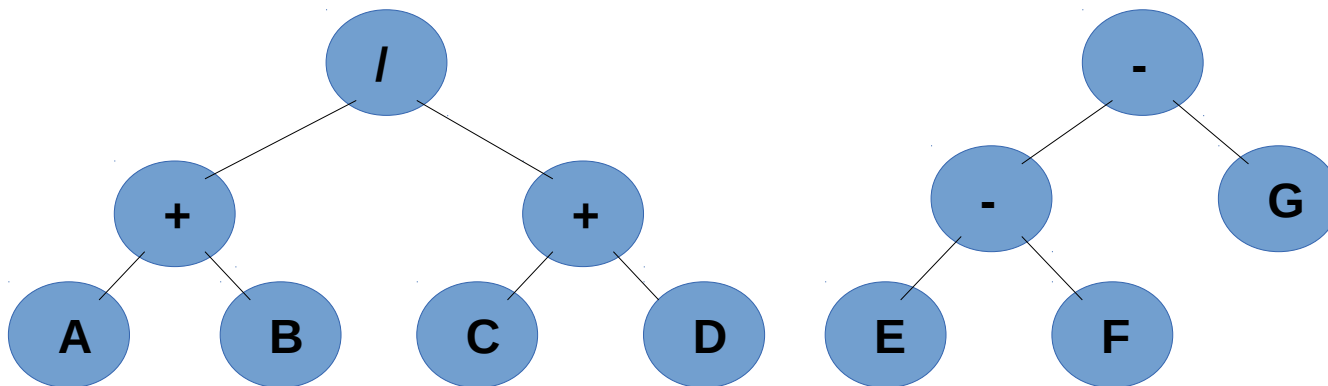
- VSTUP: /
- ZOP: + / 
- ZUZ: / -
- Exptree:

/ >= +



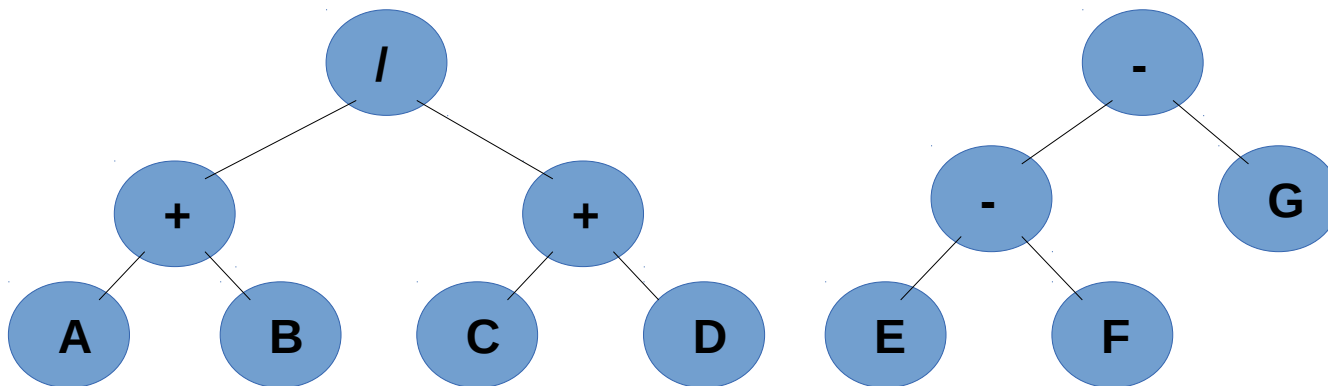
$$(A+B)/(C+D)+(E-F-G)/(\underline{H}+J)$$

- VSTUP: H
- ZOP: + / (
- ZUZ: / - H
- Exptree:



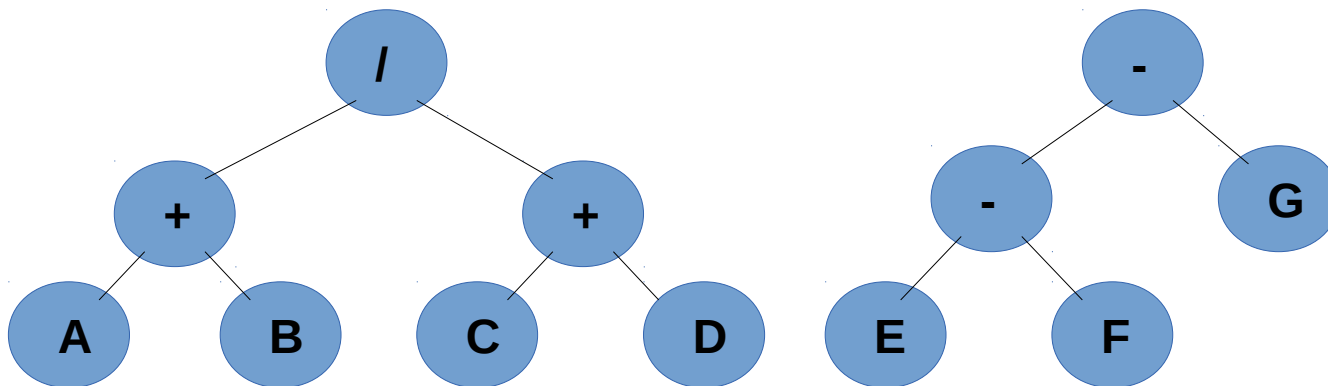
$$(A+B)/(C+D)+(E-F-G)/(H\pm J)$$

- VSTUP: +
- ZOP: + / (+
- ZUZ: / - H
- Exptree:



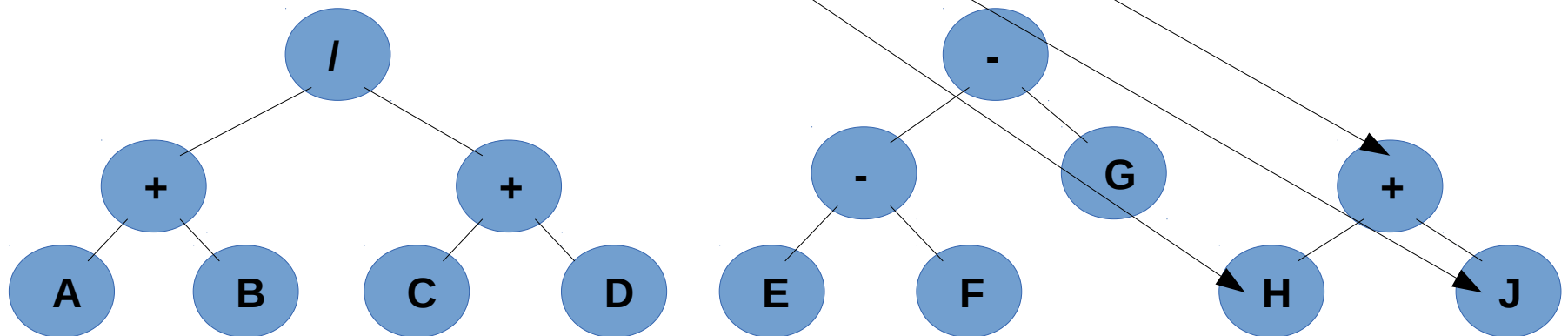
$$(A+B)/(C+D)+(E-F-G)/(H+\underline{J})$$

- VSTUP: J
- ZOP: + / (+
- ZUZ: / - H J
- Exptree:



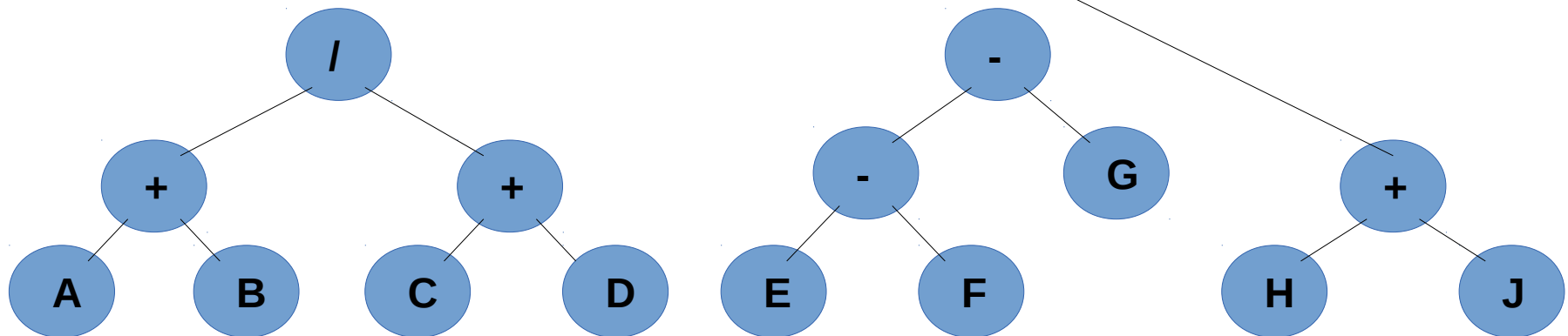
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP: + / (+
- ZUZ: / - H J
- Exptree:



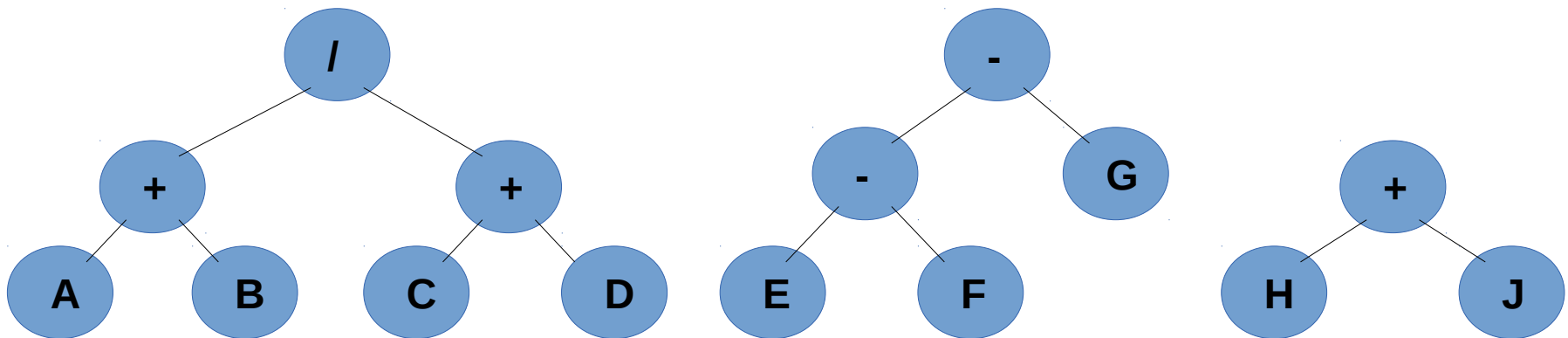
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP:)
- ZOP: + / (
- ZUZ: / - +
- Exptree:



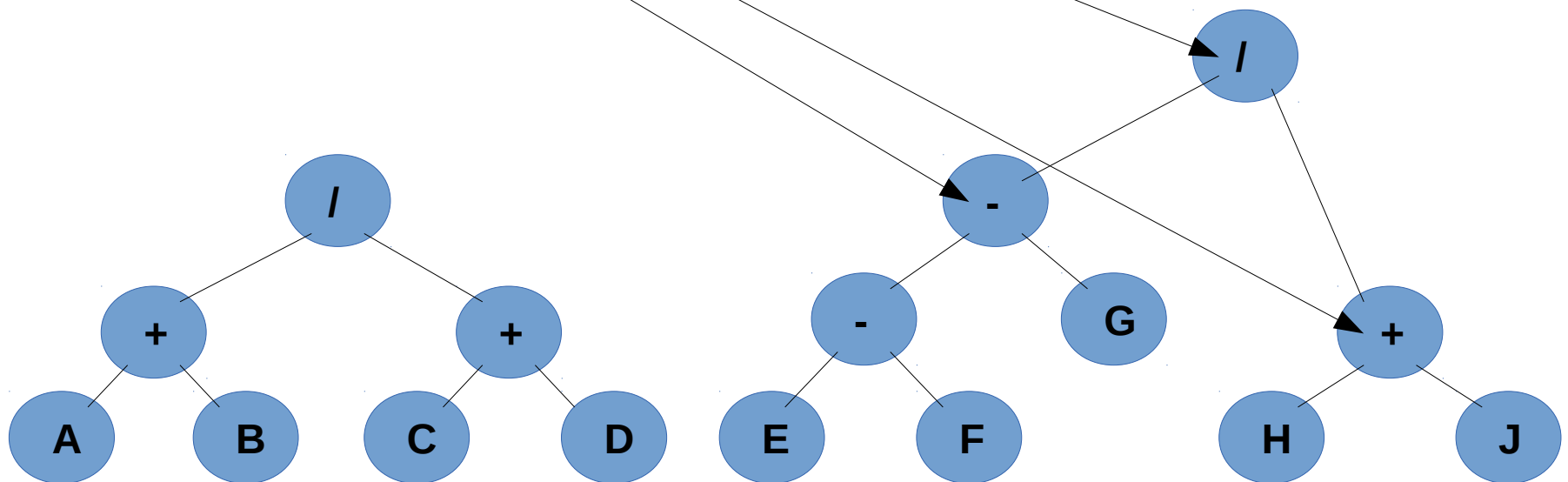
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP: END
- ZOP: + /
- ZUZ: / - +
- Exptree:



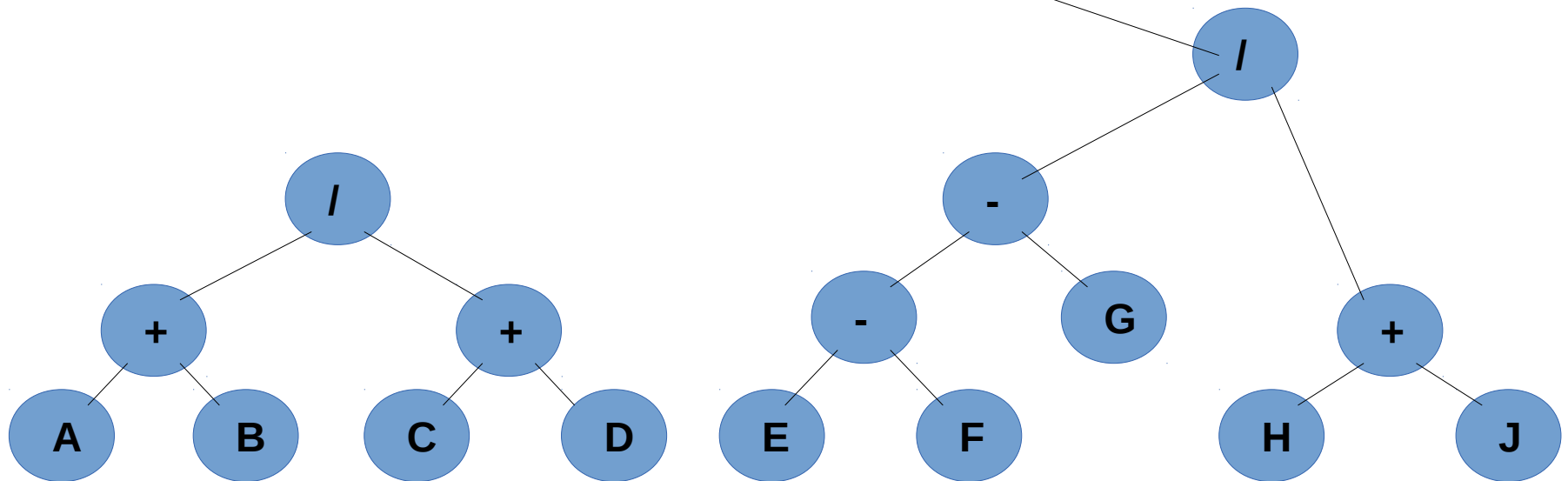
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP: END
- ZOP: + /
- ZUZ: / - +
- Exptree:



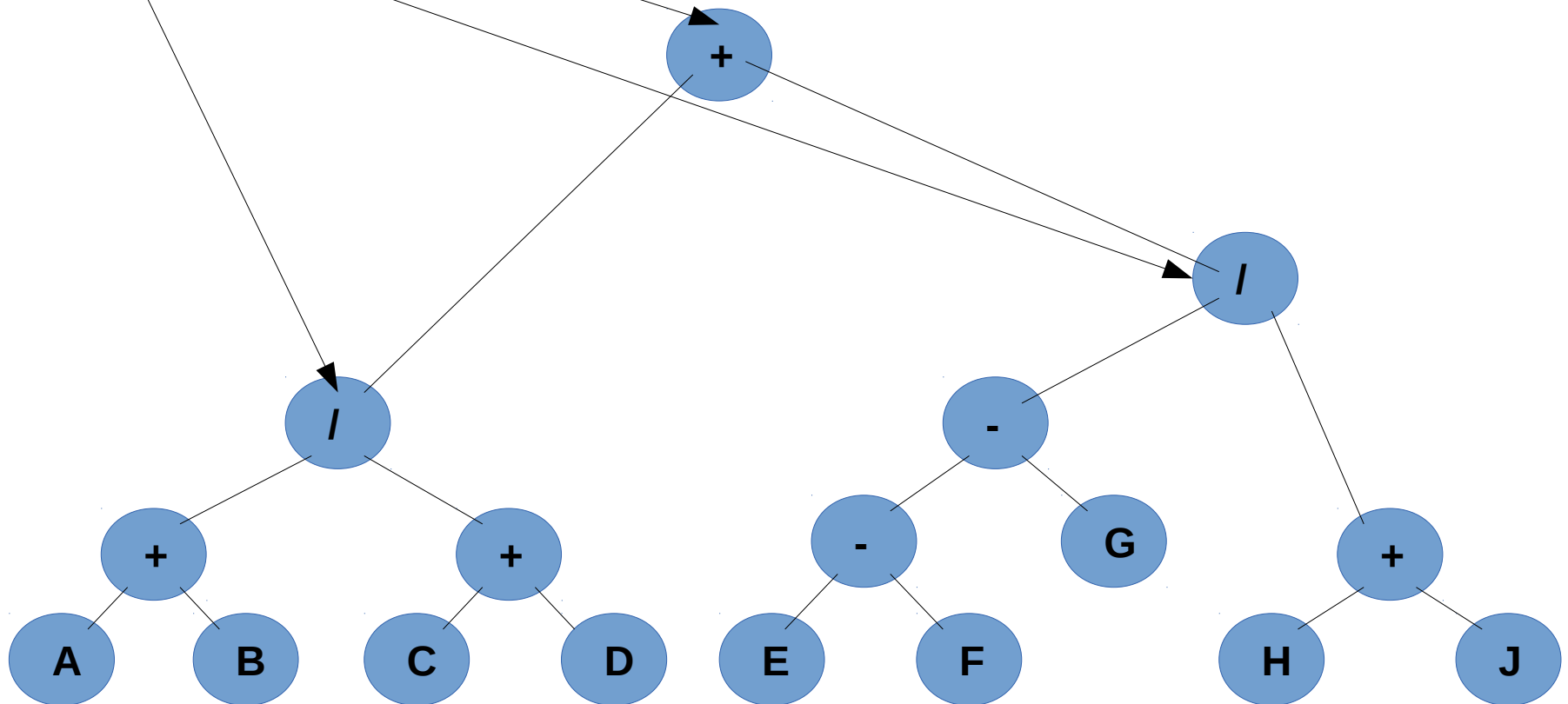
$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP: END
- ZOP: +
- ZUZ: //
- Exptree:



$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP: END
- ZOP: +
- ZUZ: //
- Exptree:



$$(A+B)/(C+D)+(E-F-G)/(H+J)$$

- VSTUP: END
- ZOP: EMPTY
- ZUZ: EMPTY
- Exptree:

