Université Bordeaux 1 LST Mention Informatique Semestre 3 2013-2014

Algorithmique 1 : Devoir Surveillé 2

Piles. Files. Arbre Binaires. Arbres Planaires

Durée : 60mn
Sans documents

Le barême est donné à titre indicatif.

Exercice 2.1 *Piles (4 points)*

On souhaite construire un mini éditeur de ligne. On donne une chaîne contenant les commandes d' édition. Il s'agit d'afficher la suite de caractères résultant de l'exécution de ces commandes. Ces caratères sont stockés dans un tampon. Chaque caractère est une commande ; on distingue quatre catégories de caractères :

- tout caractère différent des trois caractères `#` , `$` et `\newline` est un caractère du texte, on le stocke dans le tampon ;
- le caractère `#` signifie supprimer le caractère précédent dans le tampon ;
- le caractère `$` signifie supprimer tous les caractères du tampon ;
- le caractère `\newline` signifie la fin de chaîne de commande et déclenche l'affichage du tampon constitué par les caractères du texte.

Ainsi par exemple, la chaîne `"ABC$abce#de#\newline"` sera transformée en `"abcd"`.
Pour réaliser ces opérations, on représentera le tampon par une pile de caractères.

Exercice 2.2 *Files (4 points)*

Ecrire une fonction `separe` qui à partir d'un tableau d'entiers $Tab[1..n]$ *de entier* , fournit deux files, F_1 et F_2. La file F_1 doit contenir les nombres entiers appartenant au tableau Tab et dont le dernier chiffre est un nombre pair.
La file F_2 doit contenir tous les autres nombres entiers appartenant au tableau Tab mais non inclus dans F_1.
L'ordre des entiers dans les files sera respecté.
Par exemple, soit le tableau $Tab = \{12, 1, 3, 8, 144, 107, 91\}$. Le contenu de la file F_1 sera $[12, 8, 144[$ et le contenu de la file F_2 sera $[1, 3, 107, 91[$.

Exercice 2.3 *Arbres (12 points)*

a. Peut-on créer un arbre binaire dont la valeur du sommet est une liste doublement chaînée `listeDC`.

b. Doit-on utiliser une file pour parcourir un arbre binaire en ordre infixe ? Si oui, pourquoi ? Si non, quelle structure de données est utilisée ?

c. Quels sont les parcours qui existent pour les arbres binaires et pour les arbres planaires ?

d. Soit l'arbre illustré sur la Fig. 1. Donner sa hauteur.

e. Ecrire une fonction qui calcule la hauteur d'un arbre binaire.

f. On considère que cet arbre binaire représente un arbre planaire en prenant comme lien gauche de tout noeud le lien vers le premier fils, et comme lien droit le lien vers son frère de droite. Donner l'arbre planaire correspondant.

g. Ecrire une fonction qui calcule le nombre de feuilles dans un arbre planaire.

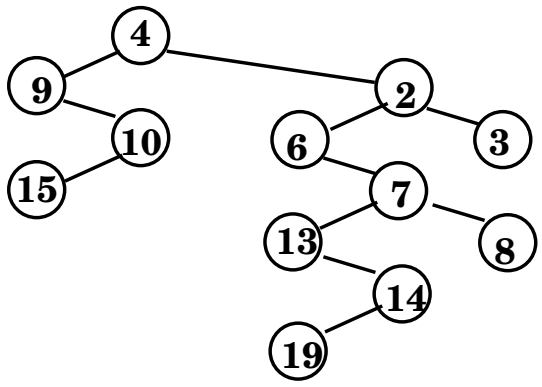

FIGURE 1 – Arbre binaire

ANNEXE A **Type abstrait** *pile de objet*

```
fonction valeur(val P:pile de objet):objet;
fonction pileVide(val P:pile de objet):booleen;
fonction creerPile(ref P:pile de objet): vide;
fonction empiler(ref P:pile de objet, val x:objet):vide;
fonction depiler (ref P:pile de objet):vide;
fonction detruirePile(ref P:pile de objet):vide;
```

ANNEXE B **Type abstrait** *file de objet*

```
fonction valeur(val F:file de objet):objet;
fonction fileVide(val F:file de objet):booleen;
fonction creerFile(ref F:file de objet): vide;
fonction enfiler(ref F:file de objet, val v:objet):vide;
fonction defiler(ref F:file de objet):vide;
fonction detruireFile(ref F:file de objet):vide;
```

ANNEXE C **Type abstrait** *arbreBinaire*

```
arbreBinaire= curseur;
sommet= curseur;
fonction creerArbreBinaire(val Racine:objet):sommet;
fonction detruireArbreBinaire(ref S:sommet):vide;
fonction getValeur(val S:sommet):objet;
fonction filsGauche(val S:sommet):sommet;
fonction filsDroit(val S:sommet):sommet;
fonction pere(val S:sommet):sommet;
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;
arbreBinaire= sommet;
```

Annexe E **Type abstrait** *sommetArbrePlanaire*

```
sommetArbrePlanaire= curseur;
fonction creerArbrePlanaire(val Racine:objet):sommetArbrePlanaire;
fonction detruireArbrePlanaire(ref S:sommetArbrePlanaire):vide;
fonction getValeur(val S:sommetArbreBinaire):objet;
fonction premierFils(val S:sommetArbreBinaire):sommetArbreBinaire;
fonction frere(val S:sommetArbreBinaire):sommetArbreBinaire;
fonction pere(val S:sommetArbreBinaire):sommetArbreBinaire;
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 F **Implémentation du type abstrait** *sommetArbrePlanaire* **par fils gauche-frère droit**

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