



Laboratorul 6

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 Link
```

```
{  
    Item elem;  
    struct Link *l;  
    struct Link *r;  
} TreeNode;
```

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(TreeNode **t, Item x)* – inițializează un nod cu valoarea x

*void Insert(TreeNode **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(TreeNode *t)* – afișează arborele în postordine

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

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

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

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

*int maxDepth(TreeNode *t)* – returnează adâncimea maximă a arborelui

(8 X 0.75 puncte = 6 puncte)



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

*void mirror(TreeNode **t)*

care construiește oglinditul unui arbore binar.



(1.5 puncte)

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).

(1.5 puncte)