



# ARBORE OARECARE

## Caracteristici:

- Graf aciclic, conex si orientat;
- Un nod radacina (root);
- Reprezentari mai eficiente fata de grafuri: FIU-FRATE, structuri in memoria heap;
- Traversare: regula de vizitare a nodurilor;
- Topologii particulare: binari, de cautare, echilibrati etc.

# ARBORE OARECARE

Reprezentarea FIU-FRATE a unui arbore –  
structura unui nod din arbore:

- ID nod pentru primul descendent;
- ID frate a primului descendent;
- Informatia stocata de nod.

# ARBORE OARECARE

## Structuri de memorare:

- Numarul de noduri al arborelui;
- ID nod radacina;
- Vector primi descendenti;
- Vector frati ai primilor descendenti.

## Exemplu:

```
int Rad = 1, n = 15;
```

```
int Fiu[] = {2, 6, 0, 9, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

```
int Frate[] = {0, 3, 4, 5, 0, 7, 8, 0, 10, 11, 0, 13, 14, 15, 0};
```

# ARBORE OARECARE

**Reprezentarea prin structuri memorate in heap:**

- **ID nod;**
- **Vector/lista/alta structura liniara cu adrese ale descendentilor nodului;**

**Eventual, se precizeaza numarul maxim de descendenti (memorare adrese descendenti in vector).**

# ARBORE OARECARE

```
/* definirea structurii unui nod de arbore oarecare cu
maxim 10 descendenti */
struct nodArb{
    int inf;
    nodArb* fiu[10];
};
```

...

```
/* definirea variabilei de gestionare a structurii
arborescente */
```

```
nodArb* radaO = NULL;
```

...



# ARBORE OARECARE

## Traversarea arborilor oarecare:

- **Preordine:**
  - **Selectia nodului radacina ca nod curent;**
  - **Se viziteaza nodul curent;**
  - **Pentru nodul curent se viziteaza sub-arborii in ordinea data in structura de stocare a adreselor acestora;**

# ARBORE OARECARE

## Traversarea arborilor oarecare (continuare):

- **Postordine:**
  - **Se viziteaza sub-arborii in ordinea data in structura de stocare a adreselor acestora;**
  - **Se viziteaza nodul radacina (dupa vizitarea tuturor nodurilor din sub-arbori);**



# ARBORE OARECARE

## Traversarea arborilor oarecare (continuare):

- Pe niveluri:
- Se viziteaza nodurile in ordinea crescatoare a distantelor fata de radacina.

# ARBORE BINAR

## Caracteristici:

- Arbore oarecare cu maxim doi descendenți;
- Orice sub-arbore respecta restricția de la punctul anterior;
- Topologii particulare: arbori binari de căutare etc;
- Reprezentare: structuri alocate în heap.

# ARBORE BINAR

Definire structura nod arbore binar:

```
struct arbBin
{
    int cheie;
    arbBin *st, *dr;
};
```

# ARBORE BINAR DE CAUTARE

## Caracteristici:

- Arbore binar;
- Noduri aranjate in structura conform proprietatilor:
  - Fiecare nod are atasata o informatie dintr-o multime ordonata de valori (cheie);
  - Pentru fiecare nod, valoarea de cheie este mai mare decat orice valoare de cheie din subarborele din stanga;

# ARBORE BINAR DE CAUTARE

- Noduri aranjate in structura conform proprietatilor (continuare):
  - Pentru fiecare nod, valoarea de cheie este mai mica decat orice valoare de cheie din subarborele din dreapta;
  - Nu exista valori de cheie duplicate in cadrul arborelui;
- Traversarea in inordine asigura obtinerea informatiilor atasate nodurilor in ordinea crescatoare a valorilor de cheie.



# ARBORE BINAR DE CAUTARE

## Operatii:

- Inserarea unui nod;
- Stergerea unui nod;
- Traversarea arborelui: preordine, inordine, postordine, pe niveluri;
- Inaltime arbore;
- Echilibrare arbore;
- Numar de noduri frunza;
- Etc.



# ARBORE BINAR DE CAUTARE

Dupa efectuarea unei operatii, arborele isi pastreaza caracteristicile.

Inserarea unui nod:

```
nod * inserare_nod(nod *rad,int k)
{
    if (rad)
    {
        if (k<rad->info) rad->stg=inserare_nod(rad->stg,k);
        else
            if (k>rad->info) rad->drt=inserare_nod(rad->drt,k);
            else cout<<"\nNodul exista in arbore!";
            return rad;
    }
    else
    {
        nod *p=new nod;
        p->stg=NULL;
        p->drt=NULL;
        p->info=k;
        return p;
    }
}
```