Programmation structurée — Arbres binaires

Dans les exercices de cette série, on considère des arbres binaires de nombres entiers. Ces arbres sont codés dans ces structures :

```
class Noeud
{
        int info;
        Noeud * fg, * fd;
};

class Arbre
{
        Noeud *racine;
    public :
        Arbre();
        ~Arbre();
...
};
```

1. Constructeur
Écrire un constructeur d'arbre vide, un constructeur d'arbre avec un pointeur de Noeud en argument et un constructeur de Noeud avec un int et 2 pointeurs de Noeud.

```
Arbre A(new Noeud(5,new Noeud(4, null, null),new Noeud(2,null,new Noeud(12,null,null)))) ;
```

2. Destructeur
Écrire les destructeurs de Noeud et Arbre

3. Destructeur
Écrire une fonction qui libère la mémoire allouée par l'arbre.

4. Profondeur d'un nœud
Fonction qui affiche pour chaque nœud sa profondeur.

5. Hauteur d'un arbre
Fonction qui retourne la hauteur d'un arbre. La hauteur d'un arbre vide est 0, la hauteur d'un arbre à 1 nœud est 1.

6. Maximum
Fonction qui détermine le maximum des valeurs contenues dans l'arbre.

7. Maximum dans un arbre ordonné
Même question, mais en supposant que l'arbre est ordonné.

8. Insertion dans un arbre ordonné
Écrire une fonction qui insère, en feuille, une valeur dans un arbre ordonné.

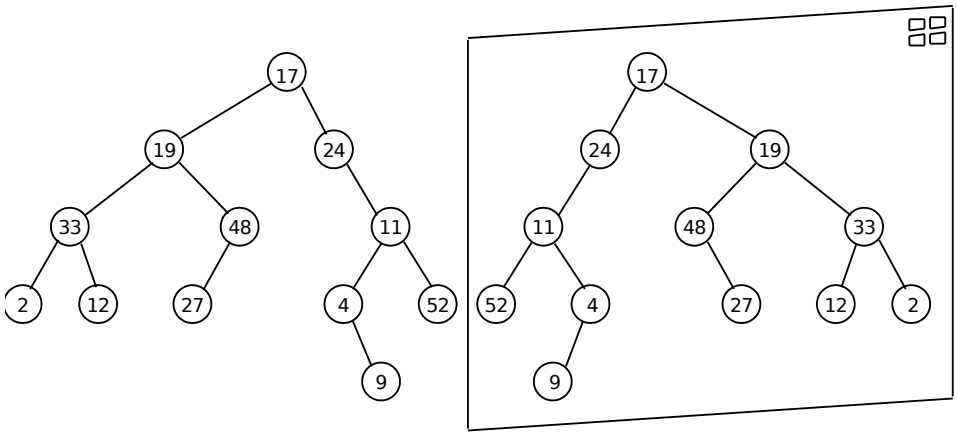
9. Égalité de deux arbres
Fonction booléenne testant si deux arbres sont égaux, c'est à dire s'ils contiennent les mêmes valeurs aux mêmes « endroits ».

10. Constructeur de recopie

Ecrire une fonction d'en-tête `noeud * arbre::copie(noeud * n)` qui fabrique la copie d'un sous-arbre et l'utiliser pour écrire le constructeur de recopie.

11. Fabrication de l'arbre miroir d'un arbre binaire.

Fabrication d'un nouvel arbre binaire, image inversée de l'arbre de départ.



12. Deux arbres binaires sont-ils des images inversées l'un de l'autre ?

Fonction booléenne testant si un arbre binaire est image miroir d'un autre.

13. Sous-arbre

Fonction booléenne testant si un arbre est contenu dans un autre.