ARBORI OARECARE ȘI ARBORI BINARI

CURS 8- 20.04.2021

Titular: Șef. Lucr. Dr. Mat. Cărbureanu Mădălina

Universitatea Petrol-Gaze din Ploiești

OBIECTIVE:

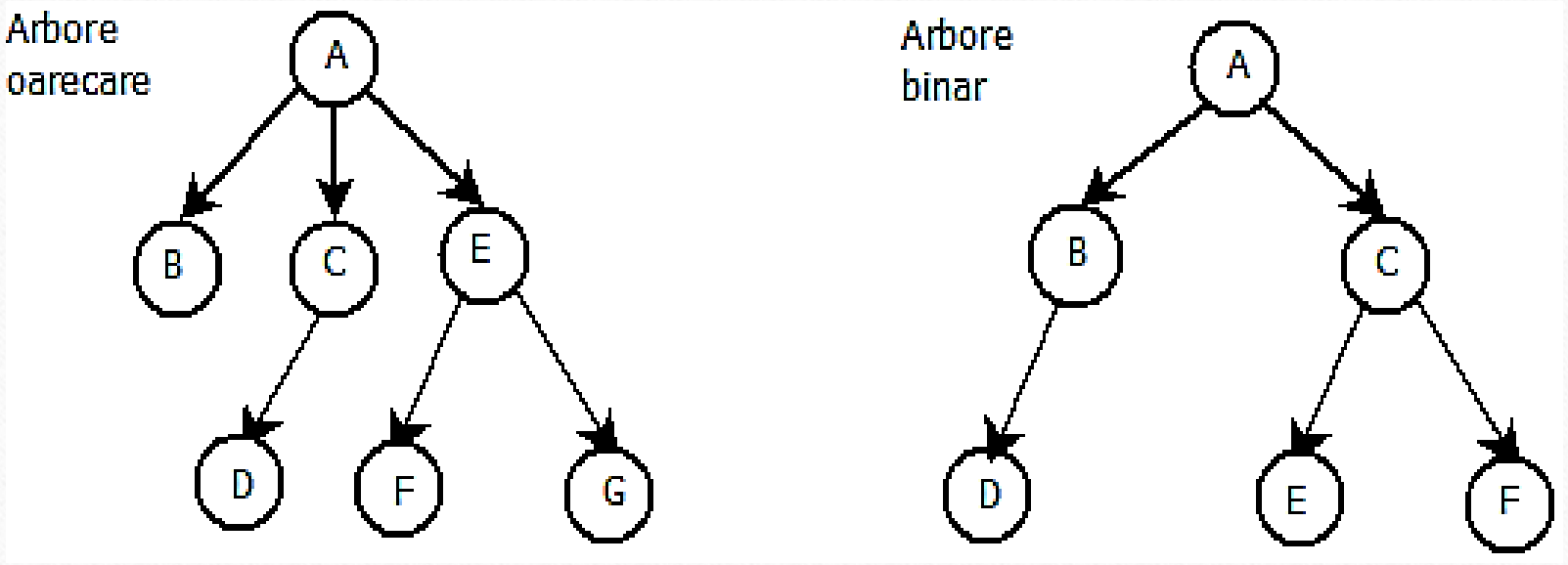
- **DEFINIRE NOȚIUNE ARBORE;**
- **EXEMPLE ARBORI;**
- **NOȚIUNI SPECIFICE ARBORI;**
- **EXEMPLIFICARE NOȚIUNI;**
- **PROPRIETĂȚI ARBORI;**
- **MODALITĂȚI DE REPREZENTARE A ARBORILOR OARECARE/BINARI ÎN MEMORIE;**
- **IMPLEMENTARE DINAMICĂ ARBORI OARECARE/BINARI;**
  - **DECLARAREA TIPULUI DE DATE ARBORE OARECARE/BINAR;**
  - **CREAREA UNUI ARBORE OARECARE/BINAR;**
  - **PARCURGEREA UNUI ARBORE OARECARE/BINAR.**
- **APLICAȚII PROPUSE.**

DEFINIRE NOȚIUNE ARBORE

Alte tipuri de date structurate și TDA?

- **Arbori ⇨ tip de date structurat+TDA;**
- **Arbore oarecare ⇨** structură de date alcătuită dintr-o mulțime nevidă și finită de elemente numite **noduri**, în care există un nod special numit **nod rădăcină**, iar celelalte noduri sunt partiționate în arbori cu rădăcină numiți **subarbori ai rădăcinii**;
- **Arbore binar ⇨** arbore în care fiecare nod (cu excepția nodurilor frunză) al arborelui are cel mult doi descendenți;

EXEMPLE ARBORI



NOȚIUNI SPECIFICE ARBORI

- **Nod rădăcină** ⇨ primul nod dintr-un arbore, singurul nod situat pe nivelul cel mai de sus al arborelui.
- **Subarbori** ⇨ restul nodurilor din arbore care formează câte un arbore**.**
- **Nod părinte sau tată** ⇨ nod ce prezintă fii, adică **descendenţi direcţi**.
- **Ascendent (predecesor nod)** ⇨ nod de pe un nivel superior unui alt nod;
- **Descendent (succesor nod)** ⇨ nod de pe un nivel inferior al unui nod.
- **Noduri fraţi** ⇨ noduri care au acelaşi ascendent direct (nod părinte sau tată);
- **Nodurile frunză** (noduri terminale) ⇨ noduri care nu au niciun succesor (nici un fiu, copil), adică nu prezintă descendenţi direcţi (noduri fii);
- **Nodul interior** (nod de ramificare sau non-frunză) ⇨ nod neterminal ce prezintă descendenţi direcţi (fii, copii).
- **Gradul unui nod** ⇨ numărul de descendenţi direcţi (fii) ai nodului.

NOȚIUNI SPECIFICE ARBORI

- **Nivelul unui nod** (*nivel(nod)*) este definit conform relației:

$$\text{nivel}(\text{nod}) = \begin{cases} 1, & \text{dacă nod=răd;} \\ \text{nivel}(\text{predecesor}(\text{nod}))+1, & \text{dacă nod}\neq\text{răd;} \end{cases}$$

- **Adâncimea unui nod x**, notată *h(y)* este definită recursiv conform relației:

$$h(y)= \begin{cases} 1, & \text{dacă } y \text{ este nod răd;} \\ h(x)+1, & \text{dacă } y \text{ este succesorul lui } x; \end{cases}$$

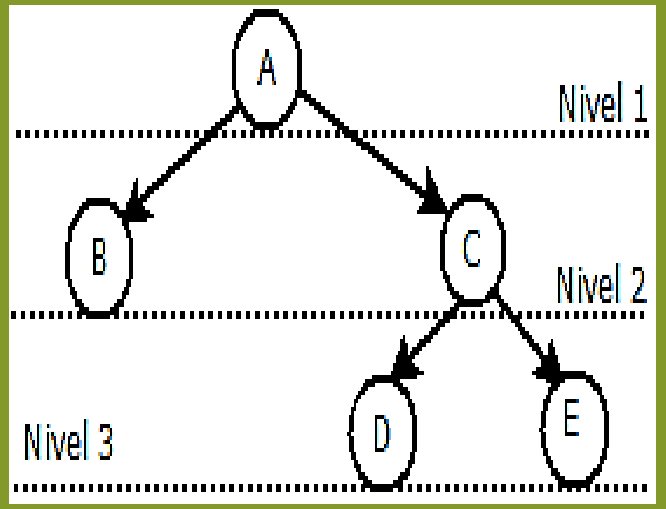
NOȚIUNI SPECIFICE ARBORI

- **Înălțimea unui arbore** *(h(arb))* este definită conform relației:

$$h(arb)=\begin{cases}0, & \text{dacă arborele este vid;}\\ \text{maxim}\big(\text{nivel}(nod)\big), & (\forall)nod\in N;\end{cases}$$

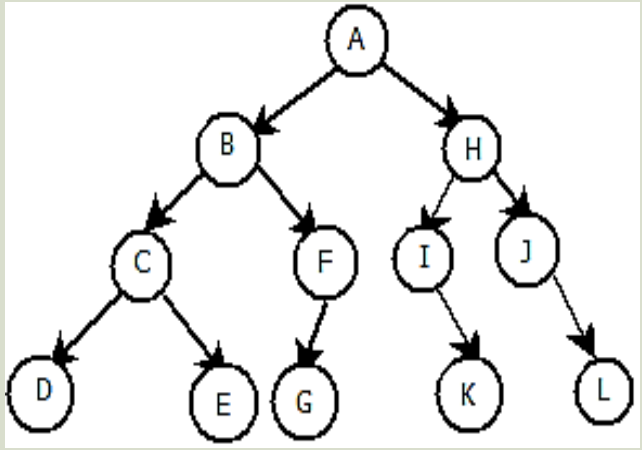
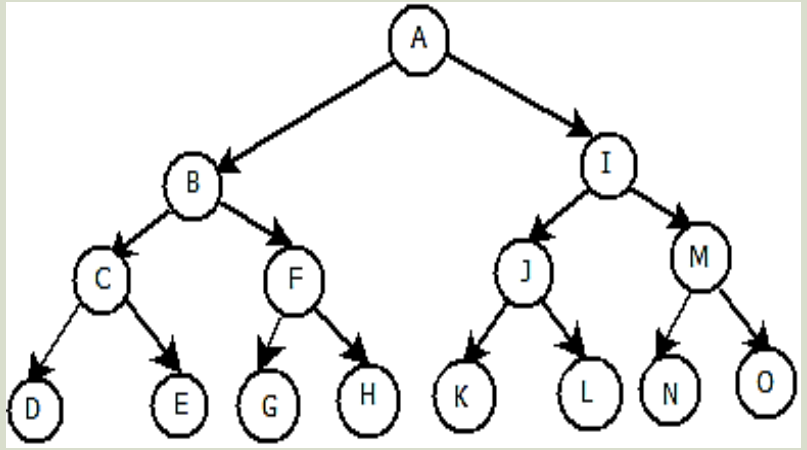
- Un **arbore** se numește **echilibrat** dacă pentru $(\forall)nod\in N$ și $(\forall)$ subarbori SA_1 și SA_2 este îndeplinită relația: $|h(SA_1)-h(SA_2)| \leq 1$;
- Un arbore este **perfect echilibrat** dacă:
  - $(\forall)nod\in N$ și $(\forall)$ doi subarbori SA_1 și SA_2 este îndeplinită condiția $h(SA_1)-h(SA_2)=0$ (subarborii au aceiași înălțime);
  - nivel (nod_1)=nivel(nod_2), pentru $(\forall)$ două frunze nod_1 și nod_2 din arbore;
  - toate nodurile interioare au toți succesorii posibili nevizi.

EXEMPLIFICARE NOȚIUNI



Nod A- nod rădăcină;
{B}- subarbore stâng; {C, D, E}- subarbore drept;
Nod C- nod părinte (tată) pentru nodurile D şi E;
Nodurile B şi C- noduri fii ai lui A;
Noduriel D şi E- noduri fii ai lui C;
Predecesor (D)=C;
Succesor (A)={B, C};
Nodurile B şi C- noduri fraţi;
Nodurile D şi E- noduri fraţi;
Nodurile B, D şi E- noduri frunză;
Nod C- nod interior;
Grad(A)=2; Grad(C)=2; Grad(B)=0; -B nod frunză;
nivel(A)=1; nivel (C)=2; nivel(D)=3;
h(A)=1; h(E)=3;
h(arb)=3;

EXEMPLIFICARE NOȚIUNI

Arbore echilibrat	Arbore perfect echilibrat
	

PROPRIETĂȚI ARBORI

- **Proprietăți arbori oarecare:**
  - într-un arbore există exact o cale care leagă oricare două noduri;
  - un arbore cu *n* noduri prezintă *n-1* muchii.
- **Proprietăți arbori binari:**
  - fiecare nod al arborelui binar prezintă exact doi fii (**arbore plin** sau **arbore propriu**);
  - arborele binar este un arbore propriu în care frunzele au aceeași adâncime (**arbore complet**).

MODALITĂȚI DE REPREZENTARE A ARBORILOR OARECARE/BINARI ÎN MEMORIE

- **Vectorii de taţi** ⇨ utilizează un tablou unidimensional în care se reţine pentru fiecare nod din arbore, eticheta sau adresa nodului său tată;

  ⇨ exemplu: tabel 7.1, pag. 126;

- **Reprezentarea fiu-frate (descendent stâng- frate drept- DSFD)** ⇨ pentru fiecare nod din arbore se memorează trei tipuri de informaţii, şi anume:
  - informaţia utilă (cheia);
  - adresa primului descendent stâng;
  - adresa primului frate drept.

  ⇨ exemplu: fig.7.10, pag. 126;

MODALITĂȚI DE REPREZENTARE A ARBORILOR OARECARE/BINARI ÎN MEMORIE

- **Tabele HASH** ⇨ fiecare nod din arbore se reţine lista descendenţilor direcţi;

  ⇨ exemplu: fig. 7.11, pag. 127;

- **Reprezentarea pe niveluri** ⇨ se utilizează un vector în care se memorează informaţiile nodurilor din arbore;

  ⇨ exemplu: fig. 7.12 şi tabel 7.2, pag. 127;

- **Parantezele incluse** (*nested parantheses*) ⇨ pentru fiecare nod din arbore se precizează între paranteze informaţia acestuia, urmată de nodurile descendente, dacă acestea există;

  ⇨ exemplu: fig.7.13, pag. 128;

- **Reprezentarea secvenţială** ⇨ se numerotează crescător, începând cu noul rădăcină, nodurile de pe nivelele arborelui, pornind de la stânga la dreapta şi se utilizează un vector în care se memorează informaţiile din nodurile arborelui;

  ⇨ exemplu: fig.7.14, tabel 7.3, pag. 128;

MODALITĂȚI DE REPREZENTARE A ARBORILOR OARECARE/BINARI ÎN MEMORIE

- **Reprezentarea prin intermediul pointerilor** ⇨ se utilizează informaţia utilă a nodurilor din arbore, un pointer către descendentul stâng, respectiv un pointer către descendentul drept al unui nod;

  ⇨ exemplu: fig. 7.16, pag. 129;

- **Reprezentarea cu referințe descendente** ⇨ pentru fiecare nod din arbore, se reţin următoarele elemente:
  - informaţia nodului;
  - adresa fiecărui descendent al nodului curent;
  - numărul de descendenţi.

  ⇨ exemplu: fig. 7.17, pag. 130.

IMPLEMENTARE DINAMICĂ ARBORI OARECARE

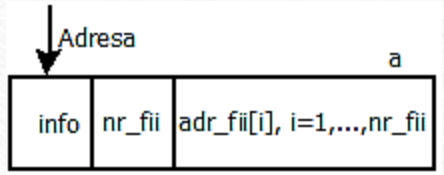
Declararea tipului de date structurat arbore oarecare	*Observaţii:*
typedef struct arbore **{int inf;** **int nr_fii;** **struct arbore*adr_fii[10];** **}arb_oarecare;**	• în reprezentarea cu referinţe descendente, pentru fiecare nod din arborele oarecare se reţin următoarele: • informaţia utilă a nodului/nodurilor (*int inf*); • numărul de fii (descendenţi) pentru fiecare nod (*int nr_fii*); • adresa fiecărui fiu (descendent) al nodului current (*struct arbore*adr_fii[10]*;-vector de pointeri către fii); • *adr_fii[10]* este practic un tablou cu adresele fiilor (descendenţilor);

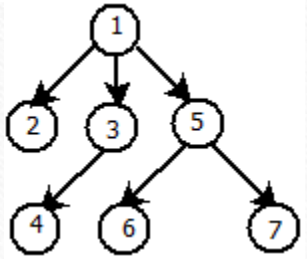
IMPLEMENTARE DINAMICĂ ARBORI OARECARE

- **Operațiile de bază asociate unui arbore oarecare sunt:**
  - **crearea arborelui oarecare** folosind reprezentarea cu referințe descendente;
  - **parcurgerea arborelui oarecare;**
    - **parcurgerea în adâncime: preordine** sau **parcurgere prefixată** (**RSD**- rădăcină, stânga, dreapta) și **postordine** sau **parcurgere postfixată** (**SDR**- stânga, dreapta, rădăcină);
    - **parcurgerea în lățime** sau pe **niveluri (Level Order Traversal)** a unui arbore oarecare are la bază parcurgerea în lățime (**Breadth First Search – BFS**) utilizată în cazul grafurilor; presupune vizitarea rădăcinii arborelui și apoi a tuturor succesorilor lui de pe nivelul 1, apoi se vizitează succesorii nodurilor de pe nivelul 2, ș.a.m.d; practic, se parcurg în ordine nodurile de pe fiecare nivel de la stânga la dreapta, începând de la primul nivel la ultimul nivel;
    - **Pseudocod BFS** – fig. 7.18, 7.19, pag. 131-132;

IMPLEMENTARE DINAMICĂ ARBORI OARECARE ⇨ Crearea arborelui oarecare

```
arb_oarecare*creare()
{int info;
arb_oarecare *a;
a=(arb_oarecare*)malloc(sizeof(arb_oare
care));
cout<<"\n Informatie nod:"; cin>>info;
a→inf=info;
cout<<"Nr. fii pt. nodul cu
informatia"<<info<<"este:";
cin>>a→nr_fii;
for(int i=1;i<=a→nr_fii;i++)
a→adr_fii[i]=creare();
return a;}
```

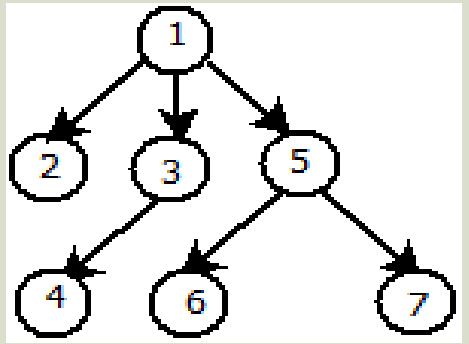




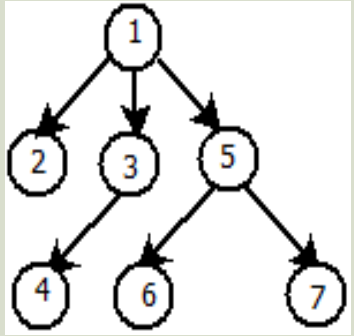
Crearea arborelui oarecare de mai sus, se realizează asfel:
Informaţie nod: 1
Nr. fii pt. nodul cu informaţia 1 este:3
Informaţie nod: 2
Nr. fii pt. nodul cu informaţia 2 este: 0
Informaţie nod: 3
Nr. fii pt. nodul cu informaţia 3 este: 1
Informaţie nod: 4
Nr. fii pt. nodul cu informaţia 4 este: 0
Informaţie nod: 5
Nr. fii pt. nodul cu informaţia 5 este: 2
Informaţie nod: 6
Nr. fii pt. nodul cu informaţia 6 este: 0
Informaţie nod: 7
Nr. fii pt. nodul cu informaţia 7 este: 0

IMPLEMENTARE DINAMICĂ ARBORI OARECARE ⇨Parcurgerea în adâncime a arborilor oarecare

Parcurgerea în preordine (RSD)	Rezultat

void RSD(arb_oarecare*t)
{if(t!=NULL)
{ cout<<t→inf<<",";
for(int i=1;i<=t→nr_fii;i++)
RSD(t→adr_fii[i]);
}
}
	
	RSD: 1, 2, 3, 4, 5, 6, 7.

IMPLEMENTARE DINAMICĂ ARBORI OARECARE ⇨ Parcurgerea în adâncime arborelui oarecare

Parcurgerea în postordine (SDR)	Rezultat
void SDR(arb_oarecare*t) {if(t!=NULL) {for(int i=1;i<=t→nr_fii;i++) RSD(t→adr_fii[i]); cout<<t→inf<<","; } }	 **SDR:** 2, 3, 4, 5, 6, 7, 1.

IMPLEMENTARE DINAMICĂ ARBORI OARECARE ⇨ Parcurgerea pe niveluri a arborelui oarecare

- Implementarea **parcurgerii pe niveluri (BFS)** a unui arbore oarecare se realizează astfel:
  - se utilizează o coadă (*arb_oarecare*coada[30];*) pentru parcurgerea pe nivele a arborelui oarecare; pentru gestionarea cozii se folosesc variabilele *prim* şi *ultim*;
  - în coadă se memorează toate adresele nodurilor (inclusiv adresele fiilor/descendenţilor) arborelui oarecare;
  - pentru adăugarea unui nod la sfârşitul cozii conform principiului FIFO, s-a utilizat funcţia cu prototipul *void adauga_nod(arb_oarecare*a);*
  - primul nod care este adăugat şi extras în/din coadă este nodul rădăcină (*adauga_nod(rad);*);
  - atâta timp cât nu au fost parcurse toate nodurile arborelui oarecare, se extrage (scoate) un nod din coadă;
  - pentru extragerea unui nod din coadă s-a utilizat funcţia cu prototipul *arb_oarecare *scoate_nod()*;
  - pentru fiecare nod extras din coadă se afişează informaţia utilă a acestuia şi apoi toţii fii (descendenţii) nodului extras sunt adăugaţi la sfârşitul cozii;
  - procedeul se reia pentru toate nodurile arborelui oarecare.
- Codul sursă complet asociat **parcurgerii pe niveluri (BFS)** a unui arbore oarecare este prezentat în fig. 7.24, pag. 135-137;
- Exemplificare parcurgere pe nivele (algoritmul **BFS**) a unui arbore oarecare ⇨ pag. 138.

IMPLEMENTARE DINAMICĂ ARBORI BINARI

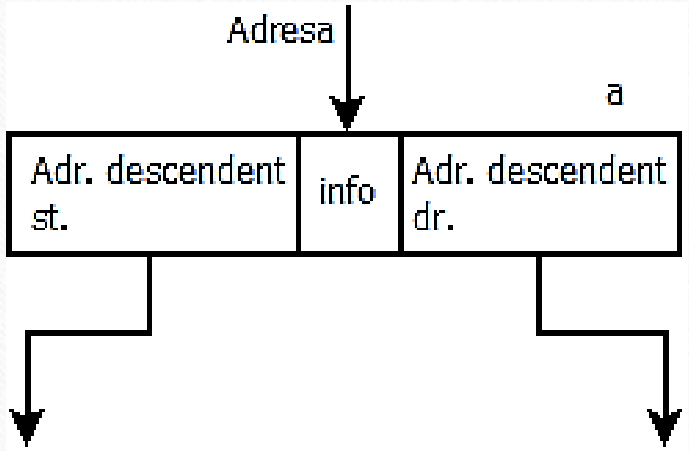
- **Operaţiile de bază** sunt:
  - **crearea arborelui binar;**
  - **parcurgerea arborelui binar**, aceasta fiind principala operaţie care se aplică arborilor, ce presupune vizitarea tuturor nodurilor arborelui dat, în vederea accesării informaţiei utile memorate în acestea; **tipurile de parcurgeri** care pot fi efectuate asupra unui arbore binar sunt:
    - **parcurgerea în adâncime: preordine** sau **parcurgere prefixată** (**RSD**- rădăcină, stânga, dreapta), **inordine** sau **parcurgere infixată** (**SRD**-stânga, rădăcină, dreapta) **şi postordine** sau **parcurgere postfixată** (**SDR**-stânga, dreapta, rădăcină);
    - **parcurgerea în lăţime** sau pe **niveluri (Level Order Traversal)**, ce presupune vizitarea rădăcinii arborelui **şi** apoi a tuturor succesorilor lui de pe nivelul 1, apoi se vizitează succesorii nodurilor de pe nivelul 2, **ş**.a.m.d.

IMPLEMENTARE DINAMICĂ ARBORI BINARI

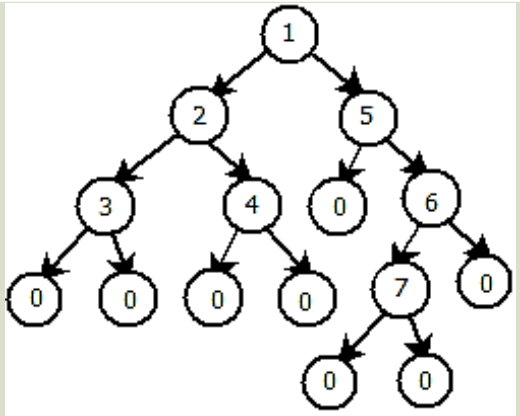
Declararea tipului de date structurat arbore binar	*Observaţii:*
typedef struct arbore **{int inf;** **struct arbore*st,*dr;** **}arbbin;**	• fiecare nod al arborelui prezintă trei câmpuri: *inf*, **st, *dr*; • câmpul *inf* reprezintă informaţia utilă a nodurilor din arborele binar; • câmpul *st* reprezintă adresa la subarborele stâng, acesta fiind un pointer către tipul de date *struct arbore*; • câmpul *dr* reprezintă adresa la subarborele drept, acesta fiind un pointer către tipul de date *struct arbore*;

IMPLEMENTARE DINAMICĂ ARBORI BINARI ⇨ Crearea arborelui binar

```
arbbin *creare()
{arbbin*a; int info;
 cout<<"Informatie nod:";cin>>info;
 if(info!=0)
  {a=(arbbin*)malloc(sizeof(arbbin));
   a→inf=info;
   a→st=creare(); a→dr=creare();
   return a;}
  else return NULL;}
```



IMPLEMENTARE DINAMICĂ ARBORI OARECARE ⇨ Parcurgerea în adâncime a arborelui binar

RSD	SRD	SDR	Rezultat:
*void RSD(arbbin*t)* *{if(t!=NULL)* *{cout<<t→inf;* *RSD(t→st);* *RSD(t→dr);* *}* *}*	*void SRD(arbbin*t)* *{if(t!=NULL)* *{SRD(t→st);* *cout<<t→inf;* *SRD(t→dr);* *}* *}*	*void SDR(arbbin*t)* *{if(t!=NULL)* *{SDR(t→st);* *SDR(t→dr);* *cout<<t→inf;* *}* *}*	 **RSD**: 1, 2, 3, 4, 5, 6, 7. **SRD**: 3, 2, 4, 1, 5, 7, 6. **SDR:** 3, 4, 2, 7, 6, 5, 1.

IMPLEMENTARE DINAMICĂ ARBORI BINARI ⇨ Parcurgerea pe niveluri a arborelui binar

- Implementarea **parcurgerii pe niveluri (BFS)** a unui arbore binar se poate realiza prin:
  - folosirea unei funcții pentru afișarea pe nivele (de la stânga la dreapta) a tuturor nodurilor arborelui binar; practic, se afișează toate nodurile de pe primul nivel, apoi de pe al doilea nivel, până la nivelul *h*, unde *h* reprezintă înălțimea arborelui binar;
  - folosirea unei cozi în care pentru fiecare nod vizitat, descendenții acestuia sunt plasați în respectiva coadă FIFO; inițial, coada conține doar rădăcina subarborelui, iar atunci când un nod extras din coadă este vizitat, descendenții acestuia sunt adăugați în coadă.

IMPLEMENTARE DINAMICĂ ARBORI BINARI ⇨ Parcurgerea pe niveluri a arborelui binar

- Pentru că varianta de implementare a **BFS** prin utilizarea unei cozi a fost utilizată în cazul parcurgerii pe nivele a arborilor oarecare), în cazul de față s-a ales prima metodă de implementare, metodă mai simplă, în care s-a folosit:
  - o funcţie pentru afișarea pe nivele (de la stânga la dreapta) a tuturor nodurilor arborelui binar (*void afisare_nivel_BFS(arbbin*rad)*), funcție ce apelează la rândul ei alte două funcţii, și anume:
  - o funcţie pentru calculul înălţimii arborelui binar (*int inaltime (arbbin*t)*);
  - o funcţie pentru afișarea tuturor nodurile de pe un anumit nivel al arborelui binar (*void afisare_noduri_nivel(arbbin*rad, int nivel)*);
- Codul sursă complet asociat **parcurgerii pe niveluri (BFS)** a unui arbore binar este prezentat în fig. 7.30, pag. 145.

APLICAȚII PROPUSE

- Testarea aplicațiilor 7.2.1, 7.2.2 și 7.2.3, pag. 154-160;
- Aplicații propuse: aplicațiile din cadrul lucrării de laborator nr. 11, pag. 161;
- **Obs: se va utiliza cartea "*Elemente de proiectarea algoritmilor. Ghid teoretic și practic*", Șef lucr. dr. mat. Cărbureanu Mădălina, Editura Universității Petrol-Gaze din Ploiești, 2021.**

Să vă fie de folos și spor la lucru!