UNIVERSITÉ DE MONTPELLIER
UNIVERSITÉ DE MONTPELLIER - FACULTÉ DES SCIENCES ET TECHNIQUES
PLACE EUGÈNE BATAILLON - 34095 MONTPELLIER CEDEX 5

Licence 2 informatique – 2016/2017

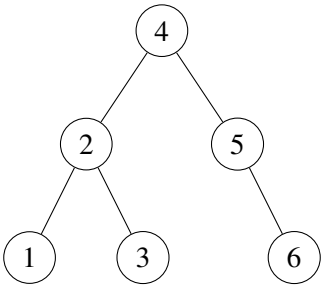
HLIN302 – TP noté

Programmation impérative avancée

Pascal GIORGI et Laurent IMBERT

Vous avez vu durant les séances de TD/TP un structure de données appelée « tas binaire ». L'intérêt d'une telle structure de données est de pouvoir effectuer un certain nombre d'opérations en temps logarithmique (au lieu de temps linéaire pour les tableaux par exemple). C'est le cas pour l'insertion ou la recherche d'un élément dans la structure. Cette structure s'appuie sur la notion d'arbre binaire. En informatique il existe plusieurs autres structures de données qui s'appuient sur des arbre binaires.

Nous considérons dans cet exercice la notion d'arbre binaire de recherche. Un arbre binaire de recherche (**ABR**) est un arbre binaire dans lequel chaque nœud possède une valeur telle que : chaque nœud du sous-arbre droit possède une valeur strictement supérieure à celle du nœud considéré, alors que chaque nœud du sous-arbre gauche possède une valeur inférieure ou égale. L'arbre binaire ci-dessous est un **Arbre Binaire de Recherche** :



En effet, le nœud racine est de valeur 4. Tous les nœuds qui apparaissent dans l'arborescence à gauche de cette racine sont tous de valeur plus petite que 4 alors que tous ceux dans l'arborescence à droite sont tous plus grand que 4. Cette propriété étant vérifié recursivement sur chacun des nœuds, nous avons bien affaire à un ABR. Remarque : un arbre avec seulement un seul nœud est un ABR. L'intérêt des ABR est de proposer une insertion et une recherche d'élément en temps logarithmique en le nombre de noeuds. De même, l'accès au plus petit et plus grand élément se fait en temps logarithmique. Par contre le parcours ordonné des données se fait en temps linéaire.

Une façon classique de représenter des ABR sur des entiers est de s'appuyer sur une structure de donnée récursive (à la manière des listes). La classe `Noeud` ci-dessous permet de faire cela ; on considérera qu'un ABR est identifié par son nœud racine :

```
1  class Noeud {
2    private:
3    int      val;
4    Noeud*   gauche;
5    Noeud*   droit;
6    ...
7  };
```

L'objectif de ce TP noté est de proposer une classe permettant la construction et la manipulation d'objet de type ABR (Abre Binaire de Recherche) avec un stockage récursif. En particulier, votre code doit permettre de faire fonctionner le code suivant :

```
#include <iostream>
#include <fstream>
#include "abr.h"


int main(){
  ABR<int>    A1,A3;
  ABR<double> A2;


  A1.insert(1);  A1.insert(8);
  A1.insert(3);  A1.insert(6);
  A3=A1;
  A1.insert(5);  A1.insert(4);
  A1.insert(7);  A1.insert(2);

  ifstream file("abr.txt");
  A2.load(file); // charge les valeurs depuis un fichier

  double x; std::cin>>x;
  if (A2.find(x))
  std::cout<<x<<" est dans A2"<<std::endl;


  std::cout<<"A1=";
  A1.affiche(std::cout);
  std::cout<<"A2=";
  A2.affiche(std::cout);
  std::cout<<"A3=";
  A3.affiche(std::cout);

  return 0;
}
```

abr.txt

```
2.5 8.3 1.1 1.2 2.1
4 5.4 1.1 12
```

Cahier des charges :

- la méthode `find` détermine si le paramètre appartient à l'ABR
- la méthode `load` permet d'insérer un ensemble de valeur dans l'ABR à partir d'un flux de fichier.
- la méthode `affiche` permet d'afficher les valeurs de l'ABR dans **l'ordre croissant**, sur un flux de sortie.
- votre classe doit être complète selon les spécifications des classes vues en cours.
- ATTENTION :
  - votre classe `ABR` doit permettre de construire des ABR vides.
  - un ABR n'est pas un arbre binaire parfait incomplet, à l'inverse des tas binaires. Par conséquent, on ne peut pas les stocker avec un tableau.

Modalités

Vous devez déposer une archive zip ou tar.gz contenant l'ensemble de vos fichiers avant la fin de la séance sur l'espace Moodle du cours prévu à cet effet.