Université Bordeaux 1 LST Mention Informatique Semestre 3 2009-2010

Algorithmique 1 : Devoir Surveillé 2

Pile et File

Durée : 40mn
Sans documents

Dans les exercices suivants on considère les types abstraits `pile de objet` et `file de objet` définis en cours[1].

Exercice 2.1 *Mystere*

1. Soit une fonction `construire` qui ajoute tous les éléments d'une liste `L` du type `listeSC` d'entiers dans une pile, puis construit une liste en vidant la pile. Quelle est le contenu de la liste renvoyée ? Illustrer sur l'exemple `L={11,2,31,405}`.
2. Même question si on utilise une file ?

```
fonction construire(val Lsrc: listeSC) : listeSC;
var P: pile de entiers;
var Ldst: listeSC;
debut
   creerPile(P);
   debutListe(Lsrc);
   tant que valeur(Lsrc) != NIL faire
      empiler(P, valeur(Lsrc));
      suivant(Lsrc);
   fintq
   creerListe(Ldst);
   tant que !pileVide(P) faire
      insererEnTete(Ldst, valeur(P));
      depiler(P);
   ftq
   retourner Ldst;
fin
```

Exercice 2.2 *Evolution*

1. Soit la pile d'entiers `P` illustrée sur Fig. 1(a). En s'inspirant de cette figure, dessiner son état après la suite d'opérations :
   `empiler(P,4);empiler(P,17);empiler(P,3);empiler(P,5);`
   Donner la représentation de la pile d'entiers `P` (structures de données,contenus) correspondant à cette implémentation.
2. Soit la file d'entiers `F` illustrée sur Fig. 1(b). Mêmes questions pour la suite d'opérations :
   `enfiler(F,4);enfiler(F,17);enfiler(F,3);enfiler(F,5);`

[1] Pour rappel voir l'annexe A

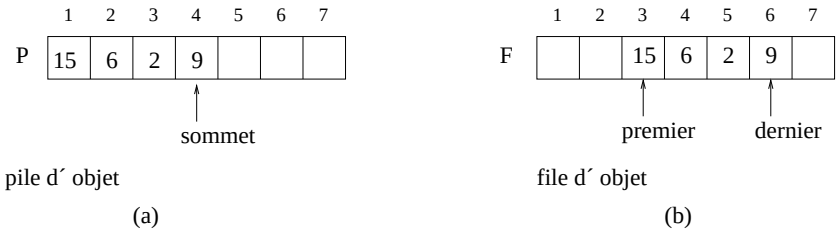

FIG. 1 – Implémentation dans un tableau du type (a) Pile d'entiers. (b) File d'entiers.

Exercice 2.3 *Implémentation*

1. Expliquer comment implémenter deux piles dans un tableau `A[1...n]` de telle manière qu'aucune ne déborde à moins que le nombre total d'éléments des deux piles vaille `n`. Les opérations `empiler` et `depiler` devront s'exécuter dans un temps en `O(1)`
2. Montrer comment implémenter une file à l'aide de deux piles. Analyser le temps d'exécution des opérations de file.

Exercice 2.4 *Utilisation*

On se donne deux piles P_1 et P_2, et une file F. La pile P_1 contient une suite d'entiers. Ecrire une fonction qui met dans P_2 les entiers pairs de P_1 et laisse dans P_1 les entiers impairs. Attention, les entiers doivent être dans le même ordre qu'au début.
Par exemple, si l'affichage de P_1 donnait la suite $1, 2, 3, 4, 5, 6$, l'affichage de P_1 doit donner $1, 3, 5$ et celui de P_2 doit donner $2, 4, 6$.

Annexe A : rappels du cours

Primitives du 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;
```

Implémentation du type abstrait pile de objet par un tableau

```
pile de objet= structure
                 taille: entier;
                 sommet: entier;
                 pile: tableau[1..taille] de objet
               finstructure
```

Implémentation du type abstrait pile de objet par une liste listeSC

```
pile de objet= listeSC de objet
```

Primitives du 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;
```

Implémentation du type abstrait file de objet par un tableau
On utilise le tableau de manière circulaire avec un pointeur donnant le premier élément et un autre donnant le dernier.

```
file de objet= structure
                 taille: entier;
                 premier: entier;
                 dernier: entier;
                 plein: booleen;
                 file: tableau [1..taille] de objet
               finstructure
```

Implémentation du type abstrait file de objet par une liste listeDC

```
file de objet= listeDC de objet
```