

# Arbori binari

SD 2019/2020

Arbori

Arbori binari (ArbBin)

Aplicație: reprezentarea expresiilor ca arbori

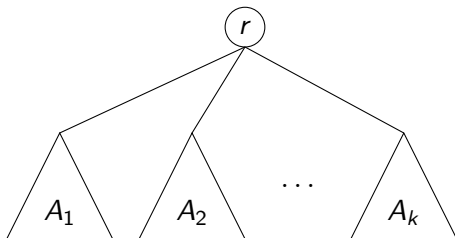
## Arbori: definiție recursivă

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

# Arbori: definiție recursivă

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

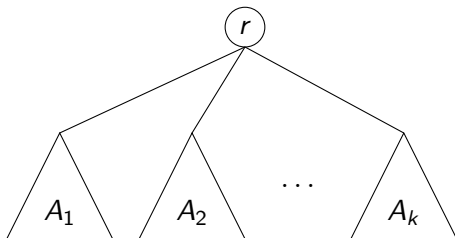
$A = \Lambda$  sau



# Arbori: definiție recursivă

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

$A = \Lambda$  sau

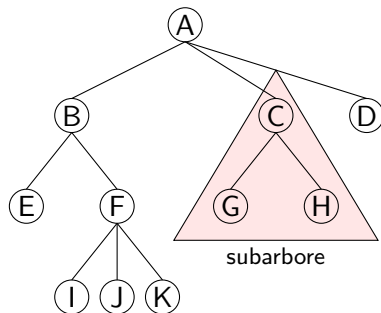


Dacă  $A$  este *ordonat (planar)*, atunci

$$\begin{array}{c} 1 \\ \swarrow \searrow \\ 2 \quad 3 \end{array} \neq \begin{array}{c} 1 \\ \swarrow \searrow \\ 3 \quad 2 \end{array}$$

# Arbori: terminologie

- ▶ **rădacină**: nodul fără părinte.
- ▶ **nod intern**: nod cu cel puțin un fiu.
- ▶ **nod extern (frunză)**: nod fără fii.
- ▶ **descendenții** unui nod: fii, nepoți, etc.
- ▶ **frați**: toate celelalte noduri având același părinte.
- ▶ **subarbor**: arborele format dintr-un nod și descendenții săi.



# Arbori: terminologie

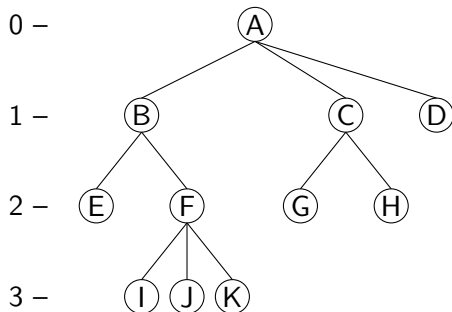
► **adâncimea uni nod  $x$ :**

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

► **înălțimea unui arbore:**

adâncimea maximă a nodurilor arborelui.

► **înălțimea unui nod:** distanța de la nod la cel mai depărtat descendent al său.



Arbori

Arbori binari (ArbBin)

Aplicație: reprezentarea expresiilor ca arbori



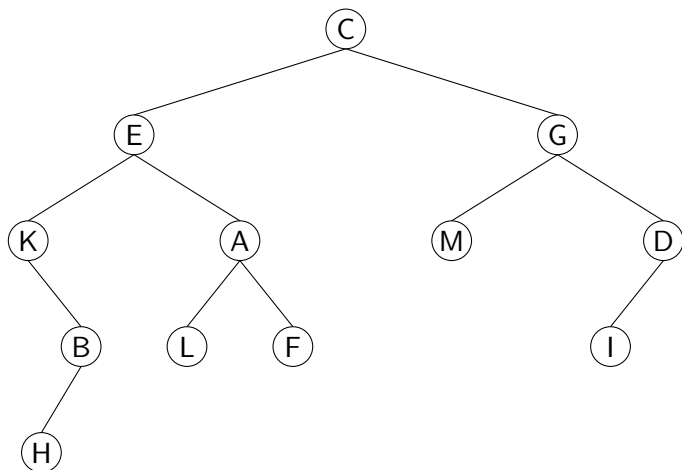
# Tipul abstract ArbBin

OBIECTE: arbori binari.

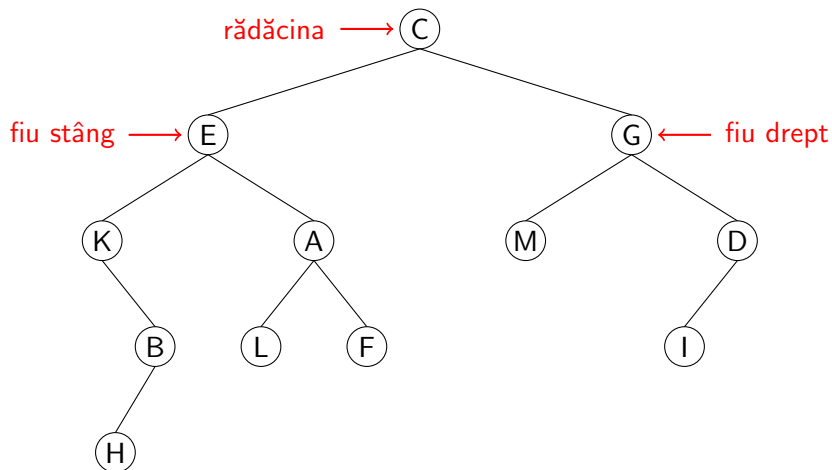
Un **arbore binar** este o colecție de noduri cu proprietățile:

- ▶ orice nod are 0, 1 sau 2 succesori (**fii, copii**).
- ▶ orice nod, exceptând unul singur — **rădăcina** — are un singur nod predecesor (**tată, părinte**).
- ▶ rădăcina nu are predecesori.
- ▶ fii sunt ordonați: fiul stâng, fiul drept. Dacă un nod are un singur fiu, atunci trebuie menționat care.
- ▶ nodurile fără fii formează **frontiera** arborelui.

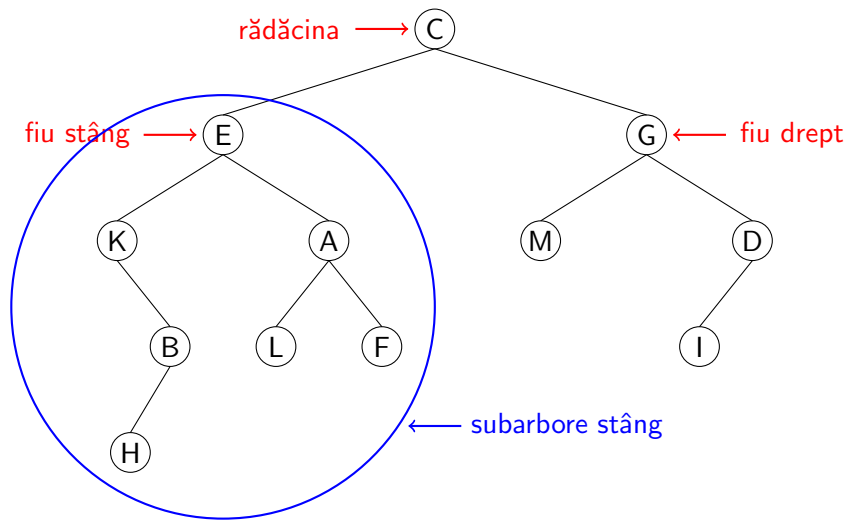
## Arbori binari: exemplu



## Arbori binari: exemplu

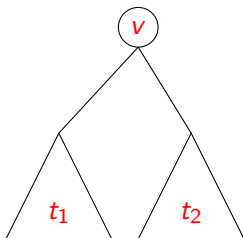


# Arbori binari: exemplu



# Arbori binari: definiție recursivă

- ▶ Arborele vid (fără nici un nod) este arbore binar.
- ▶ Dacă  $v$  este un nod și  $t_1$  și  $t_2$  sunt arbori binari, atunci arborele care are pe  $v$  ca rădăcină,  $t_1$  subarbore stâng al rădăcinii și  $t_2$  subarbore drept al rădăcinii este arbore binar.



# Arbori binari: proprietăți

Notății:

- ▶  $n$  – numărul de noduri din arbore.
- ▶  $n_e$  – numărul de noduri externe.
- ▶  $n_i$  – numărul de noduri interne.
- ▶  $h$  – înălțimea arborelui.

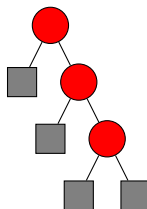
$$h + 1 \leq n \leq 2^{h+1} - 1; \quad \log_2(n + 1) - 1 \leq h \leq n - 1$$

$$1 \leq n_e \leq 2^h; \quad h \leq n_i \leq 2^h - 1$$

# Arbori binari: proprietăți

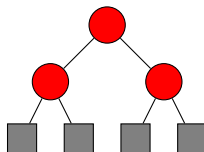
- ▶ **Arbore propriu:** fiecare nod intern are exact doi fii.

$$\begin{aligned}2h + 1 &\leq n \leq 2^{h+1} - 1; \\ \log_2(n + 1) - 1 &\leq h \leq (n - 1)/2 \\ h + 1 &\leq n_e \leq 2^h; \\ h &\leq n_i \leq 2^h - 1 \\ n_e &= n_i + 1\end{aligned}$$



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

$$\begin{aligned}\text{nivelul } i &\text{ are } 2^i \text{ noduri;} \\ n &= 2^{h+1} - 1 = 2n_e - 1\end{aligned}$$



## insereaza()

### ► intrare:

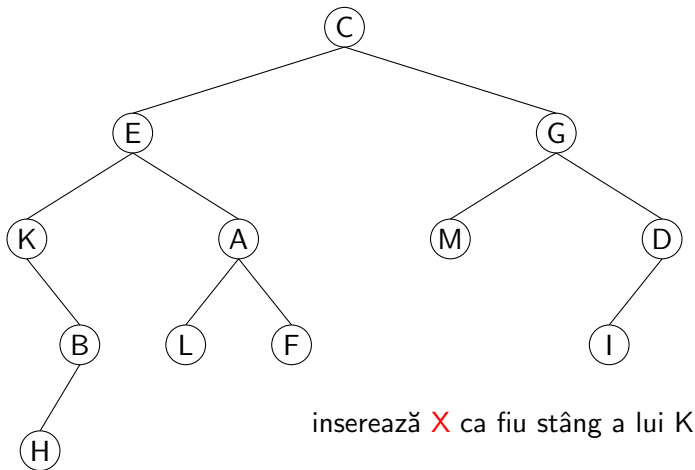
- un arbore binar **t**;
- adresa unui nod cu cel mult un fiu (tatăl noului nod);
- tipul fiului adăgat (stânga, dreapta);
- informația **e** din noul nod.

### ► ieșire:

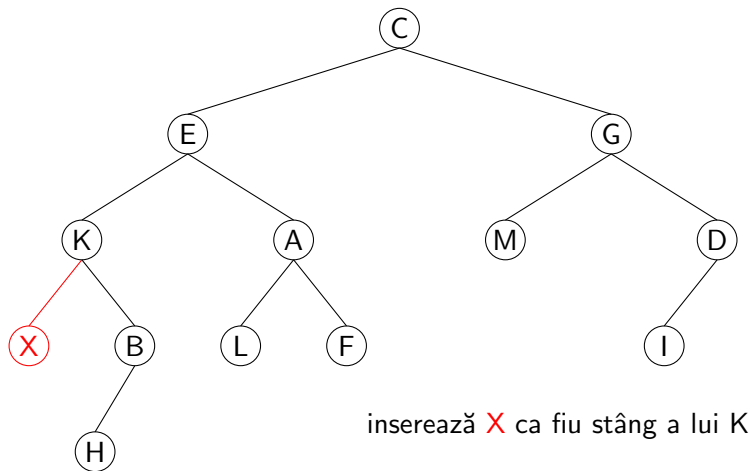
- arborele **t** la care s-a adăgat un nod ce memorează **e**;  
noul nod nu are fii.



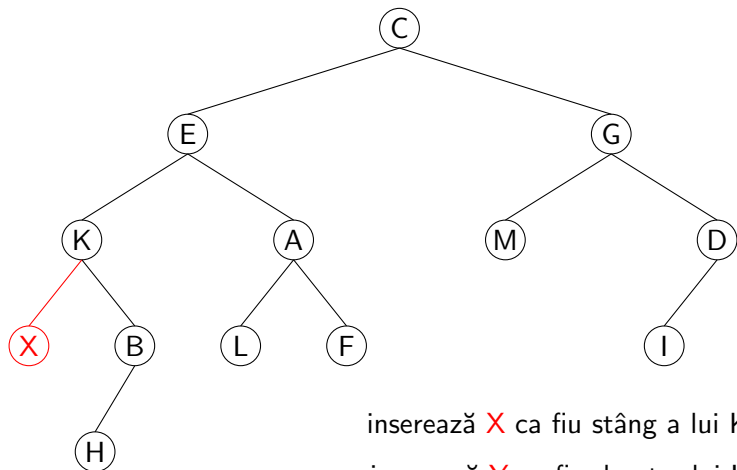
## ArbBin: inserare - exemplu



## ArbBin: inserare - exemplu



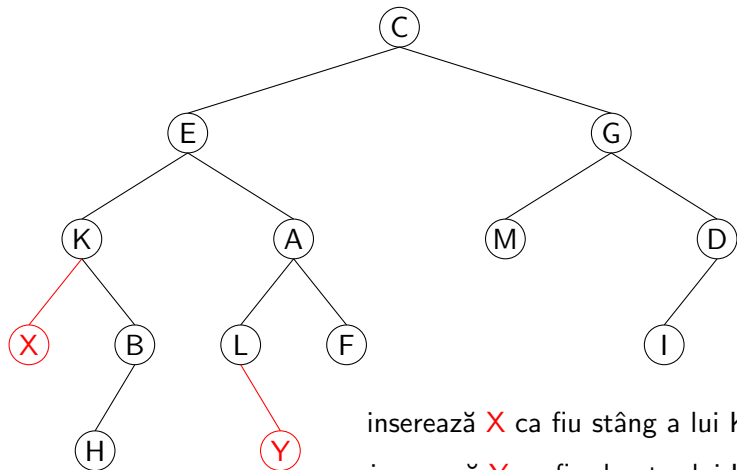
## ArbBin: inserare - exemplu



inserează X ca fiu stâng a lui K

inserează Y ca fiu drept a lui L

## ArbBin: inserare - exemplu



## elimina()

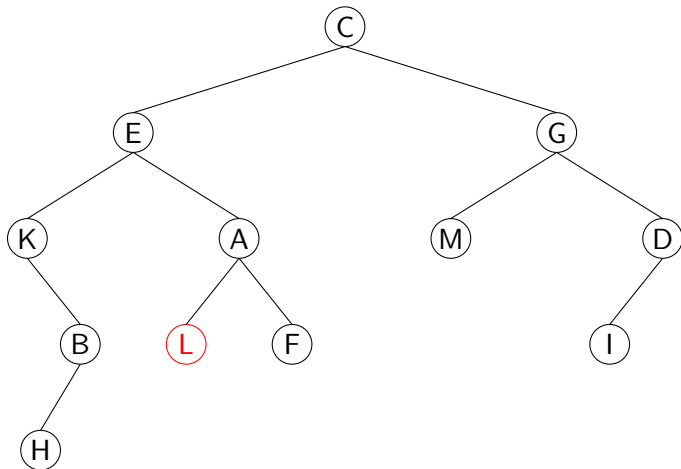
### ► intrare:

- un arbore binar **t**;
- adresa unui nod fără fii și adresa nodului părinte.

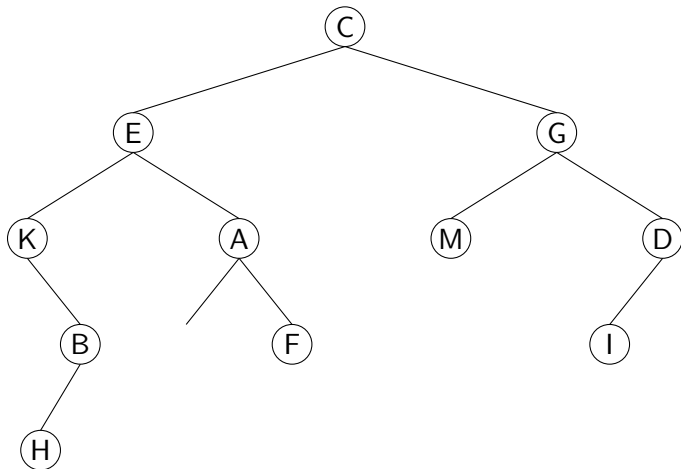
### ► ieșire:

- arborele **t** din care s-a eliminat nodul dat (de pe frontieră).

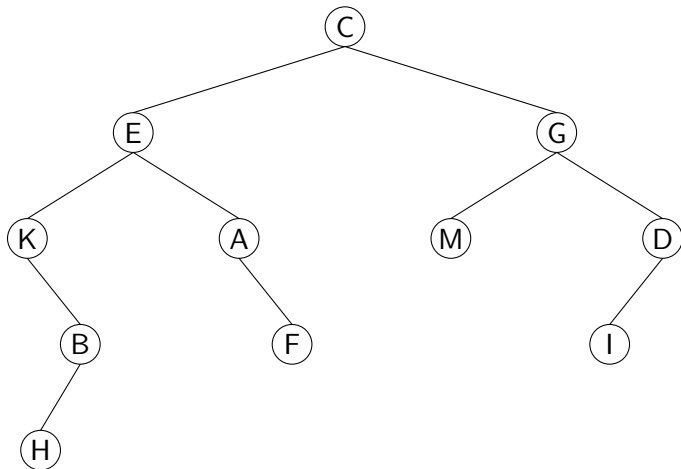
## ArbBin: eliminare - exemplu



## ArbBin: eliminare - exemplu



## ArbBin: eliminare - exemplu





## parcurePreordine()

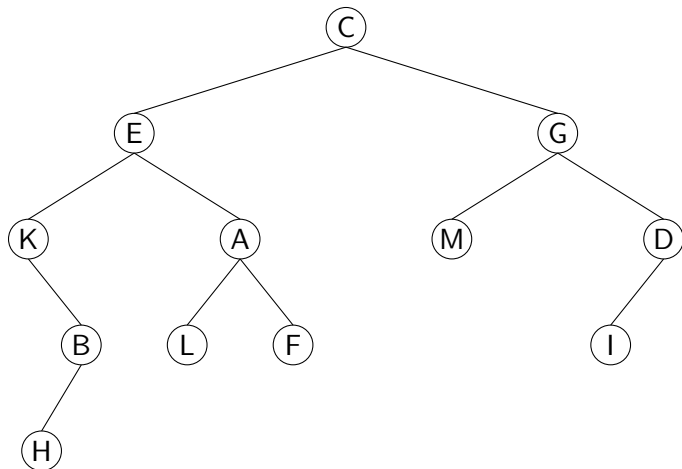
### ► intrare:

- un arbore binar **t**;
- o procedură **viziteaza()**.

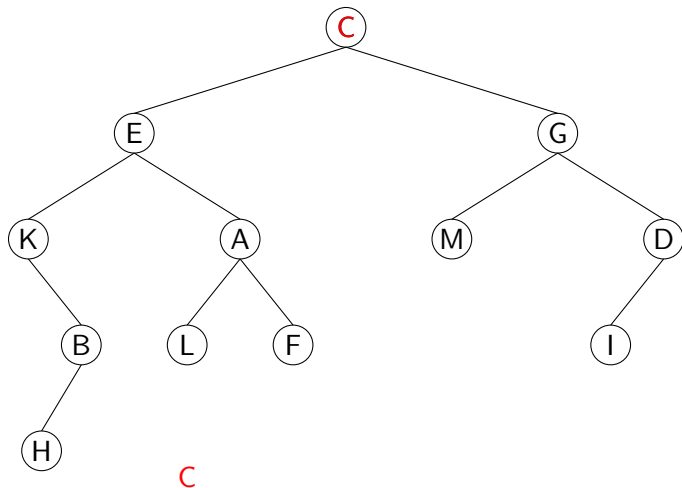
### ► ieșire:

- arborele **t**, dar cu nodurile procesate cu **viziteaza()** în ordinea
  - \* (**R**) – rădăcina
  - \* (**S**) – subarborele stânga
  - \* (**D**) – subarborele dreapta

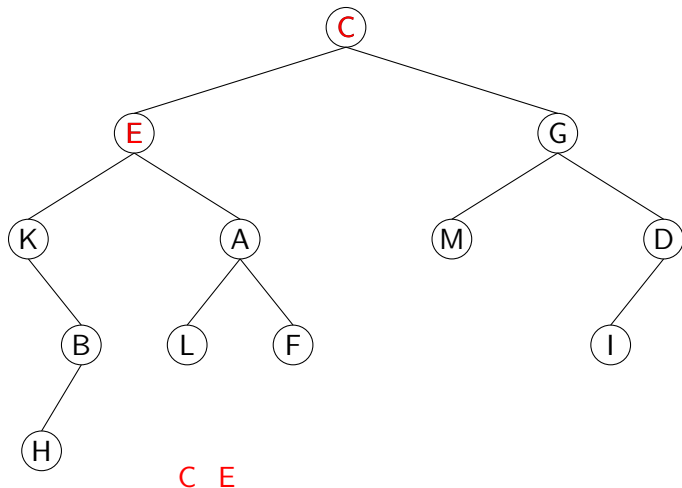
## Parcuregere preordine - exemplu



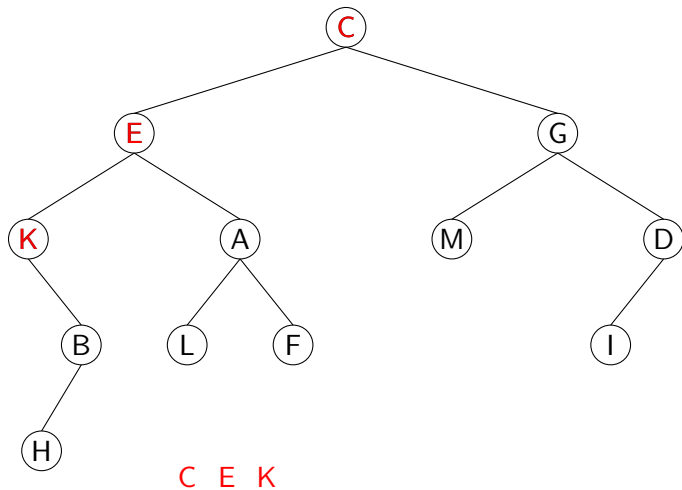
## Parcuregere preordine - exemplu



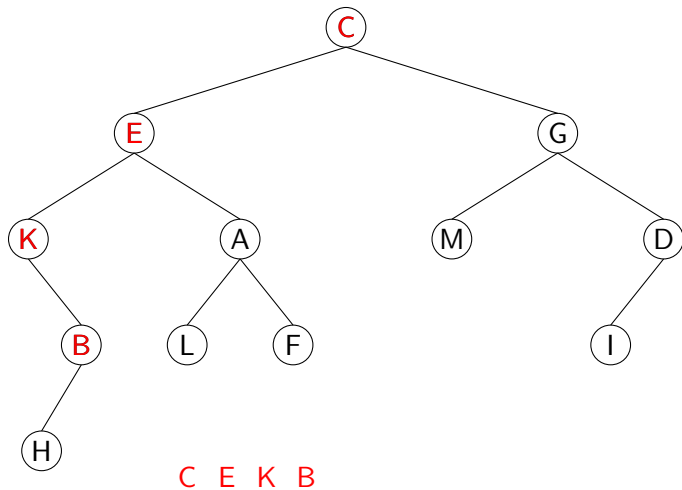
## Parcuregere preordine - exemplu



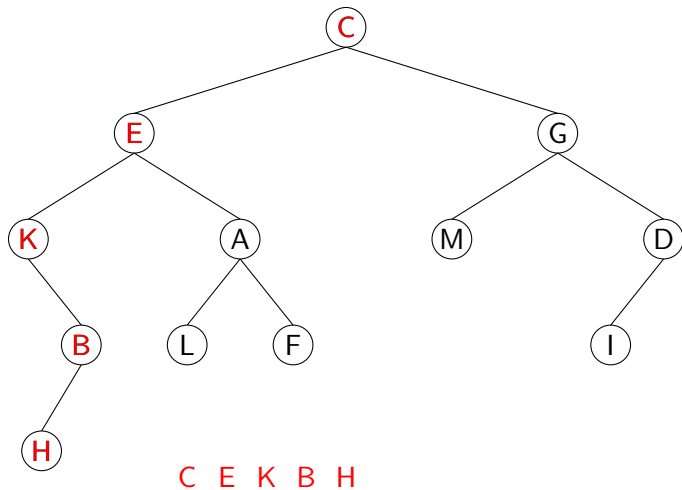
## Parcuregere preordine - exemplu



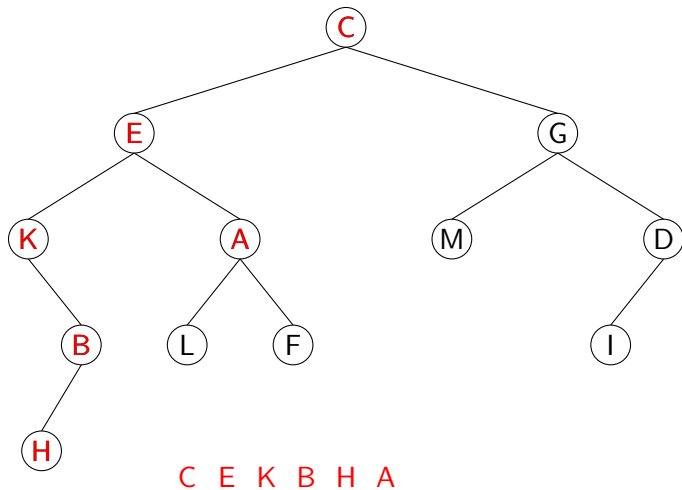
## Parcuregere preordine - exemplu



## Parcuregere preordine - exemplu

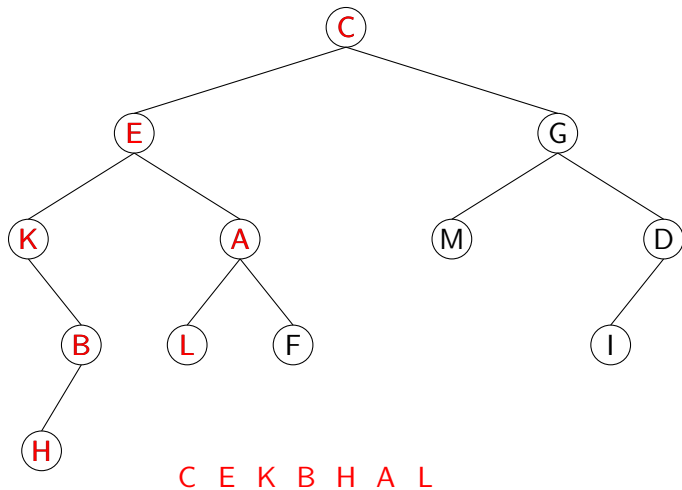


## Parcuregere preordine - exemplu

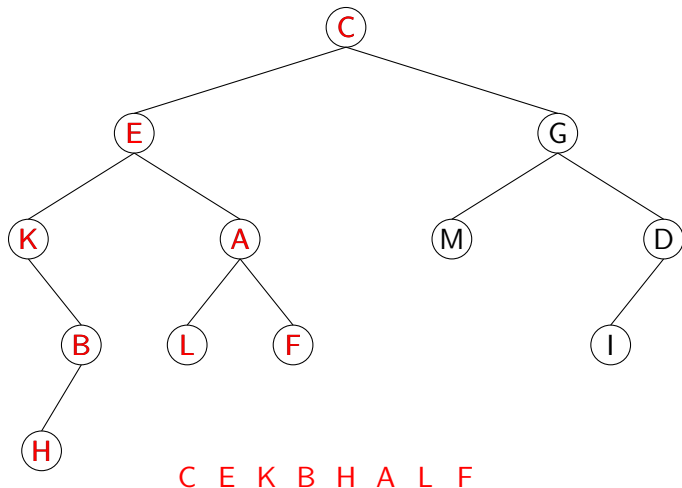




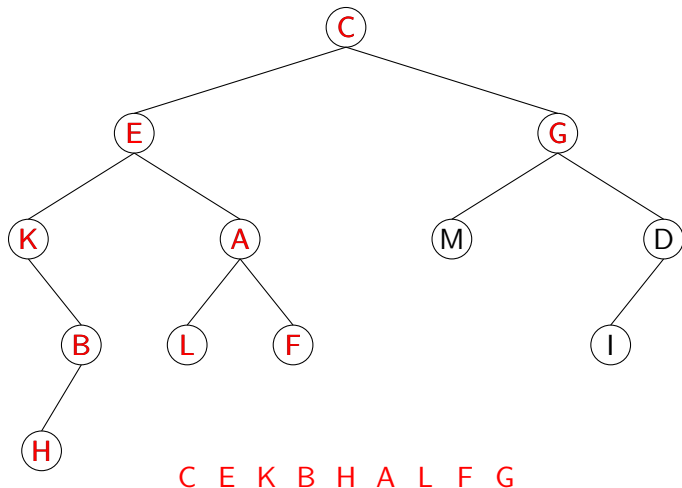
## Parcuregere preordine - exemplu



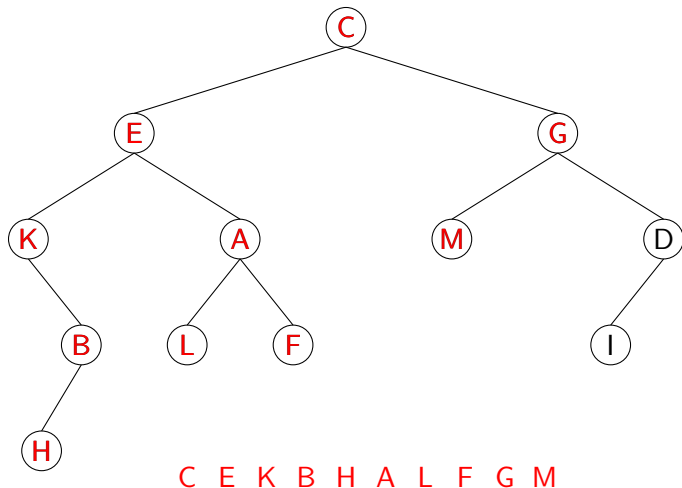
## Parcuregere preordine - exemplu



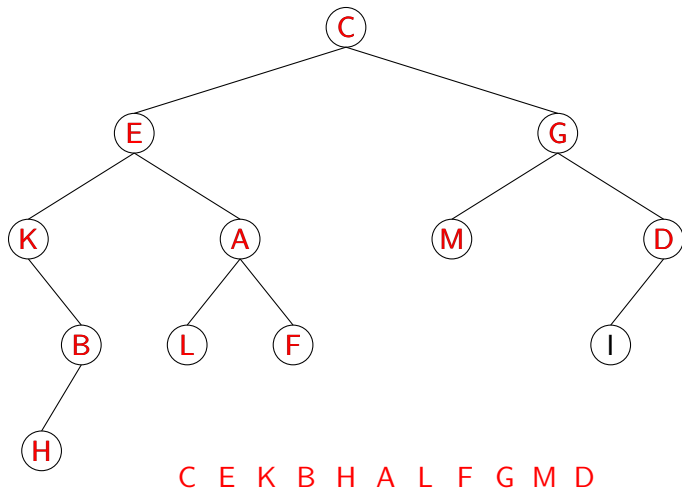
## Parcurgere preordine - exemplu



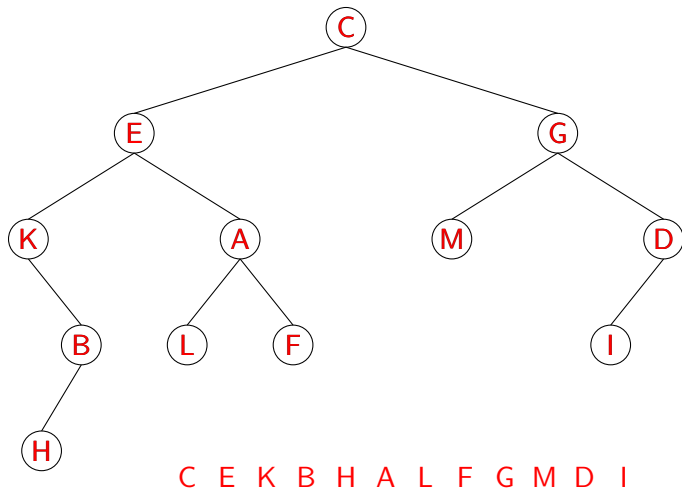
## Parcuregere preordine - exemplu



## Parcuregere preordine - exemplu



## Parcurgere preordine - exemplu



## parcureInordine()

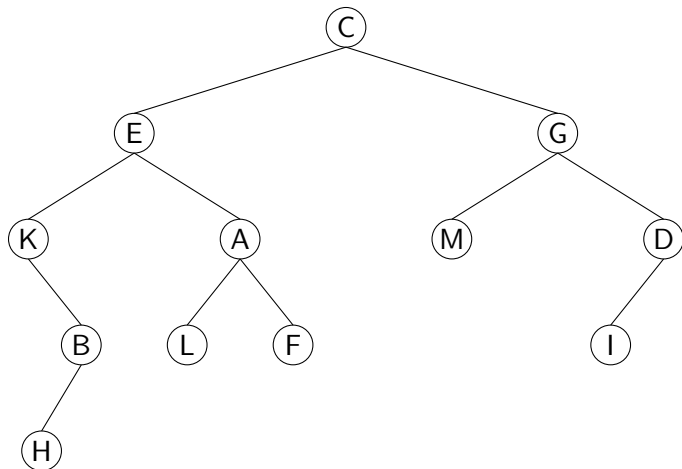
### ► intrare:

- un arbore binar `t`;
- o procedură `viziteaza()`.

### ► ieșire:

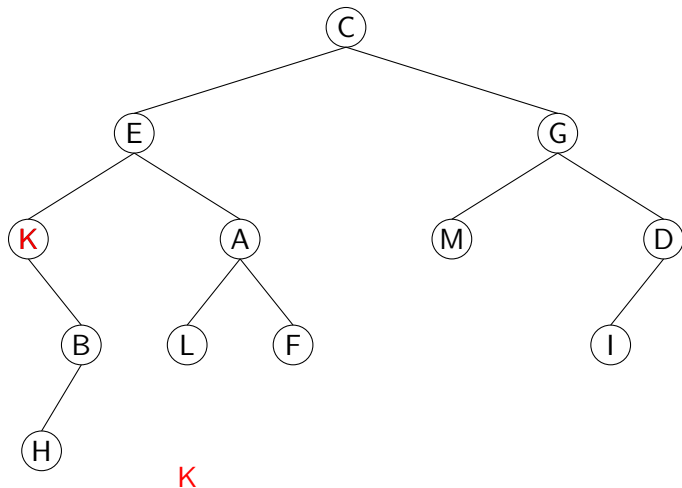
- arborele `t`, dar cu nodurile procesate cu `viziteaza()` în ordinea
  - \* (`S`) – subarborele stânga
  - \* (`R`) – rădăcina
  - \* (`D`) – subarborele dreapta

## Parcuregere inordine - exemplu

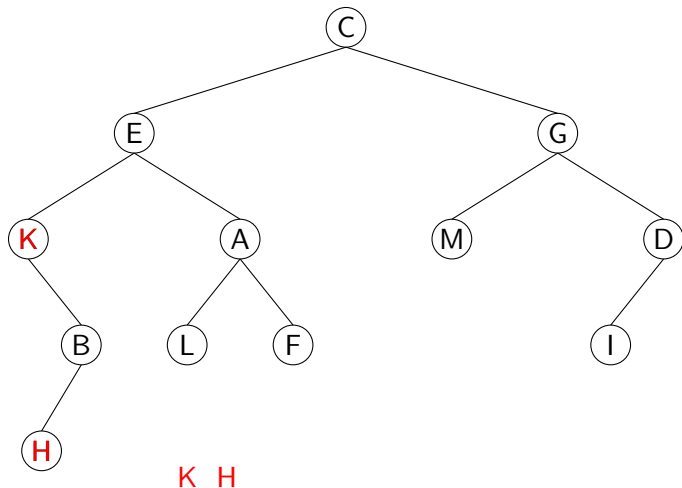




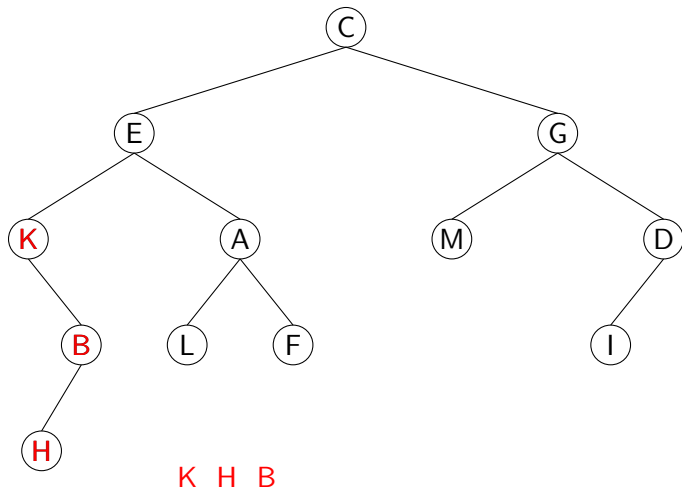
## Parcursere inordine - exemplu



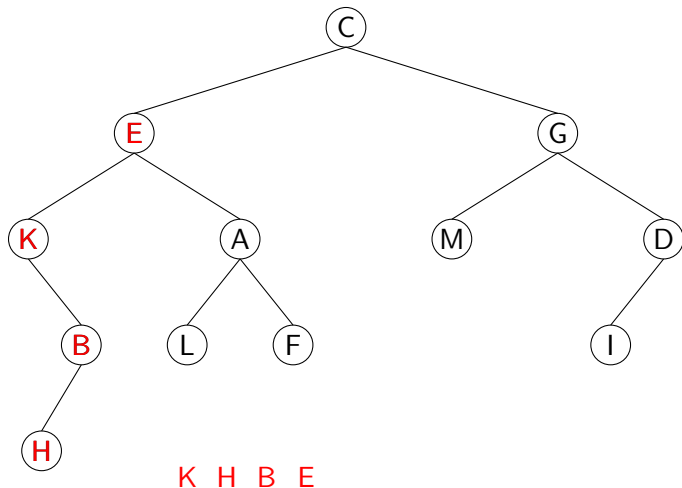
## Parcuregere inordine - exemplu



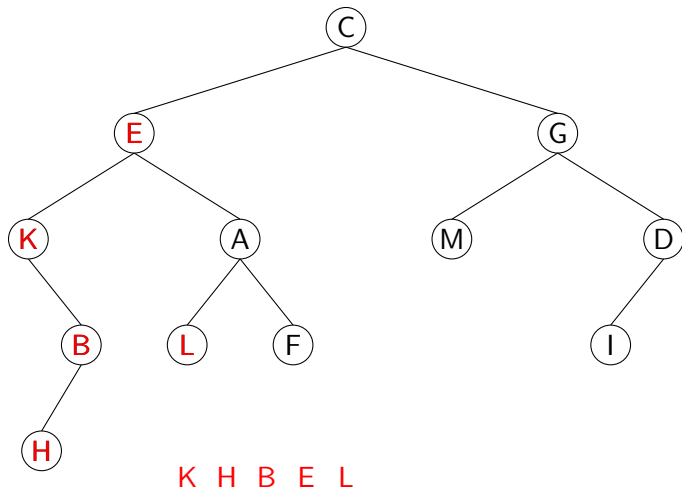
## Parcuregere inordine - exemplu



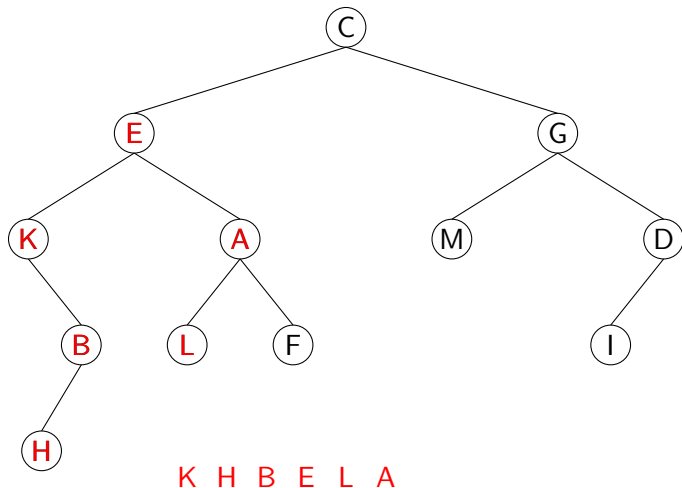
## Parcuregere inordine - exemplu



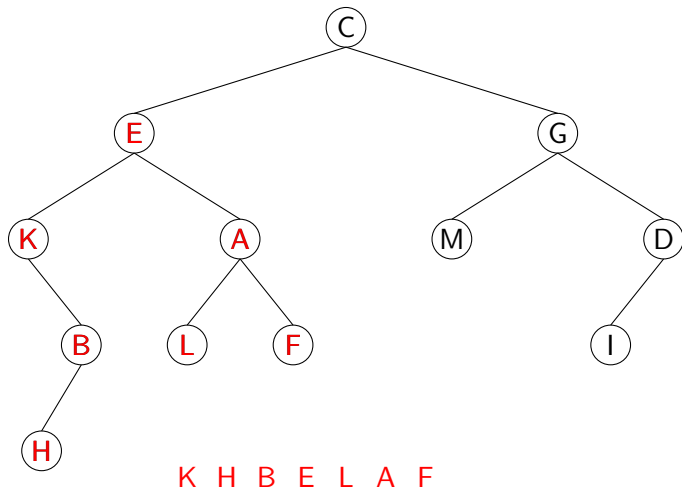
## Parcuregere inordine - exemplu



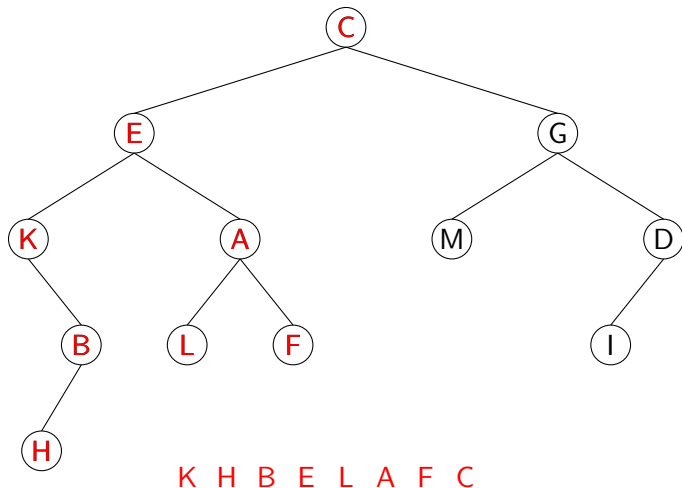
## Parcuregere inordine - exemplu



## Parcuregere inordine - exemplu

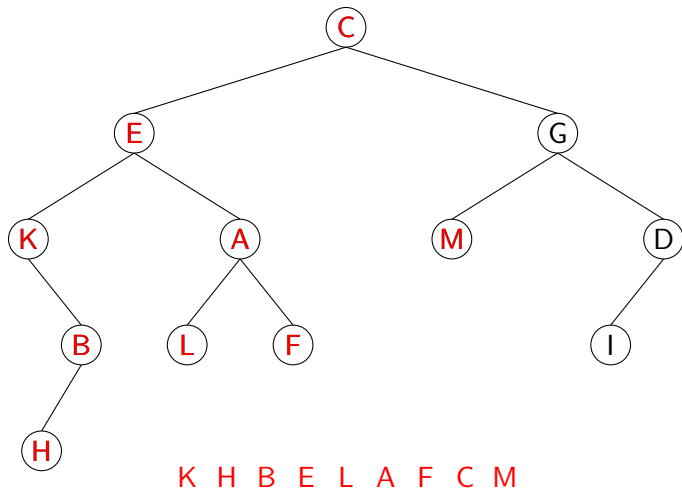


## Parcuregere inordine - exemplu

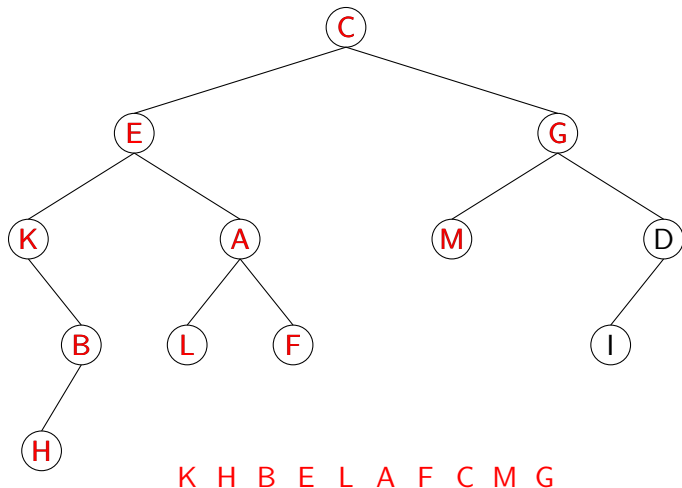




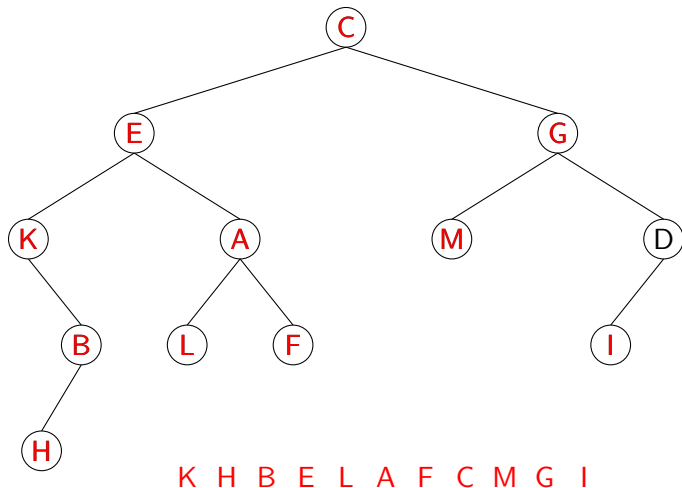
## Parcuregere inordine - exemplu



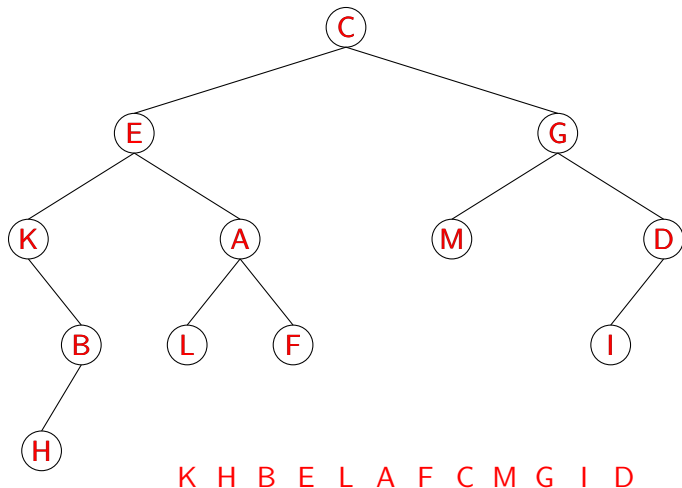
## Parcuregere inordine - exemplu



## Parcuregere inordine - exemplu



# Parcuregere inordine - exemplu



## parcurePostordine()

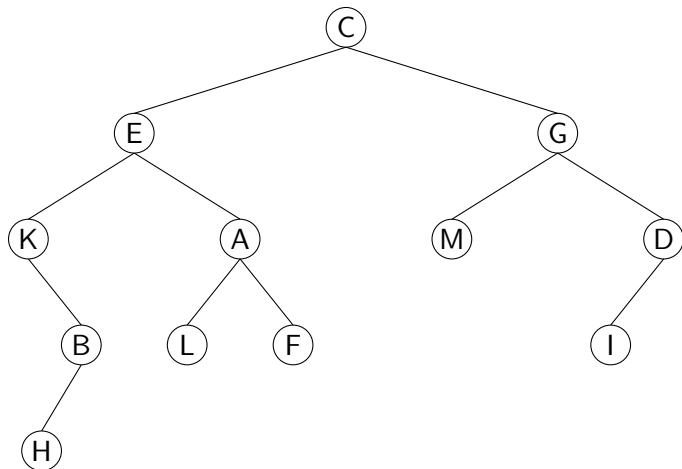
### ► intrare:

- un arbore binar **t**;
- o procedură **viziteaza()**.

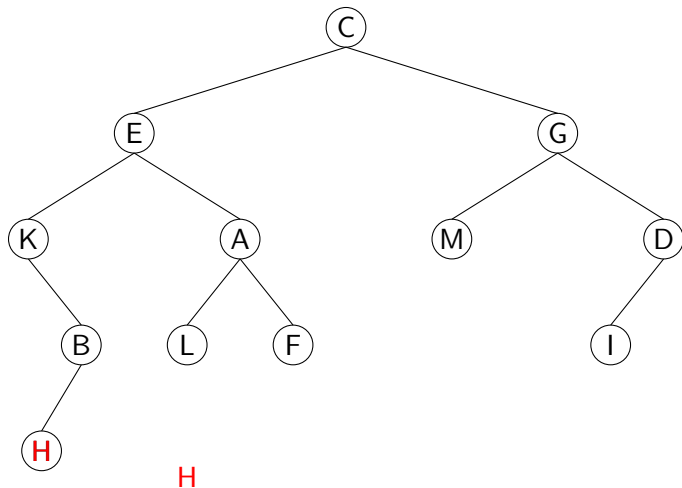
### ► ieșire:

- arborele **t**, dar cu nodurile procesate cu **viziteaza()** în ordinea
  - \* (**S**) – subarborele stânga
  - \* (**D**) – subarborele dreapta
  - \* (**R**) – rădăcina

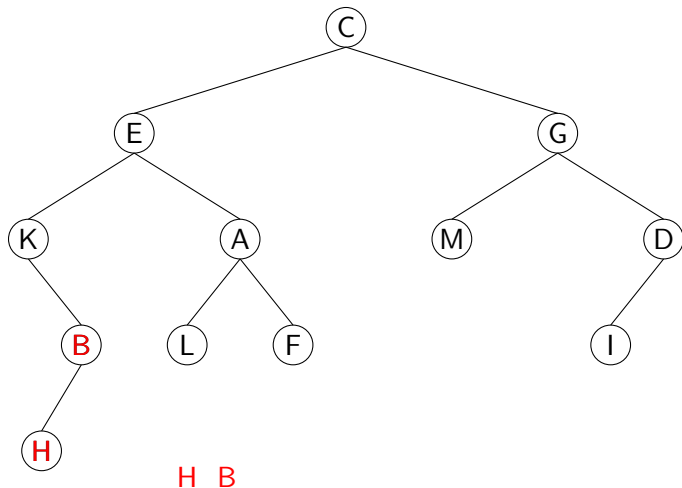
## Parcuregere postordine - exemplu



## Parcursere postordine - exemplu

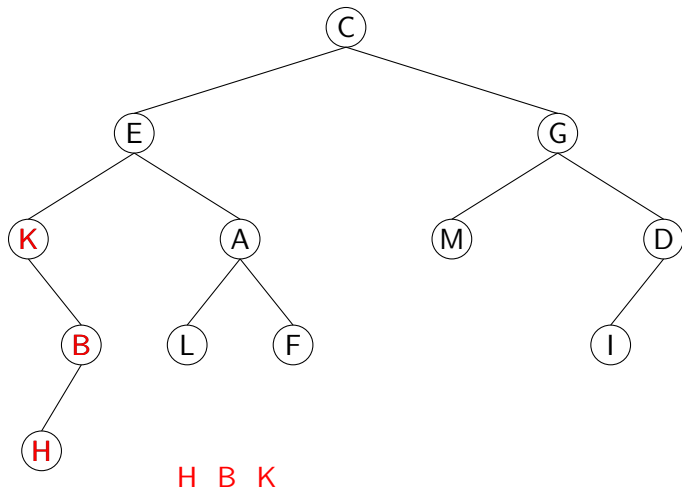


## Parcursere postordine - exemplu

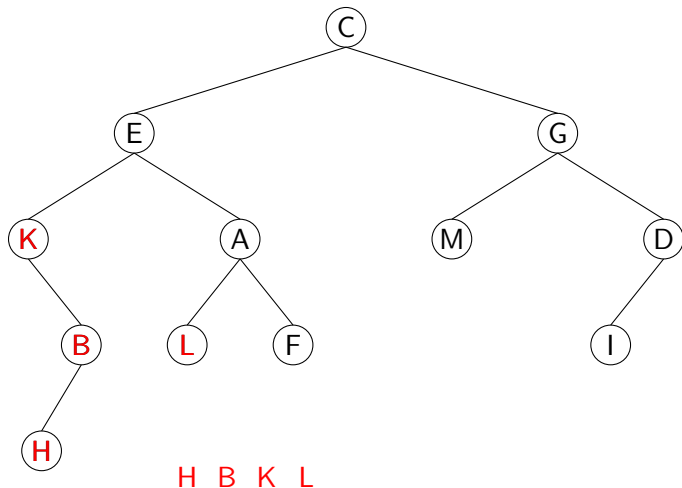




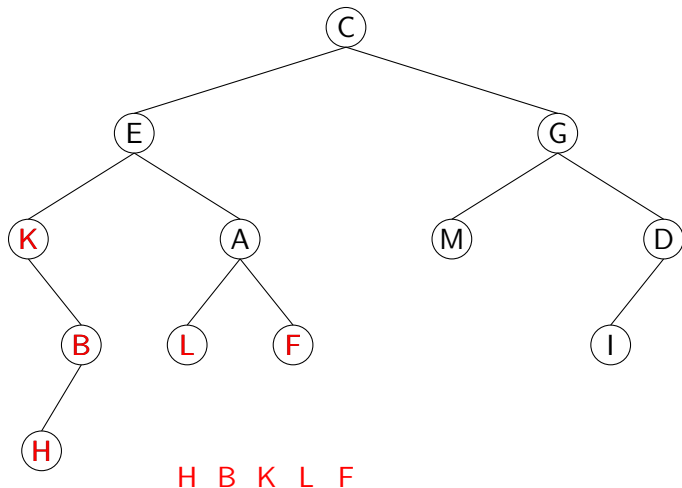
## Parcuregere postordine - exemplu



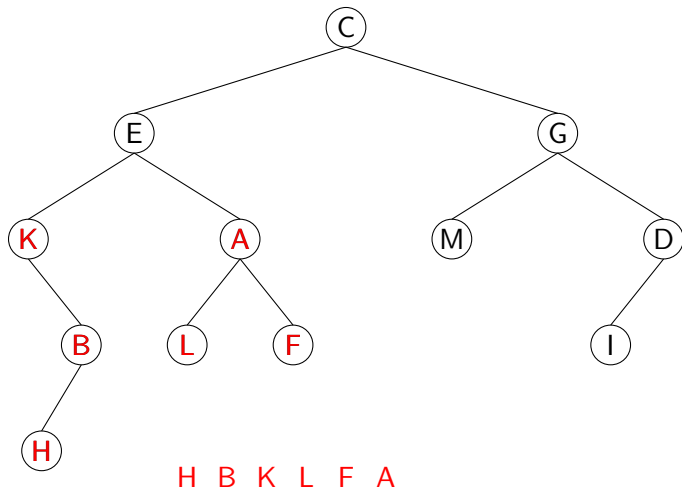
## Parcuregere postordine - exemplu



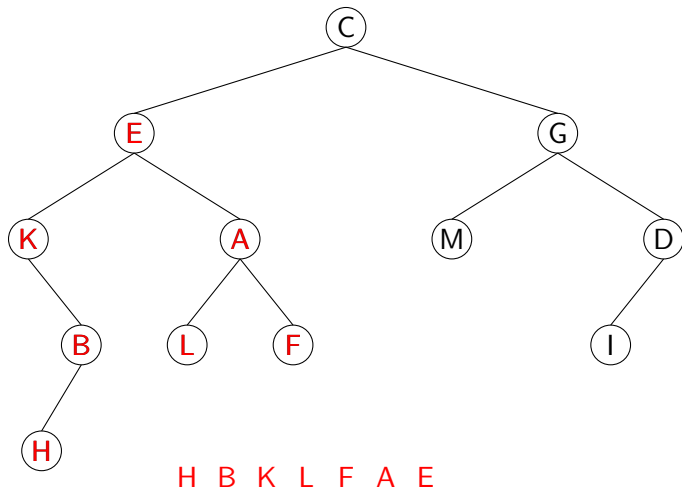
## Parcuregere postordine - exemplu



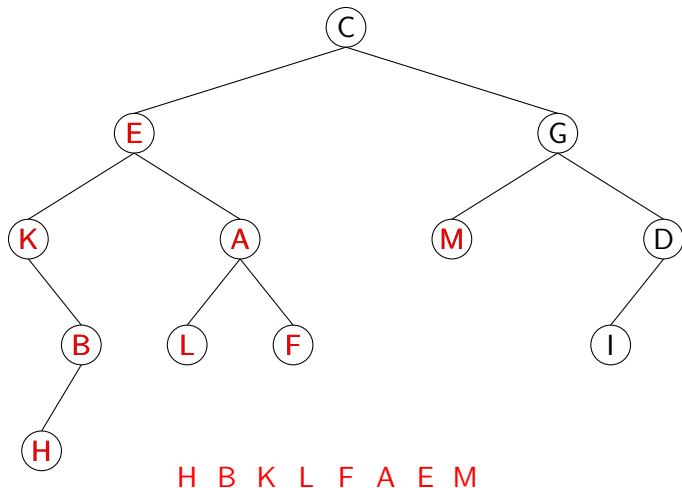
## Parcure postordine - exemplu



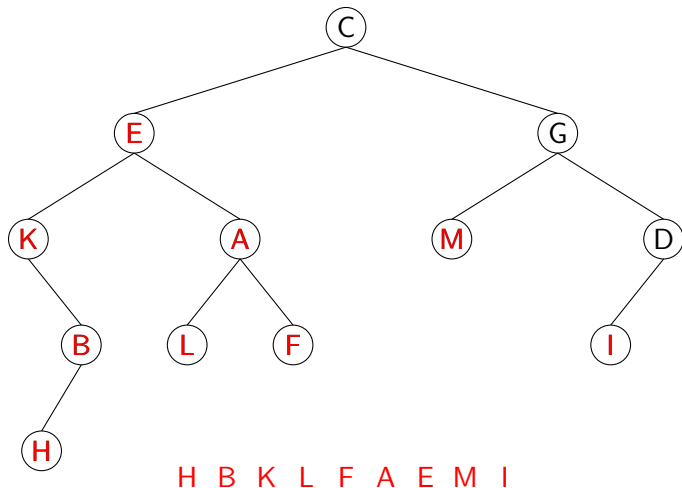
## Parcure postordine - exemplu



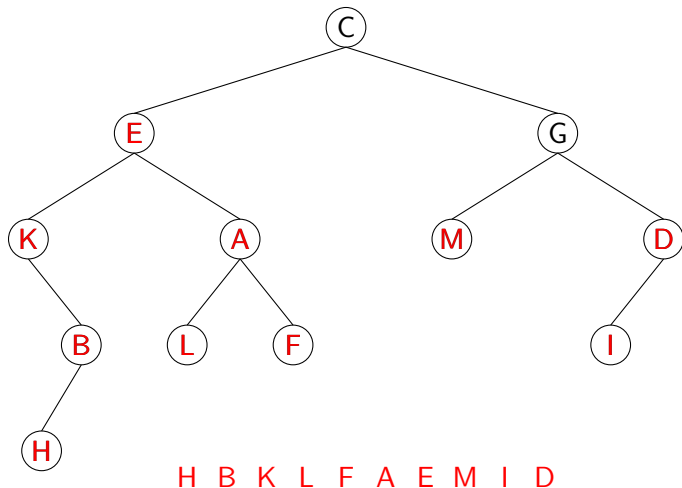
## Parcure postordine - exemplu



## Parcuregere postordine - exemplu

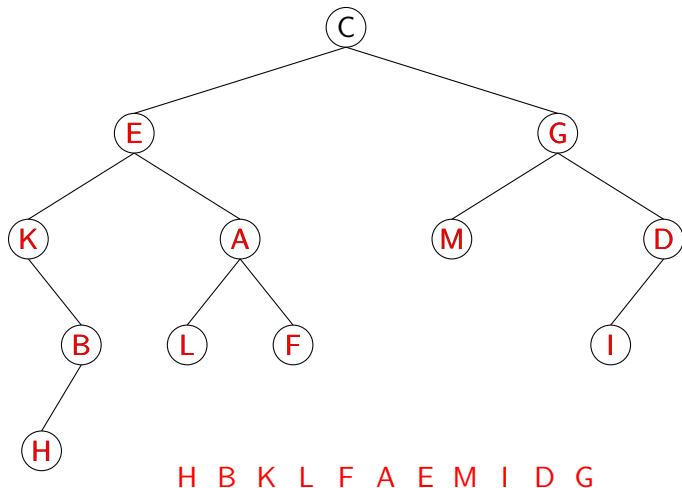


## Parcuregere postordine - exemplu

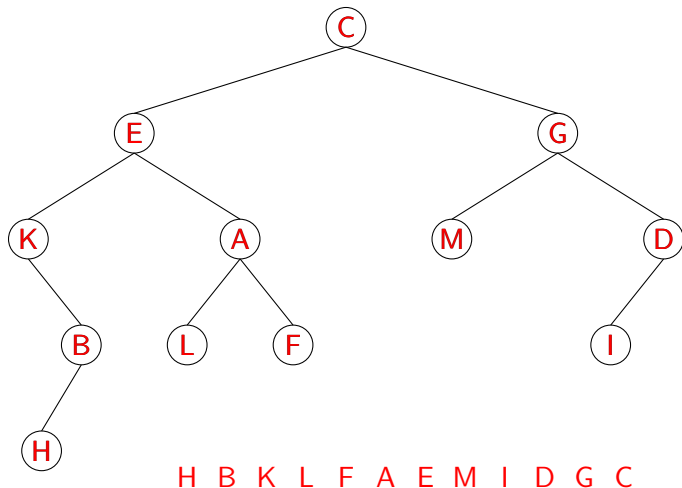




## Parcuregere postordine - exemplu



## Parcursgere postordine - exemplu



# ArbBin – parcurgerea BFS (pe lăţime)

## parcurgereBFS()

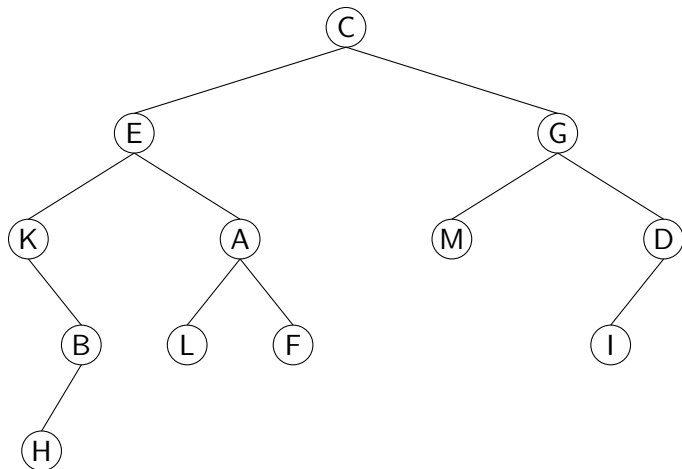
### ► intrare:

- un arbore binar `t`;
- o procedură `viziteaza()`.

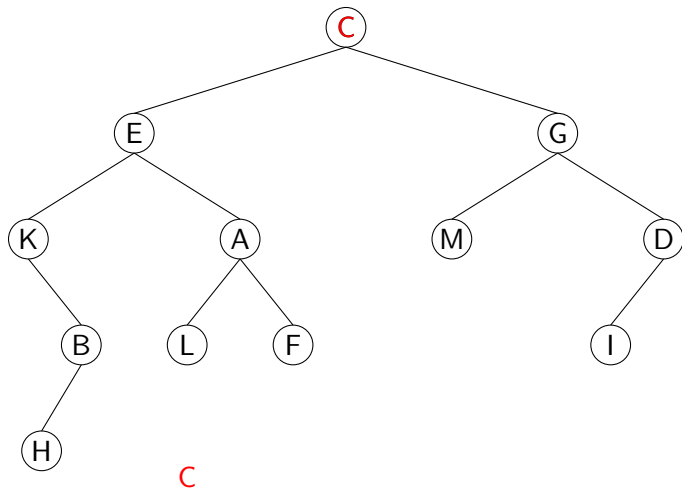
### ► ieşire:

- arborele `t`, dar cu nodurile procesate cu `viziteaza()` în ordinea BFS (pe lăţime / pe niveluri).

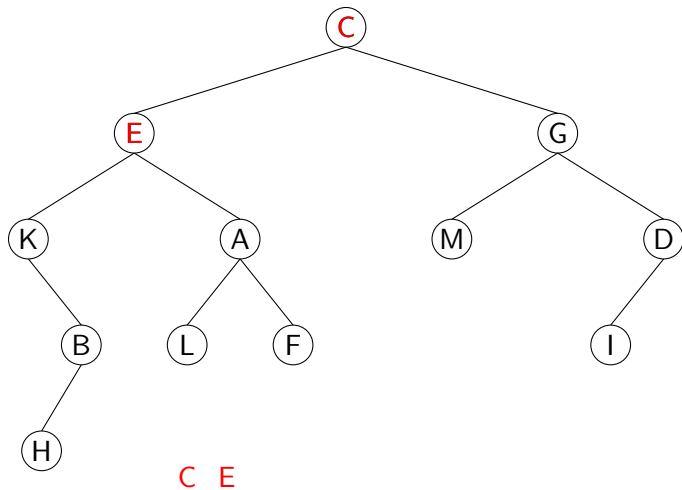
## Parcure BFS - exemplu



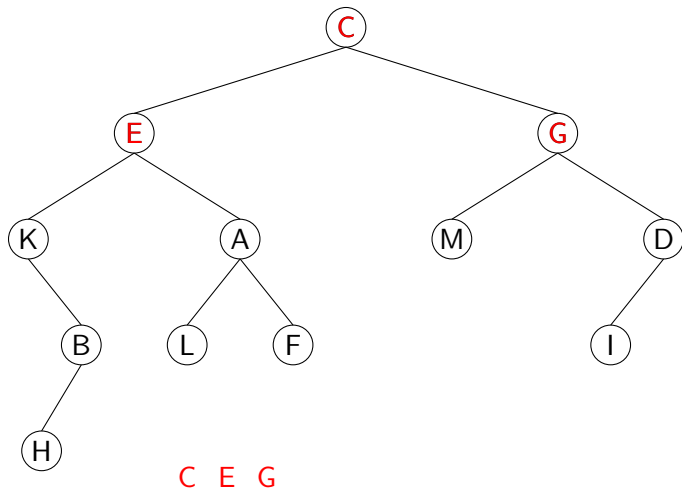
## Parcure BFS - exemplu



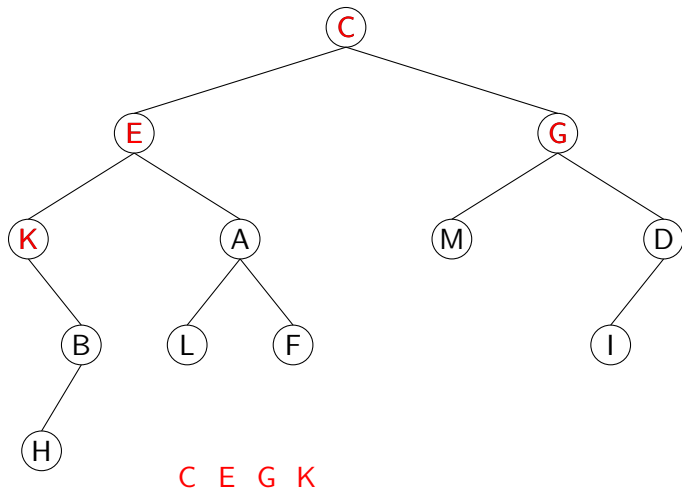
## Parcure BFS - exemplu



## Parcure BFS - exemplu

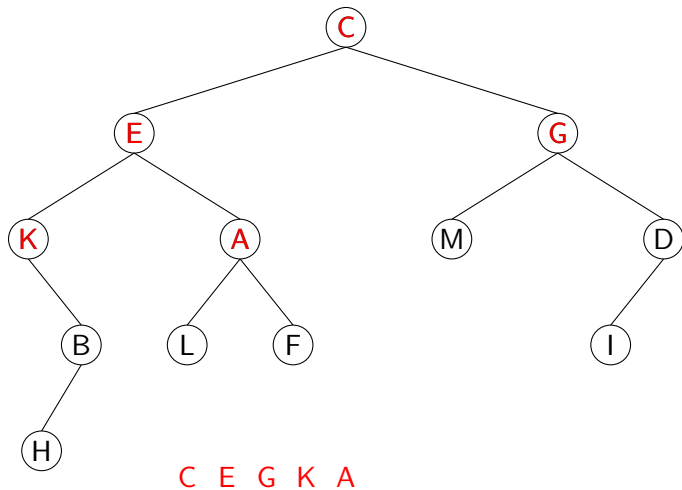


## Parcure BFS - exemplu

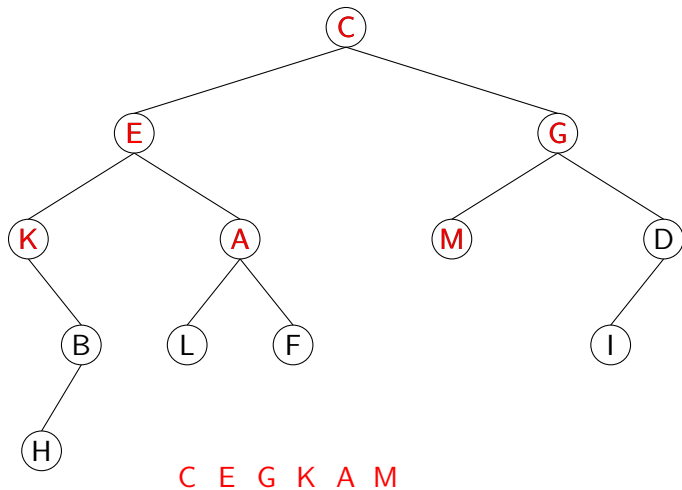




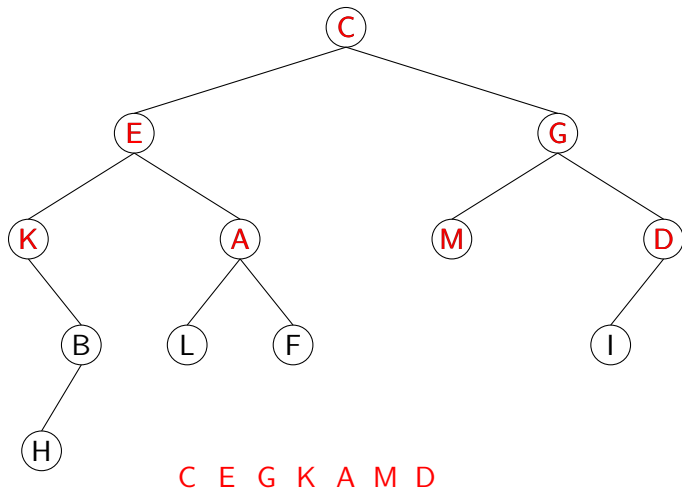
## Parcure BFS - exemplu



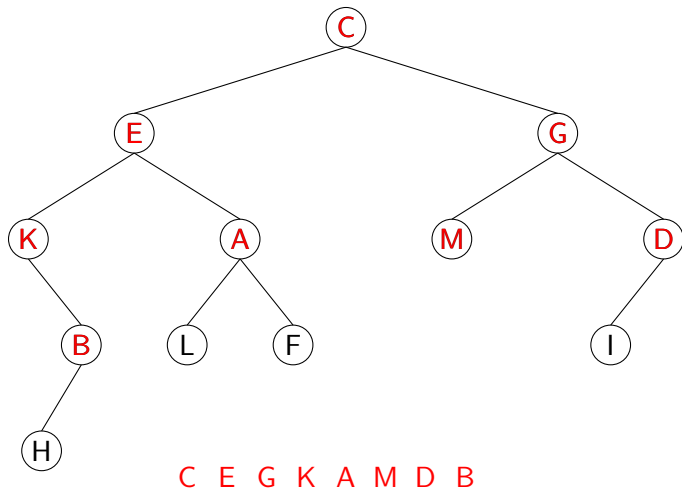
## Parcure BFS - exemplu



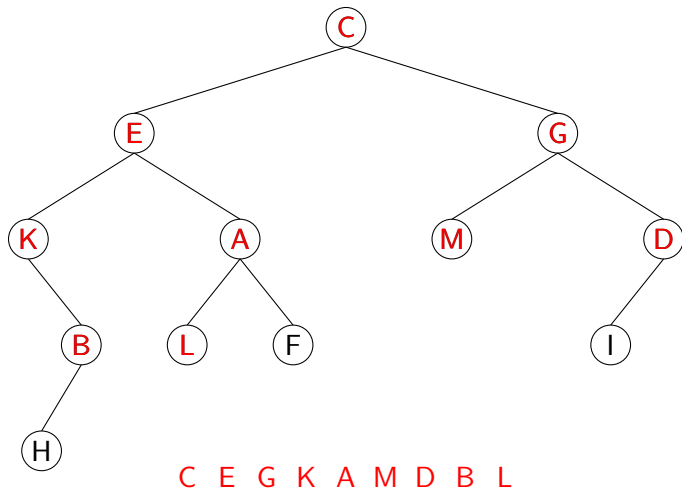
## Parcure BFS - exemplu



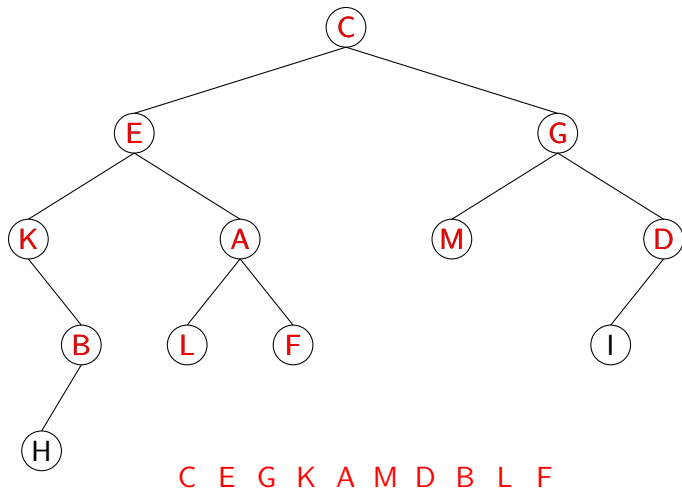
## Parcure BFS - exemplu



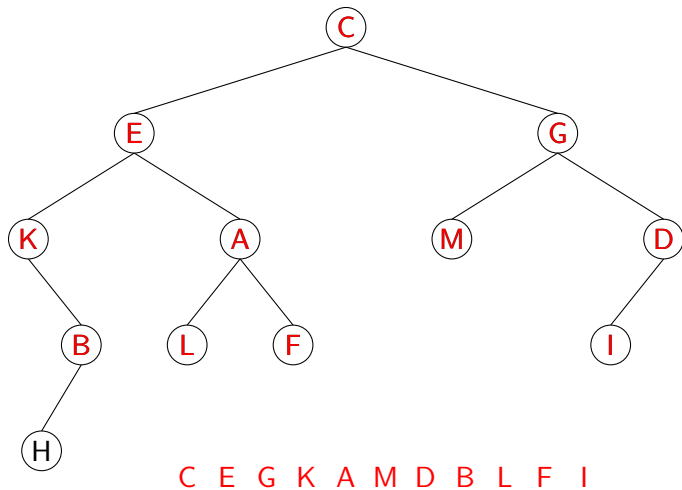
## Parcourire BFS - exemple



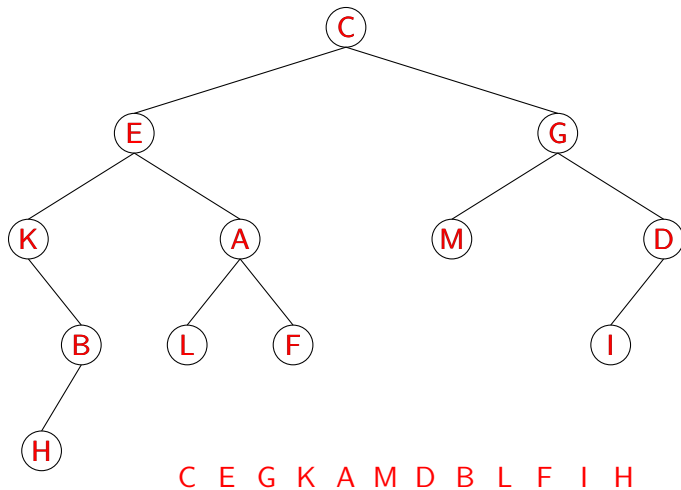
## Parcourire BFS - exemple



## Parcuregere BFS - exemplu

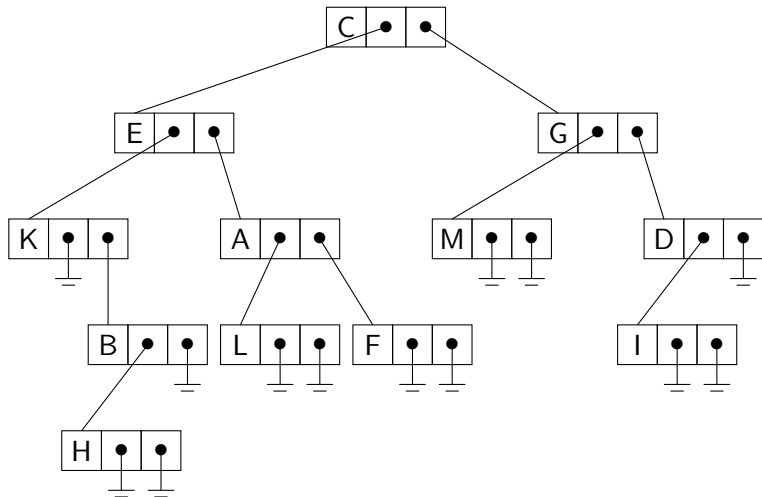


# Parcuregere BFS - exemplu





# ArbBin: implementarea cu structuri înlănțuite



## ArbBin: structura unui nod

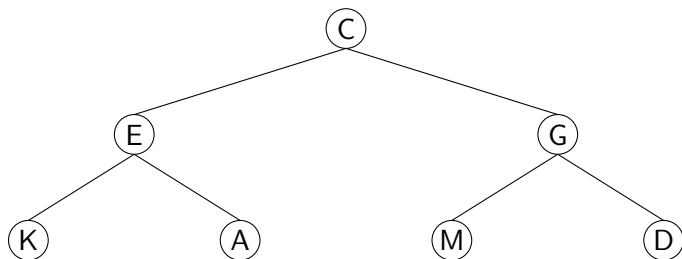
Un nod  $v$  (aflat la adresa de memorie  $v$ ) este o structură cu trei câmpuri:

- ▶  $v \rightarrow inf$  – informația memorată în nod;
- ▶  $v \rightarrow stg$  – adresa fiului stânga;
- ▶  $v \rightarrow drp$  – adresa fiului dreapta.

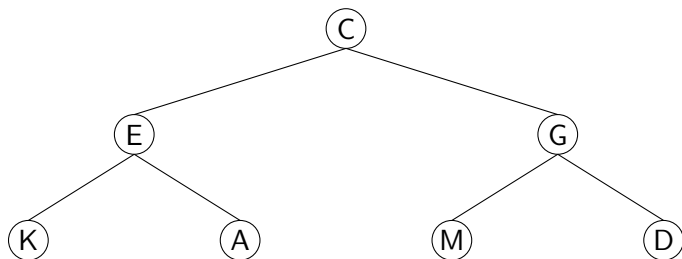
## ArbBin: `parcurePreordine()`

```
procedure parcurePreordine(v, viziteaza)  
begin  
    if (v == NULL) then  
        return  
    else  
        viziteaza(v)  
        parcurePreordine(v → stg, viziteaza)  
        parcurePreordine(v → drp, viziteaza)  
    end
```

# Implementarea parcurgerii BFS



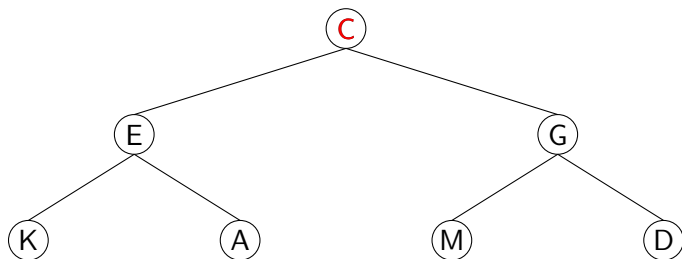
# Implementarea parcurgerii BFS



BFS =

Coadă = ( )

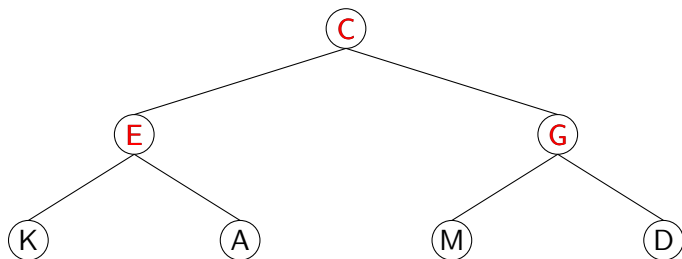
# Implementarea parcurgerii BFS



BFS =

Coadă = ( C )

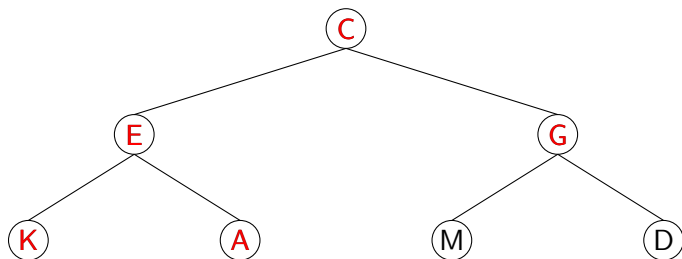
# Implementarea parcurgerii BFS



BFS = C

Coadă = ( C E G )

# Implementarea parcurgerii BFS

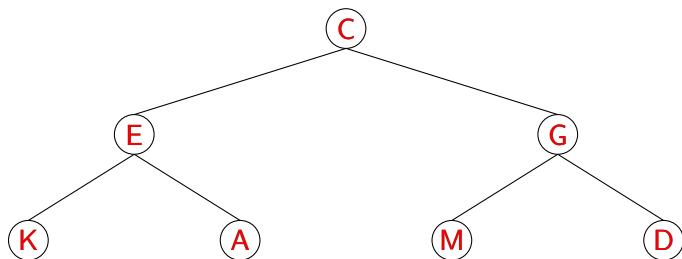


BFS = C E

Coadă = ( C E G K A )



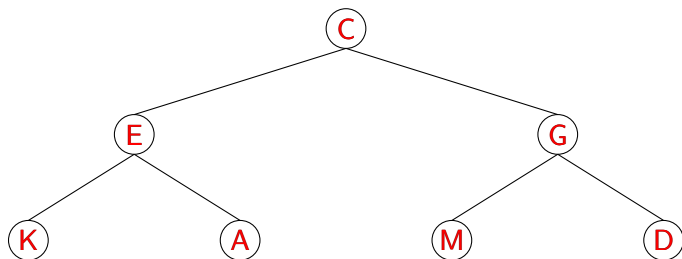
# Implementarea parcurgerii BFS



BFS = C E G

Coadă = ( C E G K A M D )

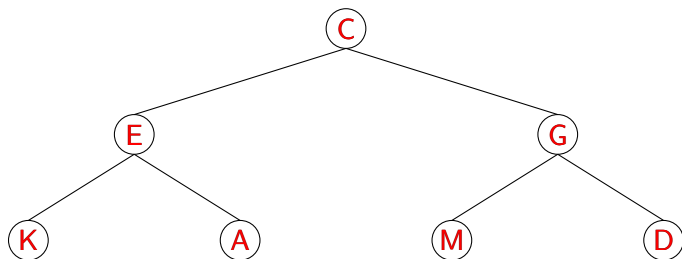
# Implementarea parcurgerii BFS



BFS = C E G K

Coadă = ( C E G K A M D )

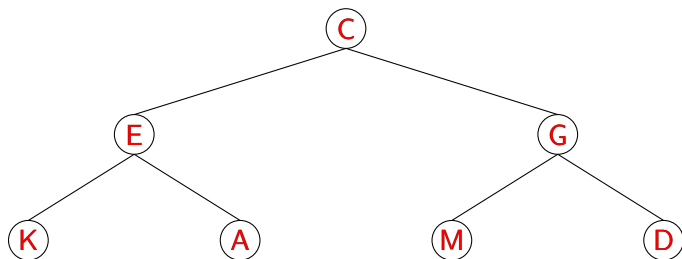
# Implementarea parcurgerii BFS



BFS = C E G K A

Coadă = ( C E G K A M D )

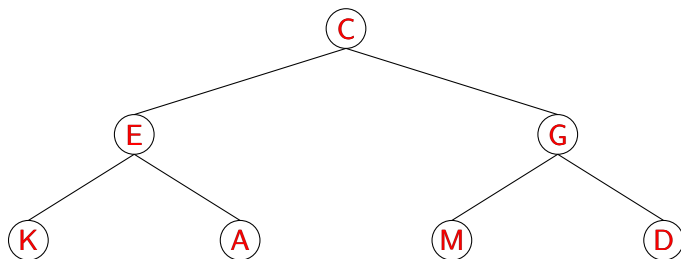
# Implementarea parcurgerii BFS



BFS = C E G K A M

Coadă = ( C E G K A M D )

# Implementarea parcurgerii BFS



BFS = C E G K A M D

Coadă = ( C E G K A M D )

# Implementarea parcurgerii BFS

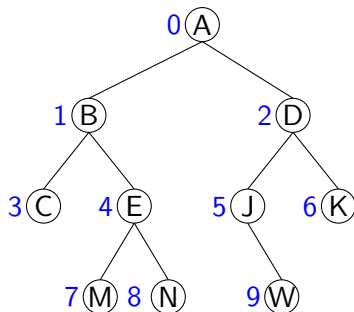
```
procedure parcurgeBFS(t, viziteaza)  
begin  
    if (t == NULL) then  
        return  
    else  
        Coadă ← coadaVida()  
        insereaza(Coadă, t)  
        while not esteVida(Coadă) do  
            citeste(Coadă, v)  
            viziteaza(v)  
            if (v → stg ≠ NULL) then  
                insereaza(Coadă, v → stg)  
            if (v → drp ≠ NULL) then  
                insereaza(Coadă, v → drp)  
            elimina(Coadă)  
end
```

# ArbBin: implementarea cu liste

- ▶ **tablou de părinți:** reprezentarea relației “părinte”.

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

- ▶ **Avantaje:**
  - simplitate;
  - acces ușor de la un nod spre rădăcină;
  - economie de memorie.
- ▶ **Inconveniente:**
  - acces dificil de la rădăcină spre noduri.

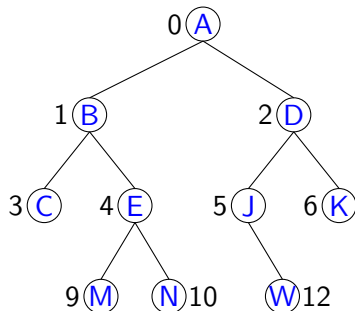


# ArbBin: implementarea cu tablouri

► Nodurile sunt memorate într-un tablou.

► Indexul unui nod este:

- $\text{index}(\text{rădăcină}) = 0$
- $\text{index}(x) = 2 * \text{index}(\text{părinte}(x)) + 1$ ,  
dacă  $x$  este fiu stâng
- $\text{index}(x) = 2 * \text{index}(\text{părinte}(x)) + 2$ ,  
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



Arbori

Arbori binari (ArbBin)

Aplicație: reprezentarea expresiilor ca arbori

## ► Expresii întregi

- definiție;
- exemple.

## ► Reprezentarea expresiilor ca arbori

- similarități între cele două definiții;
- arborele asociat unei expresii;
- notațiile prefixate, infixate și postfixate și parcurgeri ale arborilor.

# Definiția expresiilor întregi

$\langle \text{int} \rangle ::= \dots -2 \mid -1 \mid 0 \mid 1 \mid 2 \dots$

$\langle \text{op\_bin} \rangle ::= + \mid - \mid * \mid / \mid \%$

$\langle \text{exr\_int} \rangle ::= \langle \text{int} \rangle$

$\mid (\langle \text{exp\_int} \rangle)$

$\mid \langle \text{exp\_int} \rangle \langle \text{op\_bin} \rangle \langle \text{exp\_int} \rangle$

► reguli de precedență

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

► reguli de asociere

$15/4/2$  este  $(15/4)/2$  sau  $15/(4/2)$ ?

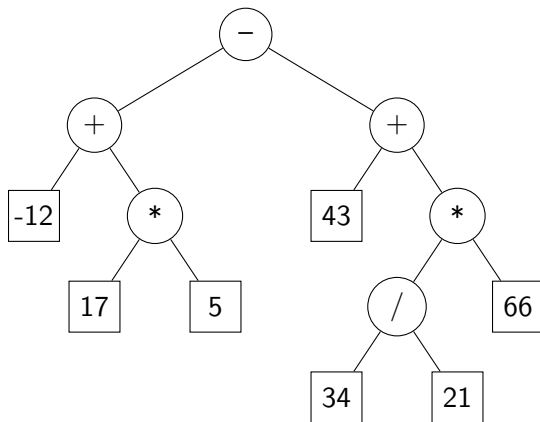
$15/4 * 2$  este  $(15/4) * 2$  sau  $15/(4 * 2)$ ?

## Expresiile reprezentate ca arbori

$$-12 + 17 * 5 - (43 + 34 / 21 * 66)$$

# Expresiile reprezentate ca arbori

$$-12 + 17 * 5 - (43 + 34 / 21 * 66)$$



# Notațiile postfixate și prefixate

- ▶ Notația postfixată se obține prin parcurgerea postordine  
-12, 17, 5, \*, +, 43, 34, 21, /, 66, \*, +, -
- ▶ Notația prefixată se obține prin parcurgerea preordine  
-, +, -12, \*, 17, 5, +, 43, \*, /, 34, 21, 66

