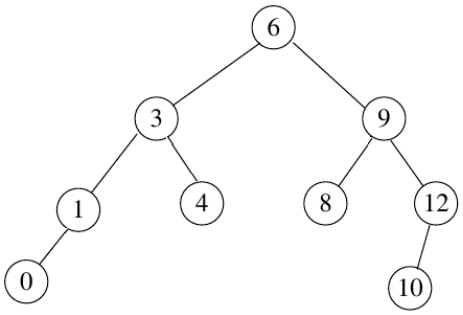
TD 7 – Arbres binaires de Recherche (ABR)

Préambule

Un arbre binaire de recherche (ABR) est un arbre binaire qui possède les propriétés suivantes :

- tous les nœuds du sous-arbre de gauche d'un nœud ont une valeur inférieure à la sienne ;
- tous les nœuds du sous-arbre de droite d'un nœud ont une valeur supérieure à la sienne.

✎ Exemple ci-contre.



Exercice 1 : Mise en place d'un arbre binaire de recherche

Considérons la structure ci-dessous d'un (nœud d'un) ABR permettant de contenir des valeurs entières.

✎ Dans ce TD – à vocation pédagogique – le contenu d'un nœud est réduit à une simple valeur entière, mais cela pourrait être tout autre chose *qui peut être ordonné* selon le problème.

```c
typedef struct _noeud
{
  int valeur;
  struct _noeud * pere;    // accès au parent (on découvrira plus tard pourquoi c'est nécessaire)
  struct _noeud * fils_gauche;
  struct _noeud * fils_droite;
} noeud;
```

À l'aide du type structuré `noeud` ci-dessus, définir en `C` la fonction

`void insertion(noeud ** p_arbre, int valeur)`

qui insert dans l'ABR pointé par `p_arbre` (*i.e.* dans `*p_arbre`) un nouvel élément dont le contenu est `valeur`.

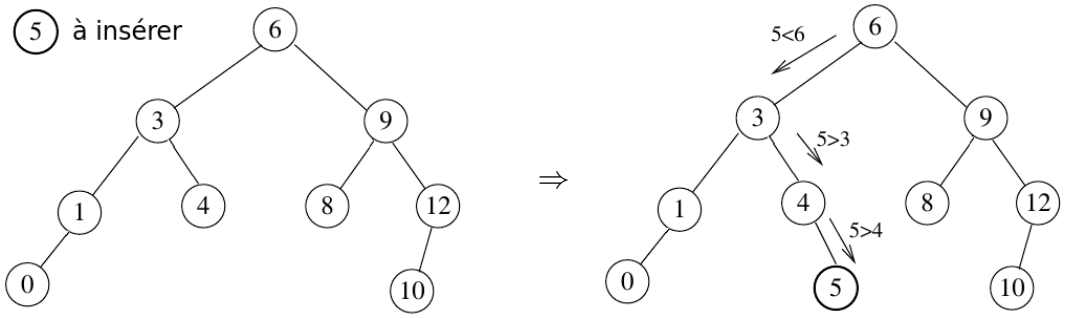
- Pour cela,
  1. chercher la position où insérer l'élément, puis
  2. si la valeur n'est pas déjà présente dans l'ABR, alors allouer dynamiquement le nouvel élément, l'initialiser et l'insérer,
  3. sinon, pas d'insertion.

ϟ Il est possible que ce soit le premier élément de l'arbre (si l'arbre est vide, *i.e.* est `NULL`).

Bien tester cette fonction, notamment à l'aide du débogueur et à l'aide d'une fonction d'affichage codée lors du TD précédent (puisqu'un ABR est un arbre binaire).

Exemple possible pour tester : insérer respectivement les éléments 6, 3, 9, 1, 0, 4, 12, 10 puis 8.

Ci-dessous une illustration montrant l'insertion de l'élément 5 dans l'ABR de test :



Avant l'insertion — Après l'insertion

Exercice 2 : Recherche dans un ABR

Comme son nom l'indique, l'intérêt d'un ABR réside dans le fait de pouvoir y chercher efficacement une valeur.

Définir en `C` la fonction `rechercher` qui permet de trouver une valeur dans un ABR. Si la valeur est trouvée alors la fonction retourne un accès sur le maillon concerné, sinon, `NULL` est retourné.

- Quid de la complexité de votre implémentation ?

Exercice 3 : Minimum et Maximum

Écrire en `C` les deux fonctions suivantes :

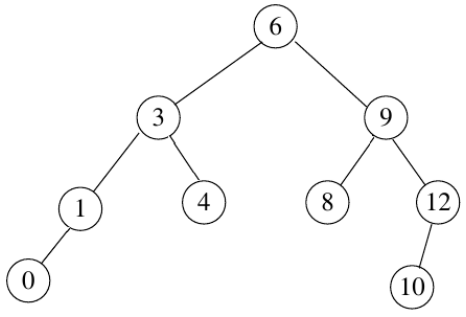
- `noeud* minimum(noeud * arbre)`
  Fonction qui retourne l'accès à l'élément dont la valeur est le minimum dans l'arbre fourni en paramètre.
- `noeud* maximum(noeud * arbre)`
  Fonction qui retourne l'accès à l'élément dont la valeur est le maximum dans l'arbre fourni en paramètre.

Exercice 4 : Nœuds précédent et suivant

On se propose de définir en `C` les fonctions suivantes :

- `noeud* suivant(noeud * element)`
  Fonction qui retourne l'accès à l'élément dont la valeur est la suivante de celle de l'élément fourni en paramètre.
- `noeud* precedent(noeud * element)`
  Fonction qui retourne l'accès à l'élément dont la valeur est la précédente de celle de l'élément fourni en paramètre.

Exemples :



- le nœud précédent de 1 est 0
- le nœud suivant de 3 est 4

↯ le précédent de 6 est 4 ; le précédent de 8 est 6

↯ le suivant de 9 est 10 ; le suivant de 4 est 6

Exercice 5 : Suppression d'un nœud dans un ABR

On se propose d'implémenter en `C` la fonction `suppression` permettant de supprimer l'élément correspondant à la valeur passée en paramètre dans l'ABR (passé en paramètre lui-aussi).

ϟ Après la suppression, l'ABR doit toujours garder ses propriétés !!!

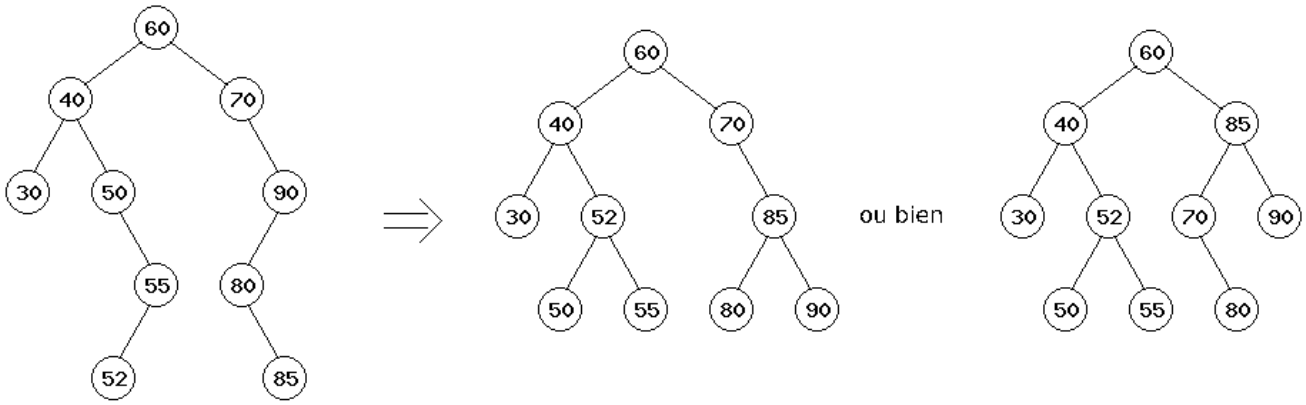
Exercice 6 (supplémentaire) : Conversion : ABR → tableau trié

On se propose maintenant de coder en `C` la fonction `abr_2_tab` qui à partir d'un ABR fourni en paramètre retourne un tableau dont les éléments sont triés dans l'ordre croissant.

Exercice 7 (optionnel) : Équilibrer un ABR

Proposer un algorithme, sans nécessairement le coder en `C`, pour équilibrer un ABR, c'est à dire que pour tout nœud, la hauteur de son sous-arbre gauche et la hauteur de son sous-arbre droit soient égales à $\pm$ 1 près.

Exemple :



Voir aussi : `https://fr.wikipedia.org/wiki/Arbre_AVL`

Aide : `https://www.youtube.com/watch?v=5C8bLQBjcDI` → vidéo (en anglais) présentant les 4 situations possibles de déséquilibre et 4 fonctionnalités pour y remédier.

Exercice 8 (optionnel) : ABR automatiquement équilibré ⇔ AVL

Proposer un algorithme permettant d'ajouter un élément dans un ABR de sorte que l'ABR soit toujours équilibré.

Proposer un algorithme permettant de supprimer un élément dans un ABR de sorte que l'ABR soit toujours équilibré.