Travaux Pratiques d'AP3 n°3

—Licence Informatique, 3ième année, 1ier semestre—

Implantation des arbres binaires

Le but de ce TP est d'implanter le type abstrait des arbres binaires.

Nous allons étudier deux façons d'implanter les arbres binaires, l'une avec un type somme et l'autre avec des pointeurs. Cette dernière implantation sera transposée en C.

Le langage OCaml permet de définir des signatures de module et des modules qui respectent cette signature. Par exemple, la signature d'un module pour le type abstrait des booléens est :

```
module type Bool =
  sig
    type bool

    val vrai : bool
    val faux : bool

    val neg : bool -> bool
    val ou : bool * bool -> bool
    val et : bool * bool -> bool
  end
```

Un module qui implante cette signature est, par exemple :

```
module MyBool : Bool =
  struct
    type bool = int

    let vrai = 1
    let faux = 0

    let neg(b : bool) : bool = if b = 0 then 1 else 0

    let ou(b1, b2 : bool * bool) = (* implantation *)

    let et(b1, b2 : bool * bool) : bool = (* implantation *)

  end
```

Pour bien comprendre comment cela fonctionne, il vous est conseillé de manipuler le code précédent en le recopiant dans un fichier ayant l'extension `.ml` que vous utiliserez avec l'interpréteur.

Il y a une contrainte syntaxique, les noms des modules commencent par une majuscule

▸ **Exercice 1. (Signature des arbres binaires)** *Écrivez une signature pour le type abstraits des arbres binaires.*

▸ **Exercice 2. (Implantation avec un type somme)**

1. *Définissez un type somme qui permet de représenter les arbres binaires.*
2. *Utilisez le type somme que vous avez défini pour écrire un module qui implante la signature des arbres binaires.*
3. *Utilisez votre module à la place du fichier* `btree.cmo` *et vérifiez que le code que vous avez écrit précédemment continue de fonctionner (sauf pour les fonctions d'affichage car vous ne les avez pas implantées). Vous pouvez faire cela de différentes façon. Une façon simple est de mettre la signature et son implantation dans un fichier (par ex.* `mybtree.ml`*) et d'utiliser la commande* `#use "mybtree.ml"` *à la place de* `#load "btree.cmo"`*.*

▸ **Exercice 3. (Implantation avec des pointeurs)** *Dans les langages impératifs les types sommes ne sont généralement pas utilisés bien qu'ils existent dans la plupart des cas (par exemple en C les* `union` *sont une réalisation des types sommes).*

L'usage est d'utiliser un type produit (une structure `struct` *en C) et des pointeurs. Nous allons réaliser cela en OCaml.*

En OCaml, les types produits s'écrivent et s'utilisent de la façon suivante :

```
type personne = {nom : string ; age : int}

let drole_de_type = {nom = "Toto" ; age = 102}

let majeur(p : personne) : bool = (p.age >= 18)
```

Vous avez aussi besoin de la notion de pointeur. Cela peut être simulé en OCaml. Dans la section « Ressources pour les TPs » récupérez les fichiers `pointer.cmo` *et* `pointer.cmi` *ainsi que la documentation du module.*

1. *Faites quelques expérimentations avec la notion de pointeur en OCaml, par exemple en définissant des pointeurs sur des entiers.*
2. *Pour définir un arbre binaire avec des pointeurs, on définit un noeud d'un arbre comme un type produit qui contient la valeur à la racine et deux pointeurs vers des noeuds pour définir les fils gauche et droit.*

   Pour implanter cela, définissez simultanément un type produit `node` *qui représente un noeud d'un arbre binaire avec des champs pour contenir la valeur de la racine et deux valeurs de de type* `t_btree` *et un type* `t_btree` *qui est un pointeur sur un noeud.*

3. *Utilisez les types que vous avez défini pour écrire un module qui implante la signature des arbres binaires.*
4. *Faites comme à l'exercice 2, remplacez le fichier* `"btree.cmo"` *par votre module dans du code que vous avez écrit et vérifiez qu'il continue de fonctionner.*

▸ **Exercice 4.** *Transposez en C votre module implantant les arbres binaires avec des pointeurs.*
— *Assurez de respecter l'encapsulation nécessaire à un type abstrait (la représentation concrète du type des arbres doit être cachée à l'utilisateur de votre module).*
— *Veillez à mettre en place ce qu'il faut pour bien gérer la mémoire.*