Arbori. Arbori binari

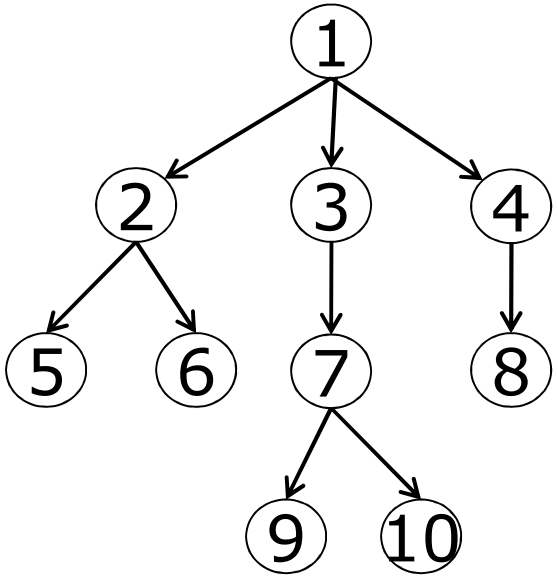
SD 2015/2016

Conținut

- arbori
- arbori binari (**`ArbBin`**)
- aplicaţie: reprezentarea expresiilor ca arbori

Arbori

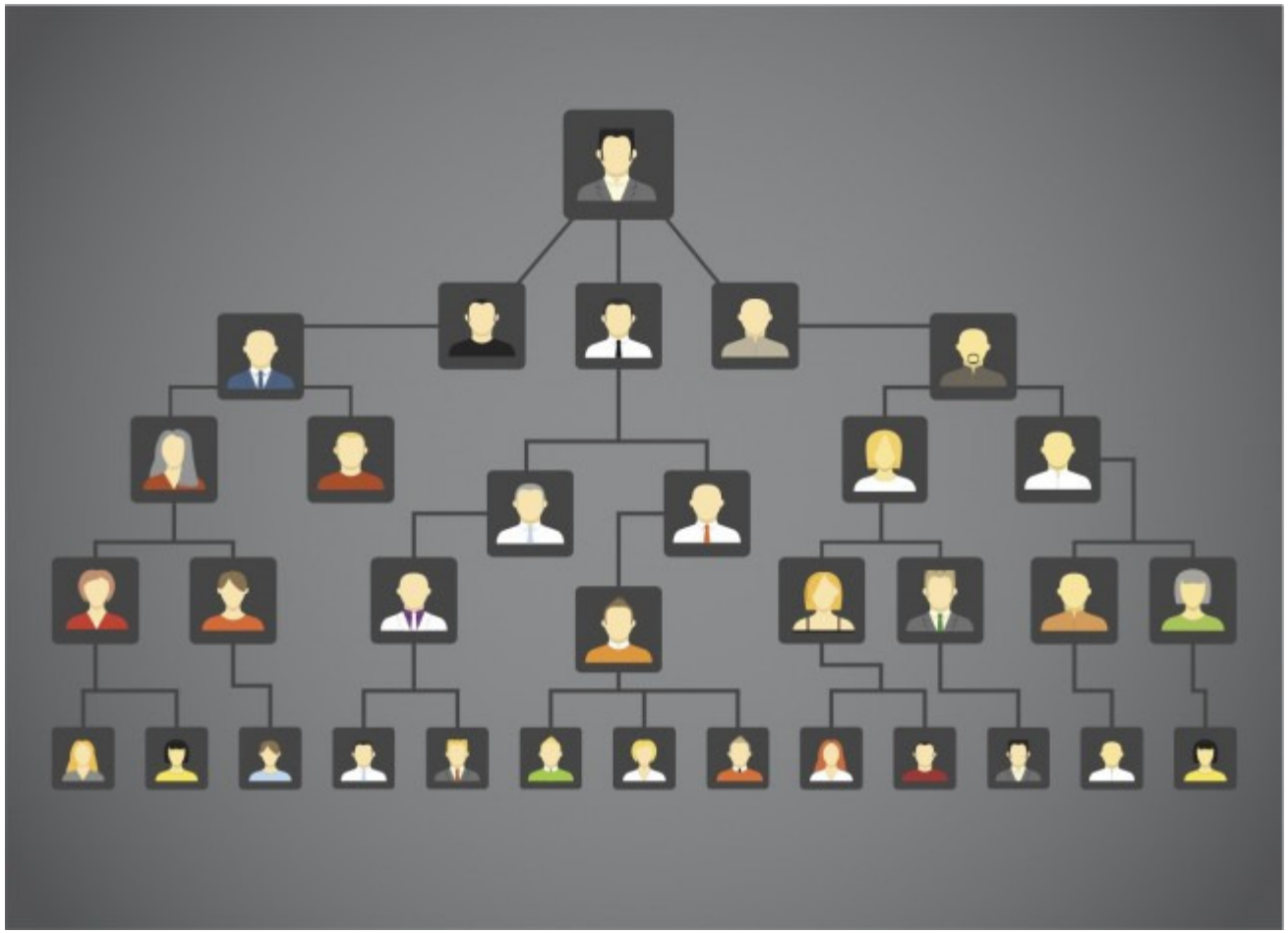
- Model abstract pentru structuri ierarhice;
- Un arbore este format din noduri legate printr-o relație *părinte-copil*
  - ***A = (N, P)***
    - *N* mulțimea de noduri;
    - *P* relație binară peste N, (“*părintele lui*”);
    - $r \in N$, nod rădăcină.
  - $\forall x \in N, \exists$ un singur drum de la *x* la *r*
  - $\forall x \in N - \{r\}$, *x* are un singur părinte.



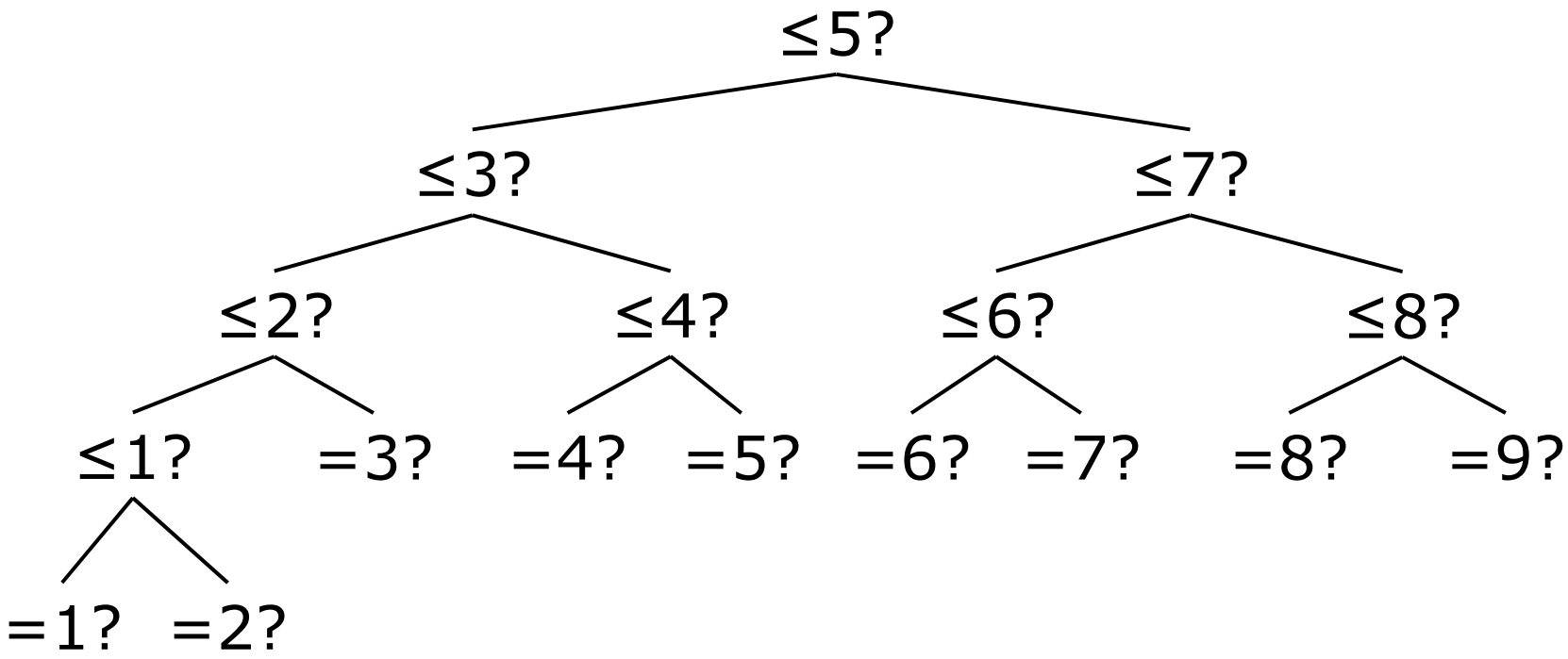
Arbori – aplicații

- Arbori genealogici;
- Colecții de carți în biblioteci;
- Organizarea fișierelor;
- Medii de programare.

Arbori genealogici

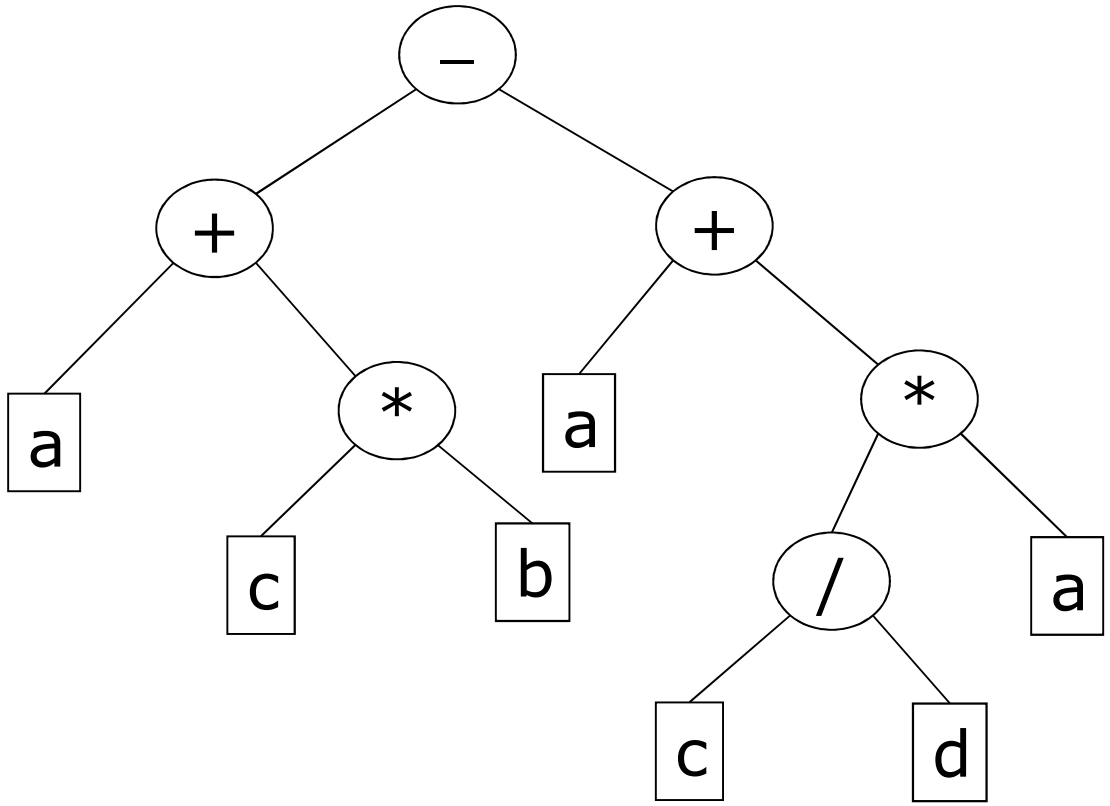


Arbori de decizie



1	2	3	4	5	6	7	8	9

Arbori sintactici

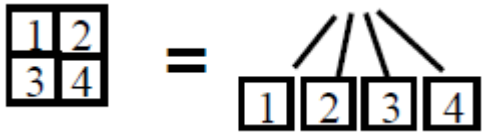


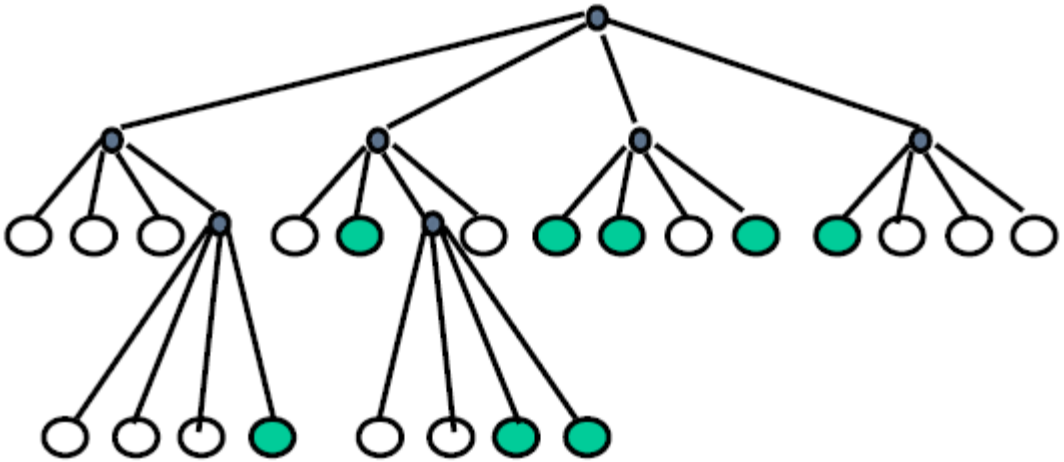
(a + c * b) - (a + c / d * a)

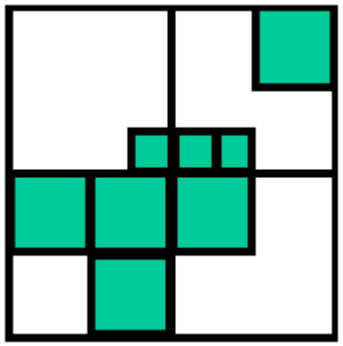
Arbori pătratici

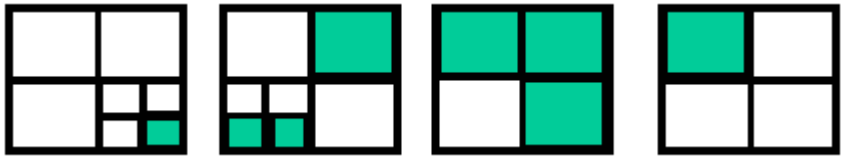
toată imaginea neagră:

toată imaginea albă:

altfel: 



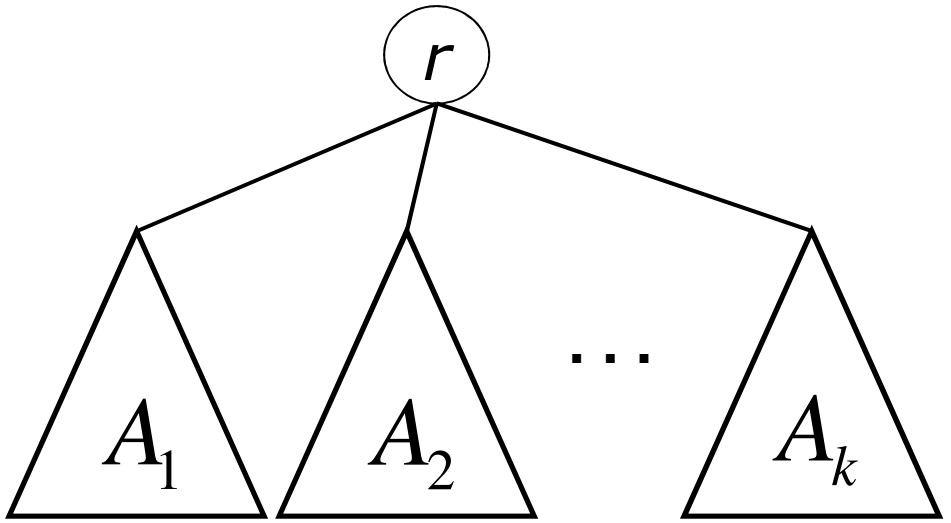




Arbori: definiție recursivă

$$A = \begin{cases} \Lambda, \text{ arborele vid,} \\ (r, \{A_1, ..., A_k\}), \ r \text{ element, } A_1, ..., A_k \text{ arbori} \end{cases}$$

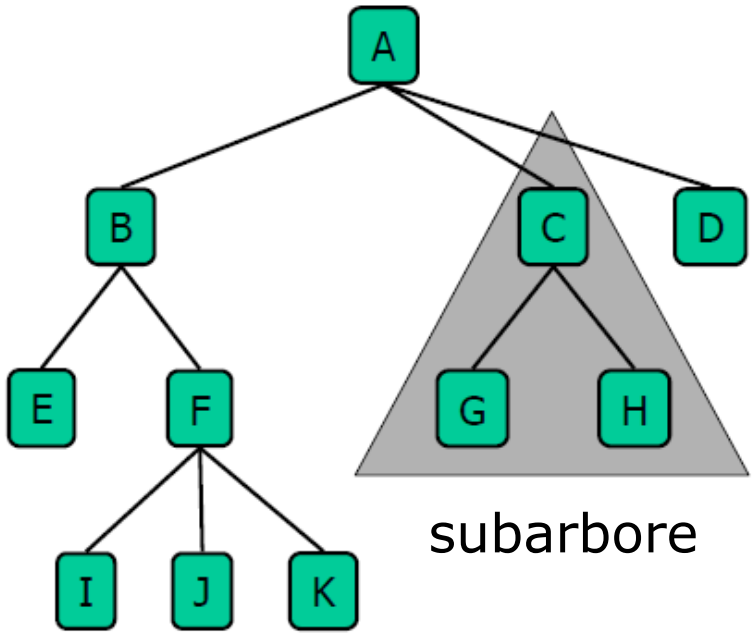
$A = \Lambda$ sau



Dacă *A* este ordonat (planar), atunci

Arbori: terminologie

- Rădăcina: nodul fără părinte
- Nod intern: nod cu cel puțin un fiu
- Nod extern (frunză): nod fără fii
- Descendenții unui nod: fii, nepoți, etc
- Frații unui nod: toate celelalte noduri având același părinte
- Subarbore: arbore format dintr-un nod și descendenții săi



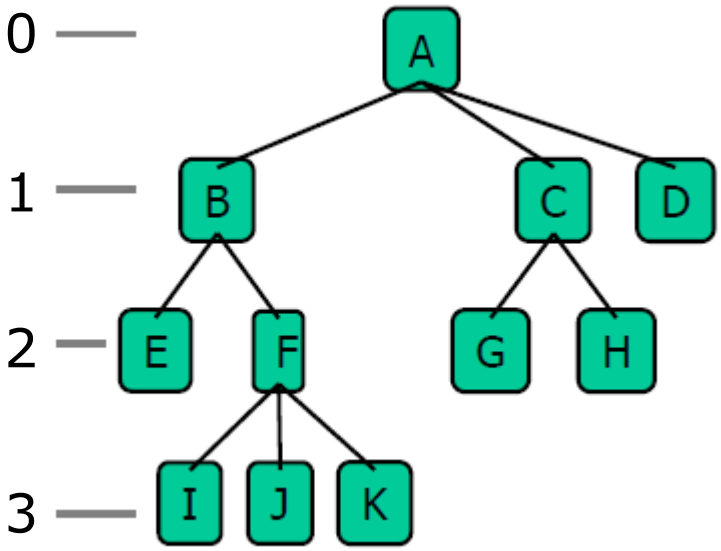
Arbori: terminologie

- Adâncimea unui nod *x*: numărul de noduri de la rădăcină la *x*

$$\text{adâncime}(x) = \begin{cases} 0, & \text{dacă } x \text{ este radacina} \\ 1 + \text{adâncime}(\text{părinte}(x)), & \text{în caz contrar} \end{cases}$$

- Înălțimea unui arbore: adâncimea maximă a nodurilor arborelui

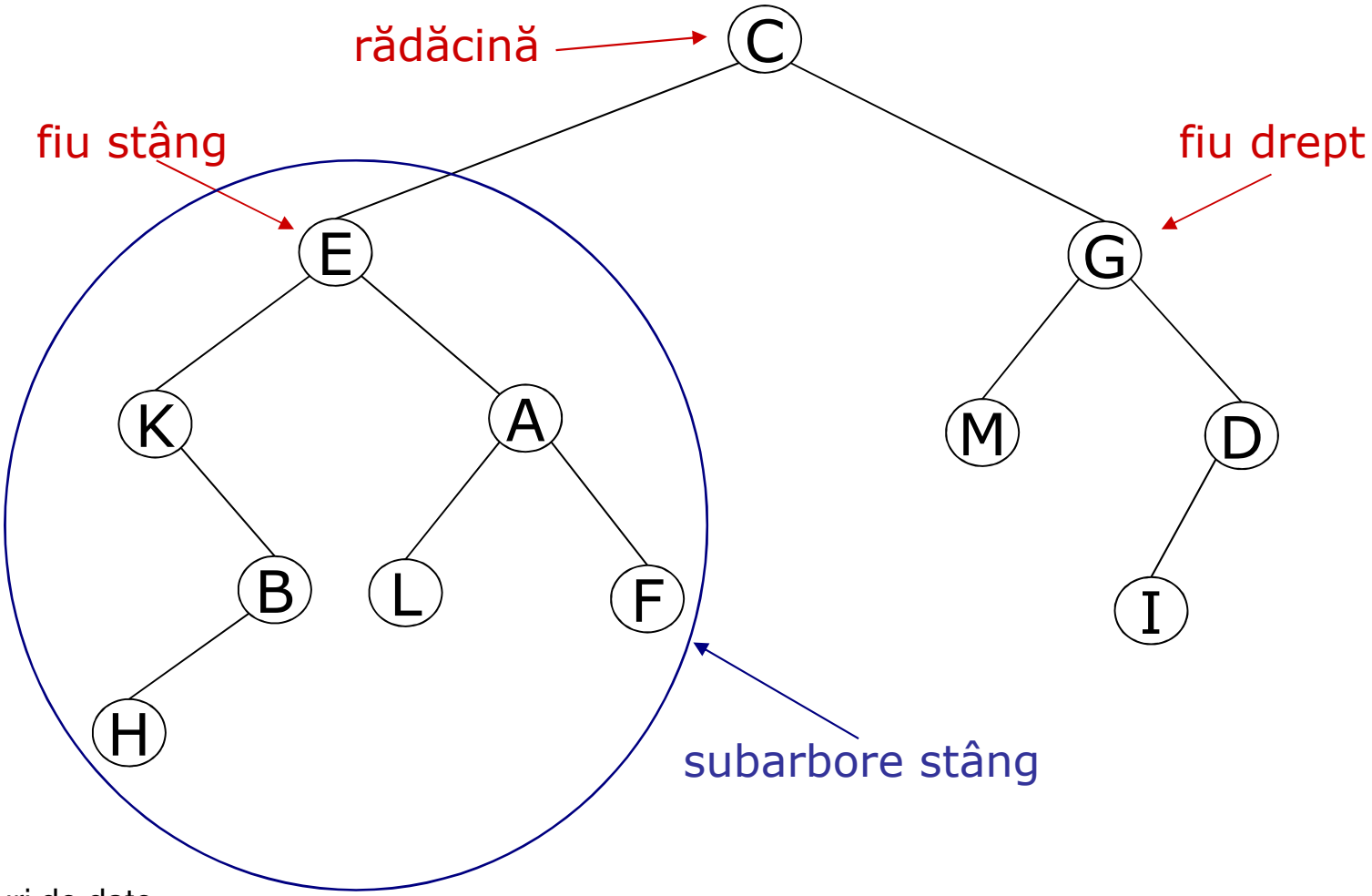
- Înălțimea unui nod *x*: distanța de la *x* la cel mai depărtat descendent al său



Tipul de date abstract **ArbBin**

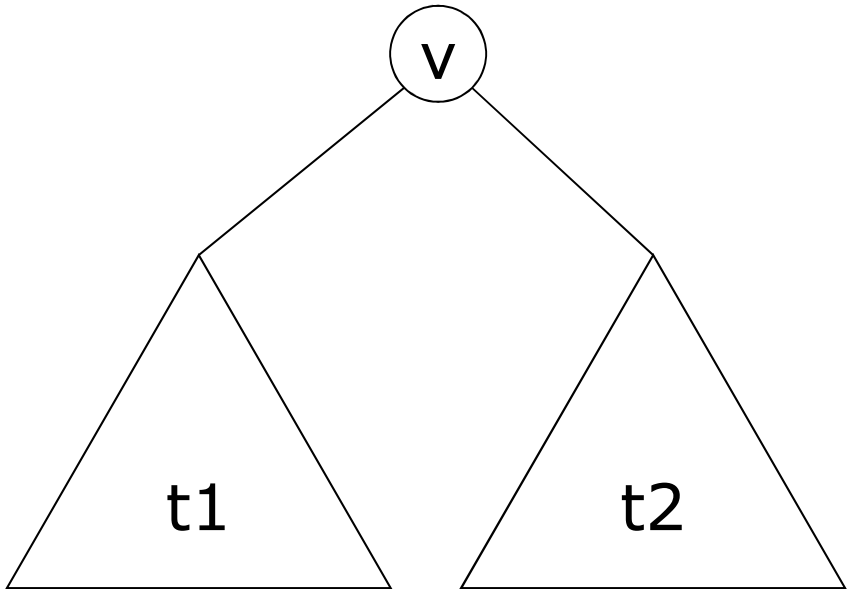
- obiecte : arbori binari.
  - un arbore binar este o colecţie de noduri cu proprietăţile:
    1. orice nod are 0, 1 sau 2 succesori (**fii, copii**);
    2. orice nod, exceptând unul singur – **rădăcina**, are un singur nod predecesor (**tatăl, părintele);**
    3. rădăcina nu are predecesori;
    4. fiii sunt ordonaţi: fiul stâng, fiul drept (daca un nod are un singur fiu, trebuie menţionat care);
    5. nodurile fără fii formează frontiera arborelui.

Arbori binari: exemplu



Arbori binari: definiţia recursivă

- Arborele cu nici un nod (vid) este arbore binar.
- Dacă **v** este un nod şi **t1**, **t2** sunt arbori binari atunci arborele care are pe **v** ca rădăcină, **t1** subarbore stâng al rădăcinii şi **t2** subarbore drept al rădăcinii, este arbore binar.



Arbori binari: proprietăți

- Notații
  - n numărul de noduri
  - n_e numărul de noduri externe
  - n_i numărul de noduri interne
  - h înălțimea

$$h+1 \le n \le 2^{h+1}-1 \qquad 1 \le n_e \le 2^h$$

$$\log_2(n+1)-1 \le h \le n-1 \qquad h \le n_i \le 2^h-1$$

Arbori binari: proprietăți

- Arbore propriu: fiecare nod intern are exact doi fii

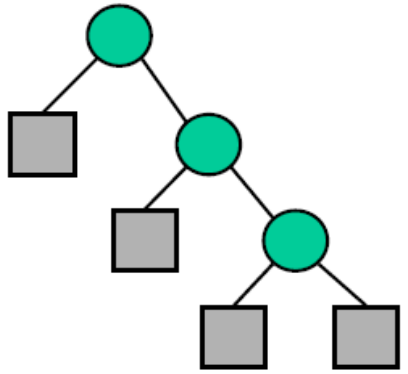
$$h+1 \le n \le 2^{h+1}-1$$

$$\log_2(n+1)-1 \le h \le (n-1)/2$$

$$h+1 \le n_e \le 2^h$$

$$h \le n_i \le 2^h - 1$$

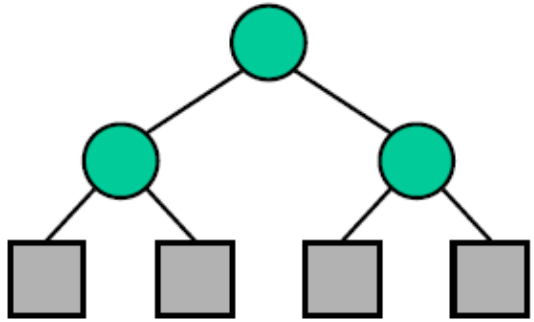
$$n_e = n_i + 1$$



- Arbore complet: arbore propriu în care frunzele au aceeași adâncime

$$\text{nivelul } i \text{ are } 2^i \text{ noduri}$$

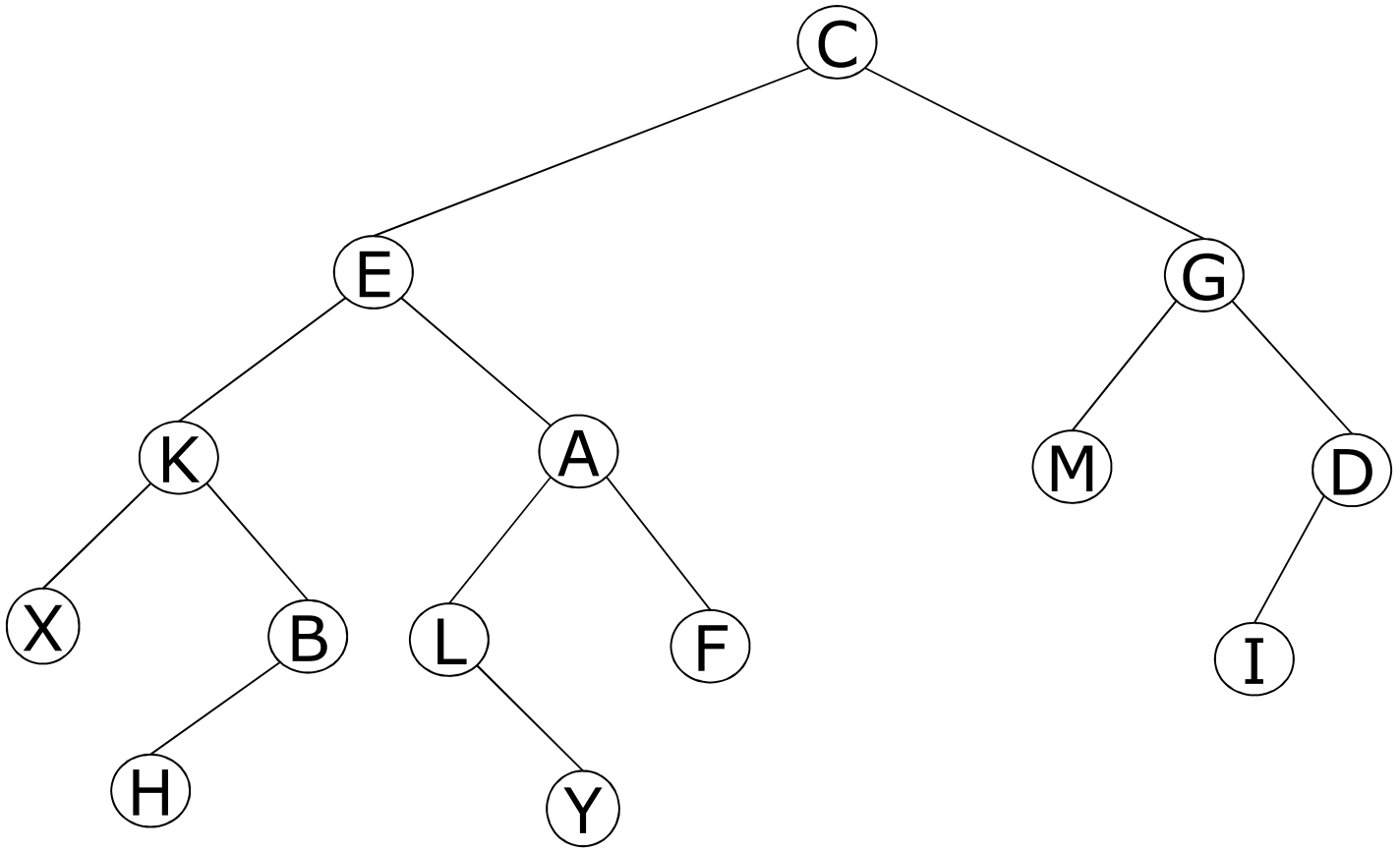
$$n = 2^{h+1} - 1 = 2n_e - 1$$



ArbBin: operaţii

- insereaza()
  - intrare:
    - un arbore binar **t**
    - adresa unui nod cu cel mult un fiu (tatăl noului nod)
    - tipul fiului adăugat (stânga, dreapta)
    - informaţia **e** din noul nod
  - ieşire
    - arborele la care s-a adăugat un nod ce memorează **e**; noul nod nu are fii

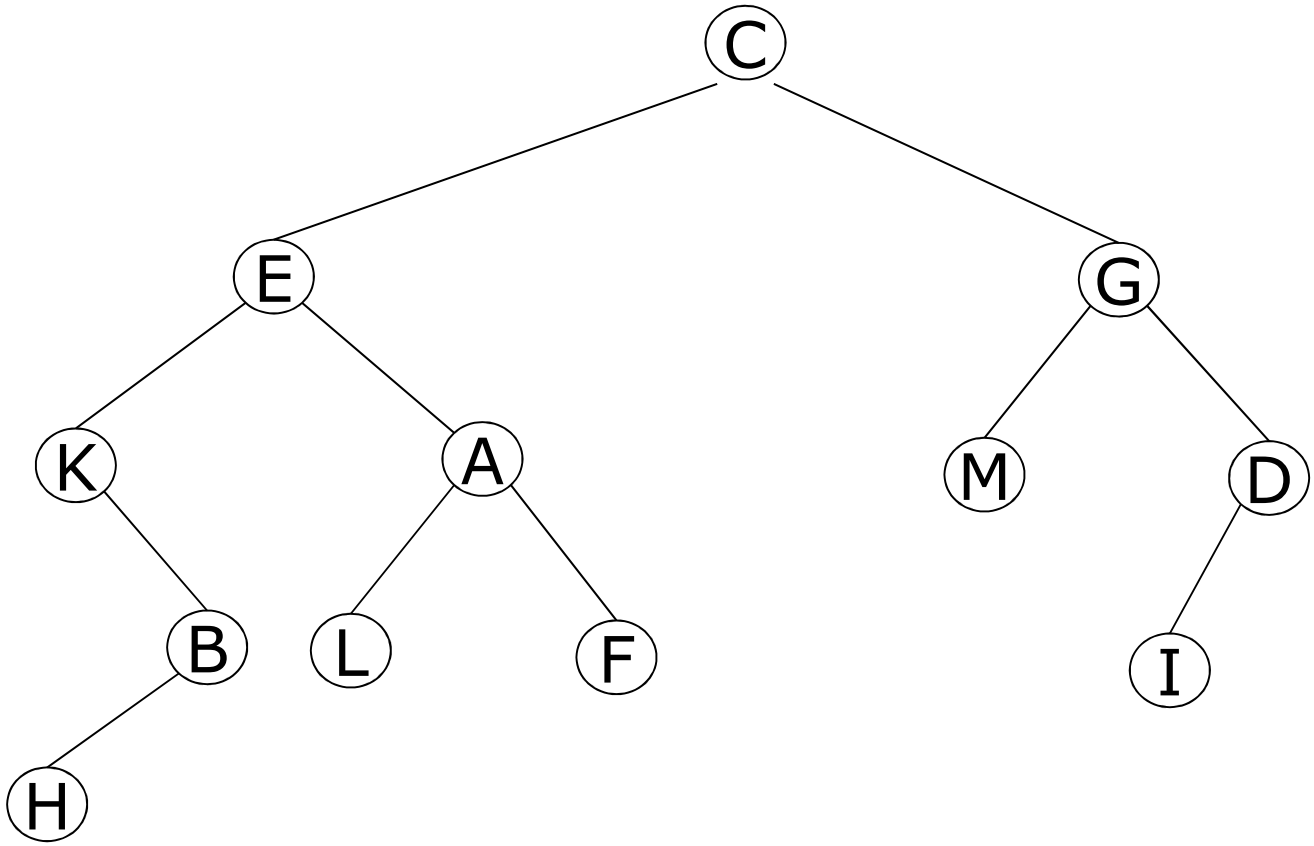
ArbBin: inserare



ArbBin: eliminare

- elimina()
  - intrare:
    - un arbore binar **t**
    - adresa unui nod fără fii şi adresa nodului-tată
  - ieşire
    - arborele din care s-a eliminat nodul dat (de pe frontieră)

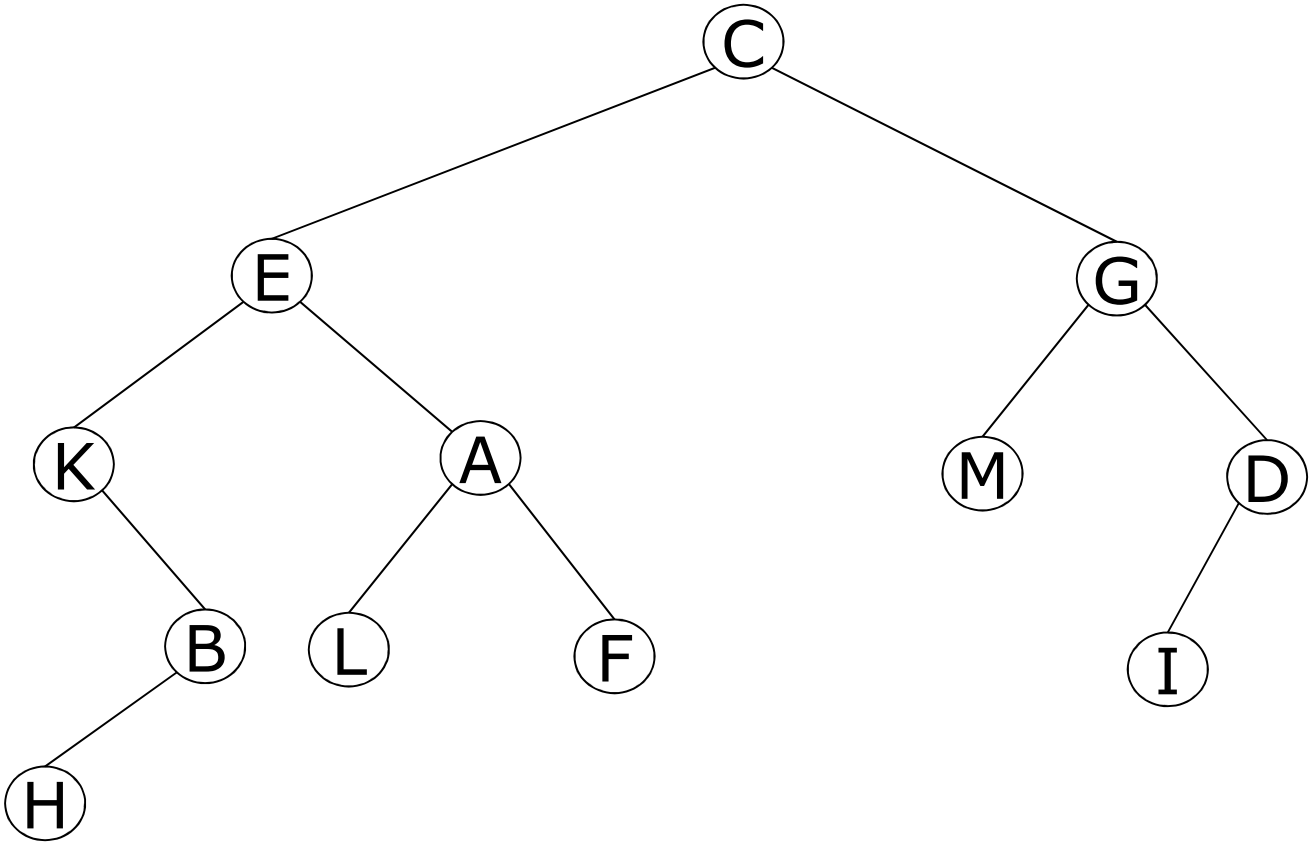
`ArbBin`: eliminare



ArbBin: parcurgere preordine

- parcurgePreordine()
  - intrare
    - un arbore binar **t**
    - o procedură **viziteaza**()
  - ieşire
    - arborele binar **t** dar cu nodurile procesate cu **viziteaza()** în ordinea:
      - rădăcina (R)
      - subarborele stânga (S)
      - subarborele dreapta (D)

Parcurgere preordine - exemplu

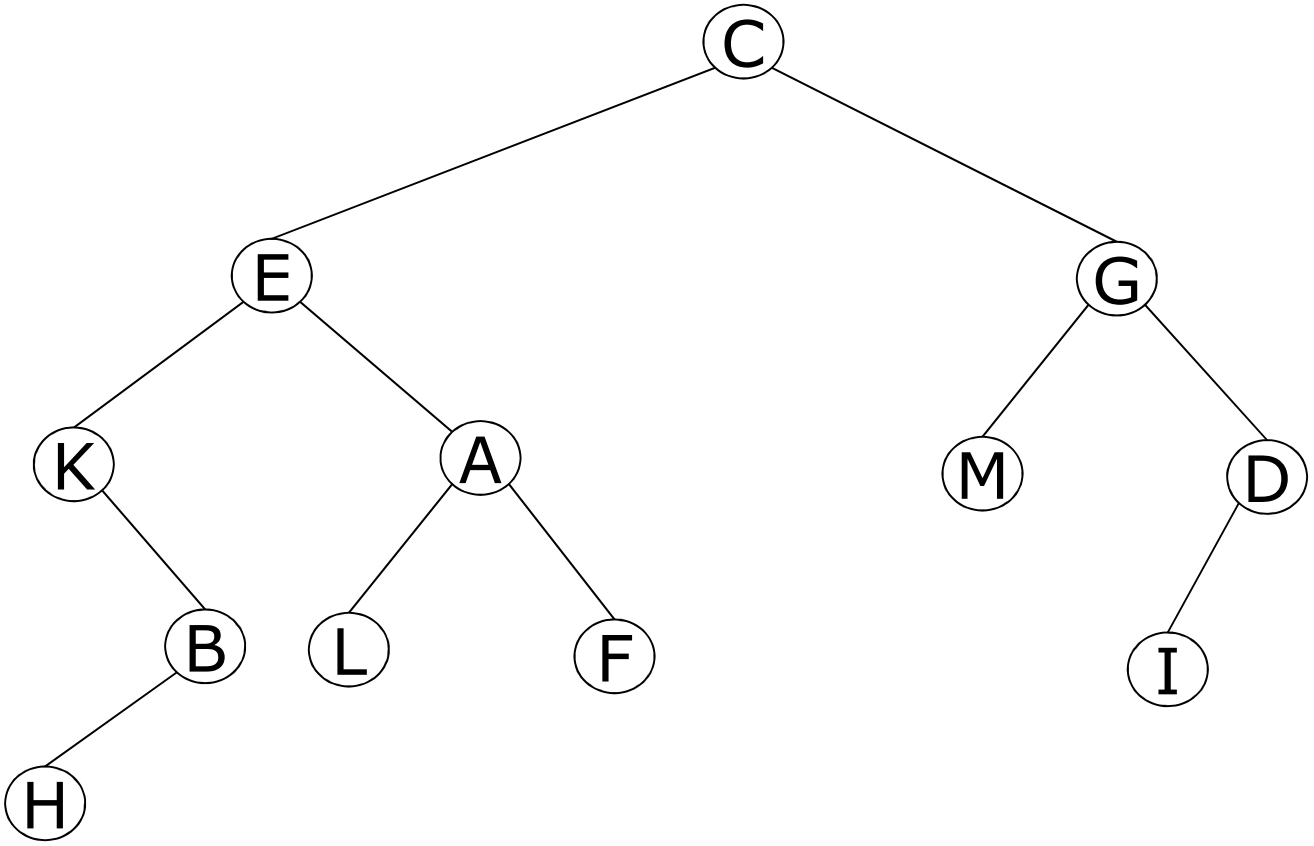


C, E, K, B, H, A, L, F, G, M, D, I

ArbBin: parcurgere inordine

- parcurgeInordine()
  - intrare
    - un arbore binar **t**
    - o procedură **viziteaza()**
  - ieşire
    - arborele binar **t** dar cu nodurile procesate cu **viziteaza()** în ordinea S R D

Parcurgere inordine - exemplu

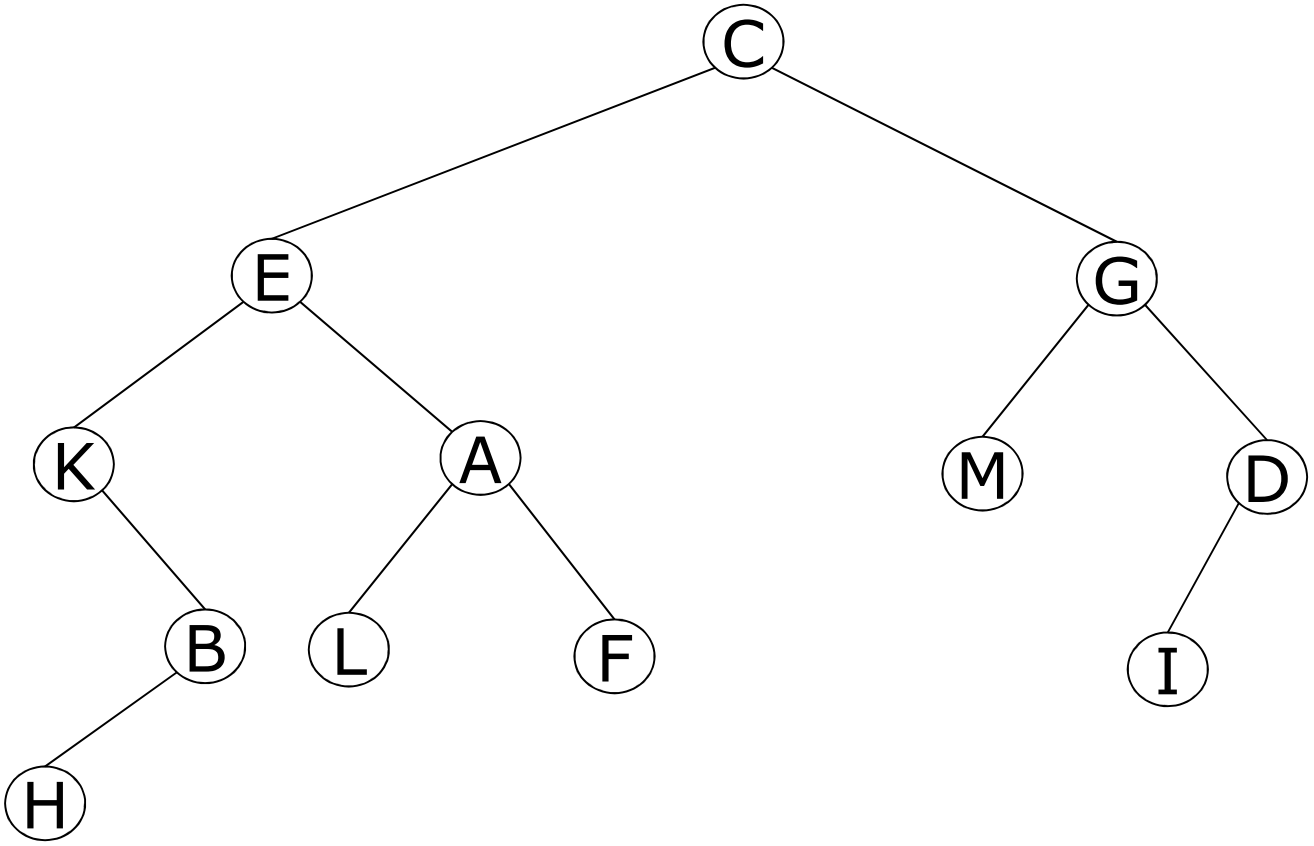


K, H, B, E, L, A, F, C, M, G, I, D

ArbBin: parcurgere postordine

- parcurgePostordine()
  - intrare
    - un arbore binar **t**
    - o procedură **viziteaza()**
  - ieşire
    - arborele binar **t** dar cu nodurile procesate cu **viziteaza()** în ordinea S D R

Parcurgere postordine - exemplu

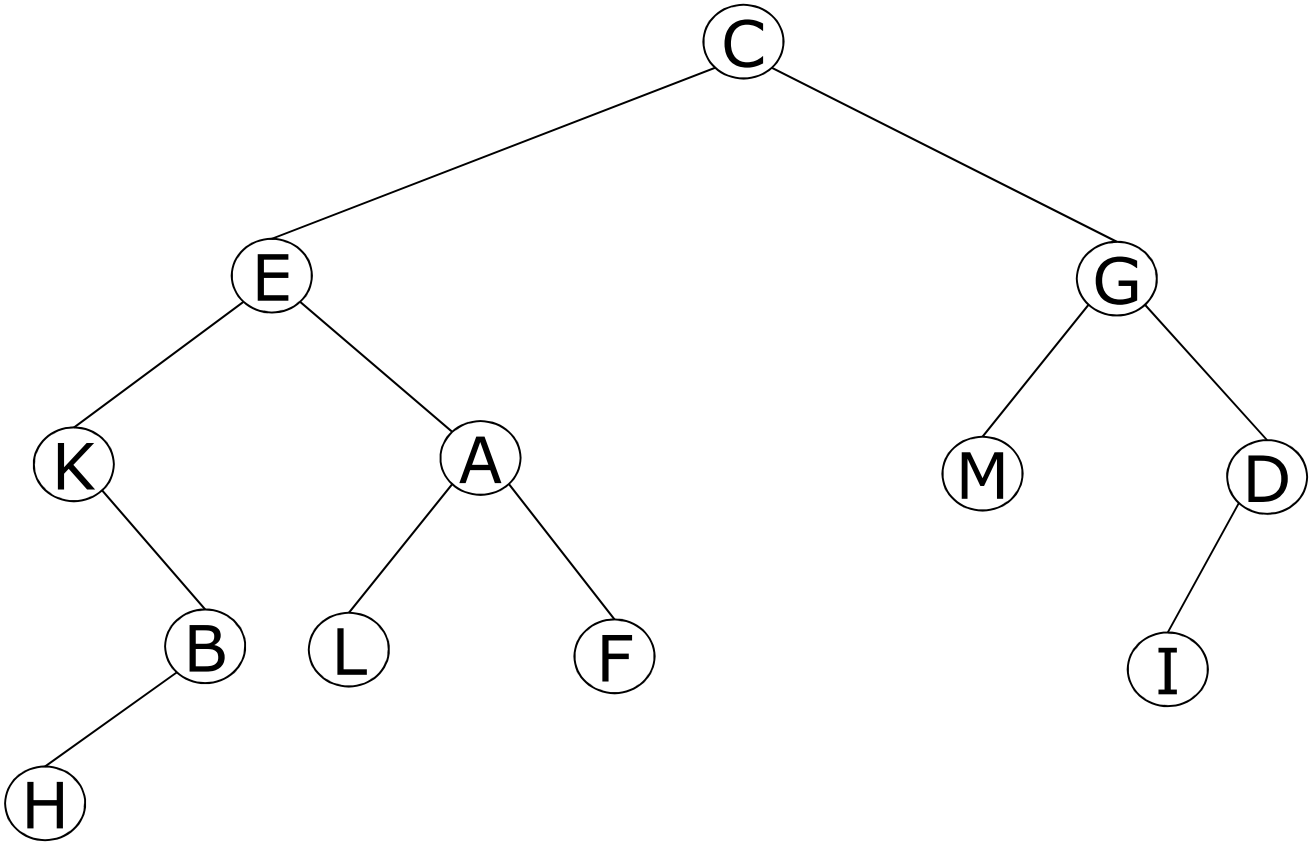


H, B, K, L, F, A, E, M, I, D, G, C

ArbBin: parcurgere BFS

- parcurgeBFS() - Breadth-first search
  - intrare
    - un arbore binar **t**
    - o procedură **viziteaza()**
  - ieşire
    - arborele binar **t** dar cu nodurile procesate cu **viziteaza()** în ordinea BFS (pe niveluri)

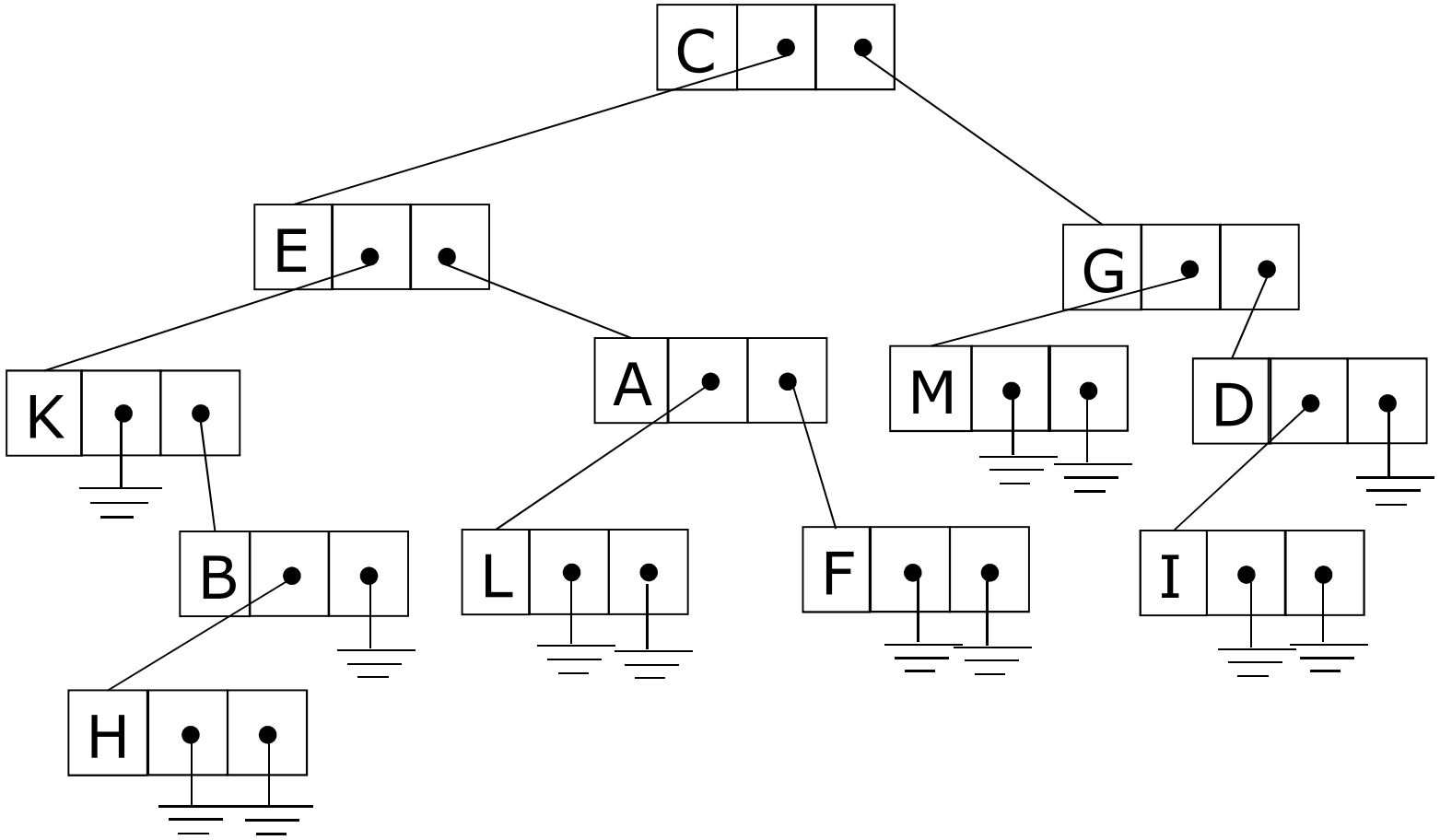
Parcurgere BFS - exemplu



C, E, G, K, A, M, D, B, L, F, I, H

ArbBin: implementare cu structuri înlănţuite

- reprezentarea obiectelor



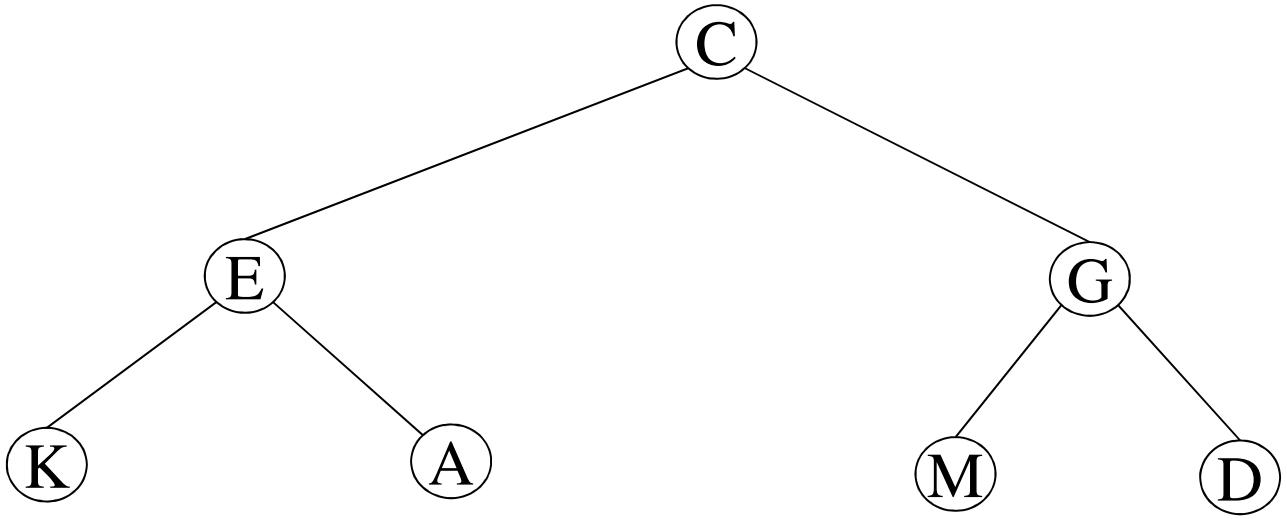
ArbBin: structura unui nod

- un nod **v** (aflat la adresa v) este o structură cu trei câmpuri:
  - **v->inf** /*informaţia memorata în nod*/
  - **v->stg** /*adresa fiului stânga*/
  - **v->drp** /*adresa fiului dreapta*/

ArbBin: parcurgePreordine()

```
procedure parcurgePreordine(v, viziteza)
begin
   if (v == NULL)
        then return
else viziteaza(v)
      parcurgePreordine(v->stg, viziteaza)
      parcurgePreordine(v->drp, viziteaza)
end
```

Implementarea parcurgerii BFS



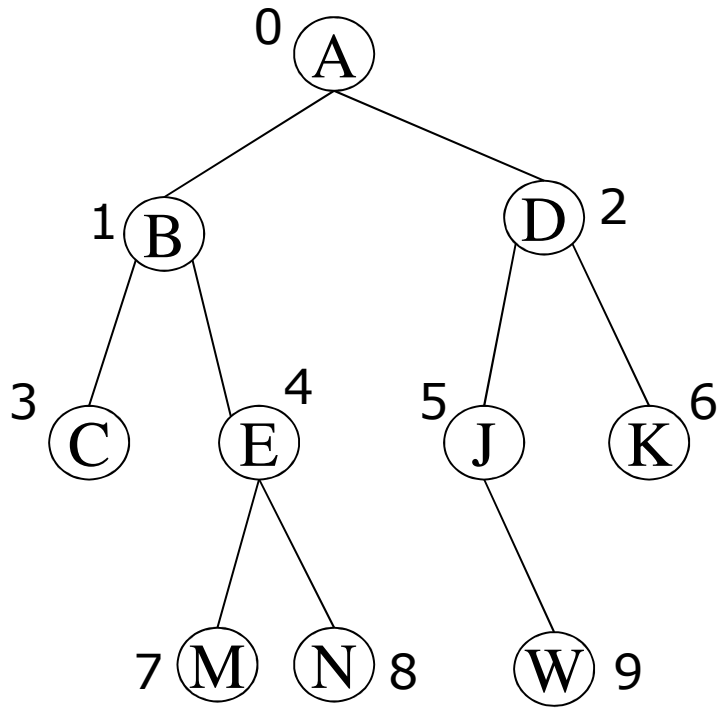
Coada = (C E G K A M D)

Implementarea parcurgerii BFS

```
procedure parcurgeBFS(t, viziteza)
begin
    if (t == NULL)
    then return
    else
         Coada ← coadaVida()
         insereaza(Coada, t)
         while (not esteVida(Coada))
           citeste(Coada, v)
           viziteaza(v)
           if (v->stg != NULL)
           then insereaza(Coada, v->stg)
           if (v->drp != NULL)
           then insereaza(Coada, v->drp)
           elimina(Coada)
end
```

ArbBin: implementarea cu liste

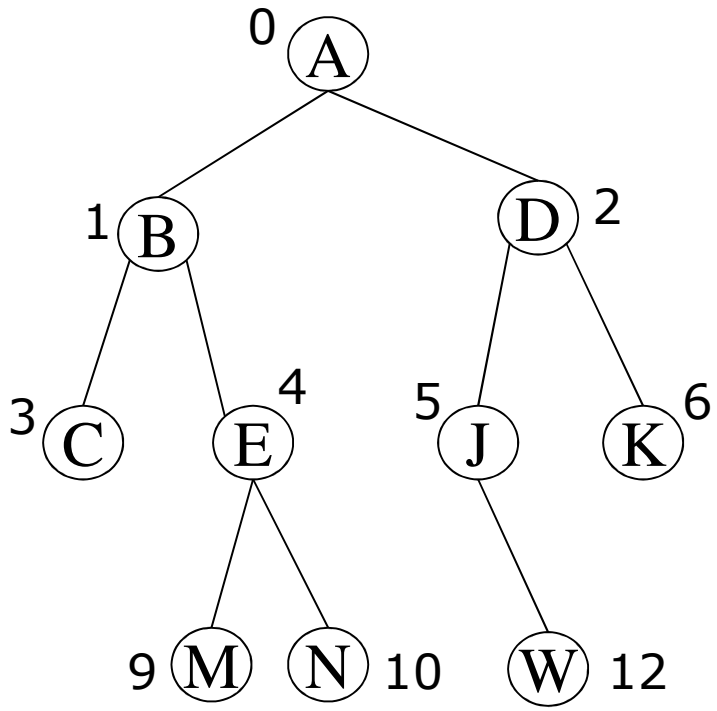
- Reprezentarea relației "părinte": tablou de părinți
- **Avantaje**:
  - Simplitate;
  - Acces ușor de la un nod spre rădăcină;
  - Economie de memorie;
- **Inconveniente**:
  - Acces dificil de la rădăcină spre noduri



-1	0	0	1	1	2	2	4	4	5
0	1	2	3	4	5	6	7	8	9

`ArbBin`:implementare cu tablouri

- Nodurile sunt memorate într-un tablou.
- Indexul unui nod este
  - index(rădăcină) = 0
  - index(x) = 2*index(părinte(x))+1, dacă x este fiu stâng
  - index(x) = 2*index(părinte(x))+2, dacă x este fiu drept



A	B	D	C	E	J	K			M	N		w		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Aplicaţie: expresii întregi

- Expresii întregi
  - definiţie;
  - exemple.

- Reprezentarea expresiilor ca arbori:
  - similarităţi între cele două definiţii;
  - arborele asociat unei expresii;
  - notaţiile prefixate, infixate şi postfixate şi parcurgeri ale arborilor.

Definiţia expresiilor întregi

<int> ::= ... –2 | -1 | 0 | 1 | 2 ...

<op_bin> ::= + | – | * | / | %

<expr_int> ::= <int>

| <exp_int> <op_bin> <exp_int>

| (<exp_int>)

- reguli de precedenţă

  `12-5*2` este **`(12-5)*2`** sau **`12-(5*2)?`**

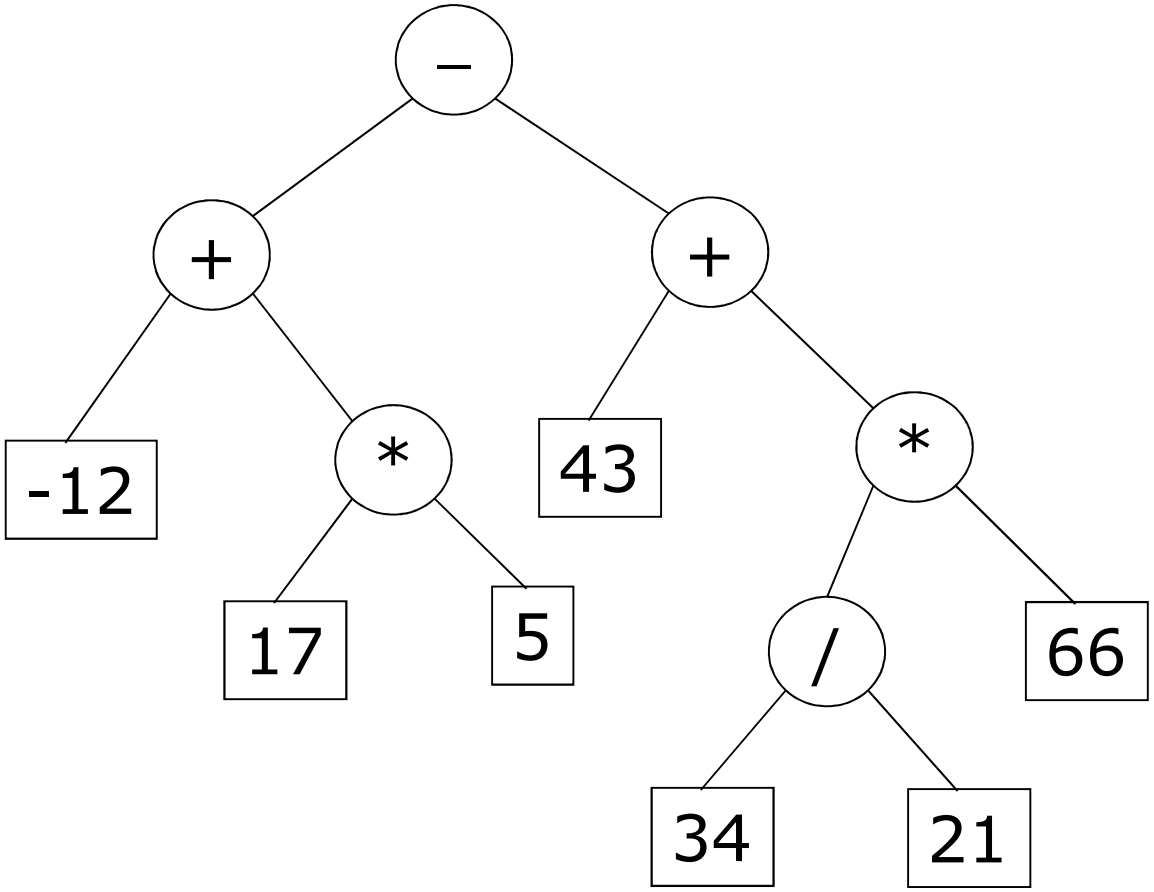
- reguli de asociere

  `15/4/2` este **`(15/4)/2`** sau **`15/(4/2)?`**

  `15/4*2` este **`(15/4)*2`** sau **`15/(4*2)?`**

Expresiile reprezentate ca arbori

−12 + 17 * 5 − (43 + 34 / 21 * 66)



Notaţiile postfixate şi prefixate

- notaţia postfixată se obţine prin parcurge postordine

  –12, 17, 5, *, +, 43, 34, 21, /, 66, *, +, –

- notaţia prefixata se obţine prin parcurge preordine

  –, +, –12, *, 17, 5, +, 43, *, /, 34, 21, 66