Université Bordeaux 1 Licence Informatique 2011/2012

Algorithmique 1

DS 3

Documents **non** autorisés

Arbres Binaire, Planaire et ABR

Exercice 3.1

1. Ecrire une fonction qui affiche les valeurs de sommets d'un arbre planaire selon le parcours postfixe en utisant une pile.
2. Quelle est sa complexité
3. Donner le principe pour écrire la même fonction sans utiliser une structure supplémentaire.

Exercice 3.2

1. Construire un arbre binaire de recherche avec les valeurs successives suivantes :

$$17, 25, 13, 6, 35, 1, 15, 13, 3, 25, 30, 28, 26$$

2. Faire la trace de l'exécution de la fonction `mystere` sur l'ABR construit ci-dessus,
3. Que fait la fonction `mystere` en générale ?

```
fonction mystere(ref A: arbreBR) : vide
var s,p: sommet;
    b: booleen;
debut
     Si A != NIL alors
        s= A; b= vrai;
        tant que (b) faire
           si filsDroit(s)
               alors s=filsDroit(s)
               sinon b = faux;
           finsi
         fin tq
         si s==A
                alors
                     A= filsGauche(s);
                     A^.pere=NIL;
```

```
                    detruire(s);
                sinon
                    p= filsGauche(s)
                    p^.pere= pere(s);
                    detruire(s);
                    s=pere(p);
                    s^.filsDroit=p;
                finsi
    finsi
fin
```

Exercice 3.3

Ecrire une fonction qui renvoie le sommet qui contient la première occurrence d'un objet dans un ABR au cas où l'objet n'est pas trouvé renvoie `NIL`.

Annexe A **Type abstrait** *sommetArbrePlanaire*

```
sommetArbrePlanaire= curseur;
```

- **Création**
  `fonction creerArborescence(val Racine:objet):sommetArbrePlanaire;`
- **Accès**
  `fonction valeur(val S:sommetArbreBinaire):objet;`
  `fonction premierFils(val S:sommetArbreBinaire):sommetArbreBinaire;`
  `fonction frere(val S:sommetArbreBinaire):sommetArbreBinaire;`
  `fonction pere(val S:sommetArbreBinaire):sommetArbreBinaire;`
- **Modification**
  `fonction setValeur(ref S:sommetArbrePlanaire, val x:objet):vide;`
  `fonction ajouterFils(ref S:sommetArbrePlanaire, val x:objet):vide;`
  `fonction supprimerSommet(ref S:sommetArbrePlanaire):vide;`

Annexe B **Implémentation du type abstrait** *sommetArbrePlanaire*

```
cellule= structure
          info: objet;
          premierFils: sommet
          frere: sommet;
          pere: sommet
         finstructure
sommet= ^cellule;
sommetArbrePlanaire= sommet;
```

Annexe C **Type abstrait** *ArbreBinaire*

```
arbreBinaire= curseur;
sommet= curseur;
```

- **Création**
  `fonction creerArbreBinaire(val Racine:objet):sommet;`
- **Accès**
  `fonction getValeur(val S:sommet):objet;`
  `fonction filsGauche(val S:sommet):sommet;`
  `fonction filsDroit(val S:sommet):sommet;`
  `fonction pere(val S:sommet):sommet;`
- **Modification**
  `fonction setValeur(ref S:sommet, val x:objet):vide;`
  `fonction ajouterFilsGauche(ref S:sommet, val x:objet):vide;`
  `fonction ajouterFilsDroit(ref S:sommet, x:objet):vide;`
  `fonction supprimerFilsGauche(ref S:sommet):vide;`
  `fonction supprimerFilsDroit(ref S:sommet):vide;`
  `fonction detruireSommet(ref S:sommet):vide;`

Annexe D **Implémentation du type abstrait** *arbreBinaire*

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