



### Laboratorul 5

#### *Arbori. Arbori binari.*

---

Matematic, un arbore este un **graf neorientat conex aciclic**.

În știința calculatoarelor, termenul de arbore este folosit pentru a desemna o structură de date care respectă definiția de mai sus, însă are asociate un **nod rădăcină** și o **orientare înspre sau opusă rădăcinii**.

Arborii sunt folosiți în general pentru a modela o ierarhie de elemente. Astfel, fiecare element (nod) poate deține un număr de unul sau mai mulți descendenți, iar în acest caz nodul este numit **părinte al nodurilor descendente**.

Fiecare nod poate avea un singur nod părinte. Un nod fără descendenți este un nod terminal, sau **nod frunză**. În schimb, există un singur nod fără părinte, iar acesta este întotdeauna **rădăcina arborelui (root)**.

Un arbore binar este un caz special de arbore, în care fiecare nod poate avea maxim doi descendenți: **nodul stâng** și **nodul drept**.

```
typedef int Item;
```

```
typedef struct node  
{  
    Item elem;  
    struct node *lt;  
    struct node *rt;  
} TreeNode, *TTree;
```

### PARCURGEREA ARBORILOR

#### Preordine

- Se parcurge rădăcina
- Se parcurge subarborele stâng
- Se parcurge subarborele drept



### Inordine

- Se parcurge subarborele stâng
- Se parcurge rădăcina
- Se parcurge subarborele drept

### Postordine

- Se parcurge subarborele stâng
- Se parcurge subarborele drept
- Se parcurge rădăcina

### CERINȚE

1. Să se scrie definiția completă a următoarelor funcții de lucru cu arborii în fișierul **Tree.h**:

*void Init(TTree \*t, Item x)* – inițializează un nod cu valoarea x

*void Insert(TTree \*t, Item x)* – adaugă un nod cu valoarea x în arbore. La sfârșit, pentru fiecare nod, subarborele stâng va conține valori mai mici decât cea a nodului, iar cel drept va conține valori mai mari decât cea a nodului (principiul Arborilor Binari de Căutare).

*void PrintPostorder(TTree t)* – afișează arborele în postordine

*void PrintPreorder(TTree t)* – afișează arborele în preordine

*void PrintInorder(TTree t)* – afișează arborele în inordine

*void Free(TTree \*t)* – eliberează spațiul de memorie alocat arborelui

*int Size(TTree t)* – returnează dimensiunea arborelui (numărul de noduri)

*int NoLeaves (TTree t)* – returnează numărul de frunze din arbore

*TTree Where (TTree t, Item x)* – returnează adresa nodului care îl conține pe x

*int MaxDepth(TTree t)* – returnează adâncimea maximă a arborelui

**(10 X 0.75 puncte = 7.5 puncte)**



2. În fișierul **testTree.c**, să se scrie definiția completă a funcției:

*void mirror(TTree \*t)*

care construiește oglinditul unui arbore binar.



**(1 punct)**

3. În fișierul **testTree.c**, să se scrie definiția completă a funcției:

*int sameTree(TreeNode \*t1, TreeNode \*t2)*

care verifică dacă doi arbori sunt identici (sunt formați din noduri cu valori identice, aranjate identic din punct de vedere al dispunerii în arbore).

**(1 punct)**