

Algorithmique 1

TD 5 Arbres binaires : implémentation statique

Soit le type abstrait `arbreBinaire` de `type_predefini` représenté comme il suit :

```
type arbreBinaire= structure
    tailleStock: entier
    arbre: tableau[1..tailleStock] d'elements de ^type_predefini
finstructure
type sommet= entier
type type_predefini= entier
```

On souhaite implémenter le type abstrait `arbreBinaire` de `type_predefini` en utilisant le parcours hiérarchique (voir la fonction `remplirTableauArbre` vue en cours). L'idée de base est de stocker la racine de l'arbre `A` dans le premier élément du tableau `A.arbre[1]` et ensuite, pour chaque élément `A.arbre[i]`, stocker les fils gauche et droit resp. dans `A.arbre[2i]` et `A.arbre[2i+1]`.

```
arbreBinaire= structure
    tailleStock: entier
    arbre: tableau[1..tailleStock] d'elements de ^type_predefini
finstructure
type sommet= entier
type type_predefini= entier
```

On dispose des primitives suivantes :

```
fonction valeurSommet(val arbreBinaire A, val sommet s): type_predefini;
fonction valeur(val A: arbreBinaire, val s: sommet): ^type_predefini;
fonction sommet racine(val A: arbreBinaire) : sommet;
fonction filsGauche(val A: arbreBinaire, val s: sommet) : sommet;
fonction filsDroit(val A: arbreBinaire, val s: sommet) : sommet;
fonction pere(val A: arbreBinaire, val s: sommet) : sommet;
fonction CreerArbreBinaire(ref A: arbreBinaire, val racine: type_predefini) : vide;
fonction ajouterFilsGauche(val x: type_predefini, ref A: arbreBinaire,
    ref s: sommet) : vide;
fonction ajouterFilsDroit(val x: type_predefini, ref A: arbreBinaire,
    ref s: sommet) : vide;
fonction supprimerFilsGauche(ref A: arbreBinaire, ref s: sommet) : vide;
fonction supprimerFilsDroit(ref A: arbreBinaire, ref s: sommet) : vide;
```

Exercice 5.1 Structures de données

L'implémentation de l'arbre de la Fig.1(a) est illustrée sur la Fig.1(b). On ajoute une feuille au dernier niveau. Dessiner la nouvelle structure.

Exercice 5.2 Construction

Dessiner l'arbre binaire construit par la fonction suivante :

```

fonction creer_arbre_fig_1(ref a: arbreBinaire): vide
  var s, s2, s3, s4, s5, s7: sommet;
  debut
    CreerArbreBinaire(a, 1);
    sommet s= racine(a);
    ajouterFilsGauche(2, a, s);
    sommet s2= filsGauche(a, s);
    ajouterFilsGauche(4, a, s2);
    sommet s4= filsGauche(a, s2);
    ajouterFilsDroit(6, a, s4);
    ajouterFilsDroit(3, a, s);
    sommet s3= filsDroit(a, s);
    ajouterFilsDroit(5, a, s3);
    sommet s5= filsDroit(a, s3);
    ajouterFilsGauche(7, a, s5);
    sommet s7= filsGauche(a, s5);
    ajouterFilsDroit(9, a, s7);
    ajouterFilsDroit(8, a, s5);
  fin
finfonction

```

Quelle est la valeur minimale de `tailleStock` pour pouvoir contenir cet arbre?

Exercice 5.3 *Primitives*

Compléter l'implémentation vue en cours par les primitives :

`valeur`, `filsDroit`, `pere`, `ajouterFilsDroit`, `supprimerFilsDroit`

Exercice 5.4 *Exemples*

Écrire les fonctions :

```

fonction est_feuille(val A: arbreBinaire, val s: sommet): boolean;
fonction hauteur(val A: arbreBinaire): vide;
fonction afficher_arbre_parcours_prefixe(val A: arbreBinaire): vide;

```

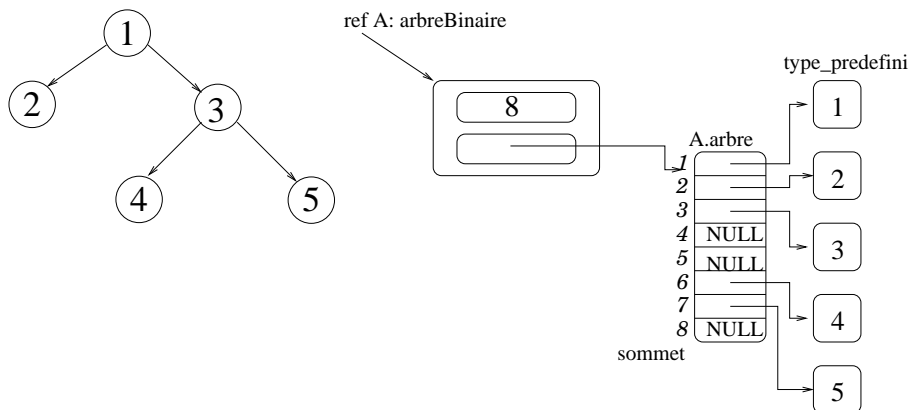


FIG. 1 – Arbre binaire parfait