

Algorithmique 1

Feuille 5 : Arbres

1 Arbres binaires

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

```
cellule=structure
    info:objet;
    gauche:^cellule;
    droit:^cellule;
    pere:^cellule;
finstructure
sommets:^cellule;
arbreBinaire= sommets;
```

On dispose des primitives suivantes :

```
fonction creerArbreBinaire(val Racine:objet):sommets;

fonction getValeur(val S:sommets):objet;
fonction filsGauche(val S:sommets):sommets;
fonction filsDroit(val S:sommets):sommets;
fonction pere(val S:sommets):sommets;

fonction setValeur(ref S:sommets, val x:objet):vide;
fonction ajouterFilsGauche(ref S:sommets, val x:objet):vide;
fonction ajouterFilsDroit(ref S:sommets, x:objet):vide;
fonction supprimerFilsGauche(ref S:sommets):vide;
fonction supprimerFilsDroit(ref S:sommets):vide;
fonction detruireSommets(ref S:sommets):vide;
```

Exercice 5.1 Structures de données

Soit l'arbre de la Fig.1(a). Son implémentation dynamique 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 `creer_arbre_non_quasi_parfait`. Illustrer sa représentation dynamique construite par la fonction suivante :

```
fonction creer_arbre_non_quasi_parfait(ref a: arbreBinaire):vide;
var a, s1, s2, s3, s4, s6: sommets;
debut
    a= creerArbreBinaire(1);
    sommets s1= a;
    ajouterFilsGauche(s1,2);
    sommets s2= filsGauche(s1);
```

```

ajouterFilsDroit(s1,3);
sommet s3= filsDroit(s1);
ajouterFilsGauche(s2,4);
sommet s4= filsGauche(s2);
ajouterFilsDroit(s2,5);
ajouterFilsGauche(s3,6);
sommet s6= filsGauche(s3);
ajouterFilsDroit(s3,7);
ajouterFilsGauche(s4,8);
ajouterFilsDroit(s4,9);
ajouterFilsGauche(s6,6);
fin
finfonction

```

Exercice 5.3 Primitives

Compléter l'implémentation vue en cours par les primitives :

filsDroit, pere, ajouterFilsDroit, supprimerFilsDroit

Exercice 5.4 Exemples

Écrire les fonctions :

```

fonction detruireArbreBinaire(ref A:arbreBinaire):vide;
fonction est_noeud_interne(val s: sommet): boolean;
fonction hauteur(val A: arbreBinaire): entier;
fonction afficher_arbre_parcours_prefixe(val A: arbreBinaire): vide;
fonction longueur_de_cheminement(val A:arbreBinaire):entier;
/* calcule la somme des hauteurs des feuilles*/

```

Exercice 5.5 Même squelette

Écrire une fonction `meme_squelette` qui teste si deux arbres binaires sont semblables à la valeur près des valeurs contenues dans les sommets.

Exercice 5.6 Arbre binaire de Calder

Écrire une fonction `poids` qui calcule la somme des valeurs contenues dans les sommets d'un arbre binaire. En déduire un prédicat `est_Calder` qui teste si, en tous les noeuds, les deux fils ont le même poids.

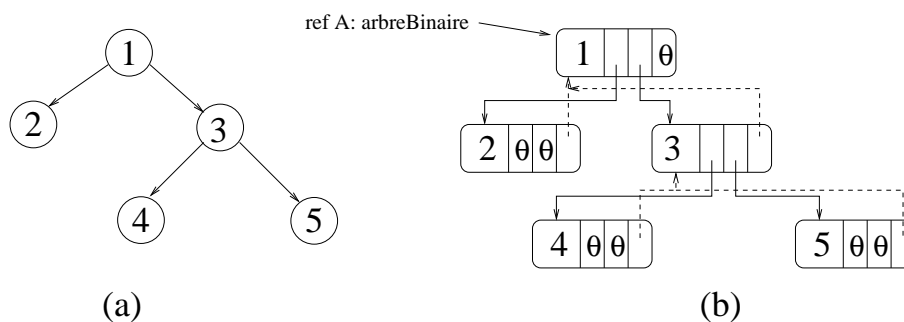


FIG. 1 – Arbre binaire complet

2 Arbres planaires

Soit le type abstrait `arbrePlanaire` représenté comme il suit :

```

cellule=structure
    info:objet;
    premierFils:^cellule;
    frere:^cellule;
    pere:^cellule;
finstructure
sometArbrePlanaire=^cellule;
arbrePlanaire= sometArbrePlanaire;

```

On dispose des primitives suivantes :

```

fonction creerArborescence(val Racine:objet):arbrePlanaire;

fonction getValeur(val S:sometArbrePlanaire):objet;
fonction premierFils(val S:sometArbrePlanaire):sometArbrePlanaire;
fonction frere(val S:sometArbrePlanaire):sometArbrePlanaire;
fonction pere(val S:sometArbrePlanaire):sometArbrePlanaire;

fonction setValeur(ref S:sometArbrePlanaire, val x:objet):vide;
fonction ajouterFils(ref S:sometArbrePlanaire, val x:objet):vide;
fonction ajouterFrere(ref S:sometArbrePlanaire, x:objet):vide;
fonction supprimerSometPlanaire(ref S:sometArbrePlanaire):vide;

```

Exercice 5.7 Construction

Écrire une fonction qui construit la représentation illustrée sur la figure 2(b) en utilisant l'arbre planaire de la figure 2(a).

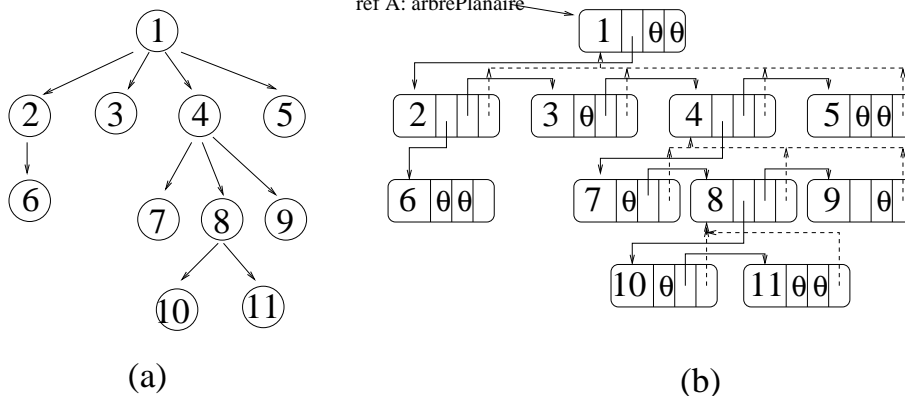


FIG. 2 – Arbre planaire

Exercice 5.8

Démontrer que les primitives du type `arbreBinaire` sont réalisables par les primitives du type `arbrePlanaire` en temps constant et vice versa.

Exercice 5.9 Gestion d'une piste d'atterrissage des avions

Un avion est un enregistrement contenant :

- l'indicatif (6 caractères)
- la destination (30 caractères)
- l'autonomie résiduelle de carburant comptée en heures de vol (entier)
- deux booléens indiquant s'il y a un pirate à bord et s'il y a le feu.

1. Définir les structures de données nécessaires.
2. Ecrire la fonction `Priorité` ainsi que la gestion complète de la piste.
3. Envisager le cas de suppression d'un élément quelconque de la file lorsque le pirate a mis sa menace de détournement à exécution.

Quelles sont les notions que vous venez de voir qui peuvent permettre d'amorcer le fil d'ariane ?