Université Bordeaux 1 LST Mention Informatique Semestre 3 2008-2009

Algorithmique 1 : Devoir Surveillé 2

Pile et File

Durée : 40mn
Sans documents

Exercice 2.1 *Évolution d'une pile*
Montrer l'évolution d'une pile au cours d'une série d'empilements et de dépilements provoqués par la suite :

```
Q*UES***TI*ON*FAC***IL***E**
```

Chaque lettre provoque un empilement et chaque astérix un dépilement.

Exercice 2.2 *Implémentation du type pile*
Dans cet exercice on considère l'implémentation du type abstrait `pile` par un tableau

```
type objetPile= car
type pile d'objetPile= structure
                         taille: entier;
                         sommet: entier;
                         pile: tableau[1..taille] d'objetPile;
                       finstructure
```

Donner l'implémentation des primitives du type `pile`.

```
fonction valeurPile(ref P: pile d'objetPile): objetPile;
fonction pileVide(ref P: pile d'objetPile): bool;
fonction creerPile(ref P: pile d'objetPile): vide;
fonction empiler(ref P: pile d'objetPile, val x: ): vide;
fonction depiler(ref P: pile d'objetPile): vide;
fonction detruirePile(ref P: pile d'objetPile): vide;
```

Exercice 2.3 *Mystere*
Soit la fonction `mystere` donnée en annexe A. Quel est le résultat si on l'applique sur un tableau T d'entiers `T[]={ 10, 2, 45, 88, 9, 65, 76, 3, 100 }`. Justifier votre réponse.
La fonction `partitionnement`

```
fonction partitionnement(ref T:tableau d'entiers, var g:entier, var d:entier): entier;
```

renvoie un indice `i` tel que tous les éléments $T[g], \ldots, T[i-1]$ sont inférieurs à $T[i]$ et tous les éléments $T[i+1], \ldots, T[d]$ sont supérieurs à $T[i]$ en réorganisant au besoin le sous-tableau $T[g], \ldots, T[d]$.

Exercice 2.4 *Évolution d'une file*
Montrer l'évolution d'une file au cours d'une série d'enfilements et de défilements provoqués par la suite :

```
Q*UES***TI*ON*FAC***IL***E**
```

Chaque lettre provoque un enfilement et chaque astérix un défilement.

Exercice 2.5 *Implémentation du type file*
Dans cet exercice on considère l'implémentation du type abstrait `file` par une liste du type `listeDC_car`, liste doublement chaînée implémentée par l'allocation dynamique vue en cours. Le rappel sur la définition et l'ensemble de primitives de `listeDC_car` est donné sur table 1.

1. Écrire l'implémentation des primitives du type `file`

```
type objetFile= car
type file d'objetFile = listeDC_car;
fonction valeurFile(ref F: file d'objetFile): objetFile:
fonction fileVide(ref F: file d'objetFile): bool
fonction creerFile(ref F: file d'objetFile): vide;
fonction enfiler(ref F: file d'objetFile, var x: objetFile): vide;
fonction defiler(ref F: file d'objetFile): vide;
fonction detruireFile(ref F: file d'objetFile): vide;
```

Exercice 2.6
Est-il possible d'implémenter une file avec deux piles ?
Justifier votre réponse.

Annexe A

```
fonction mystere(ref T: tableau[1..taille] d'entiers): vide
var i, g, d: entier;
var P: pile d'entiers;
debut
  g= 1;
  d= taille;
  creerPile(P);
  tant que (Vrai) faire
    tant que(d > g)
      i= partitionnement(T, g, d);
      si(i - g > d - i)alors
        empiler(P, g);
        empiler(P, i - 1);
        g= i + 1;
      sinon
        empiler(P, i + 1);
        empiler(P, d);
        d= i - 1;
      fsi
    ftq
    si( pileVide(p) == Vrai ) alors
      break;
    sinon
      d= valeurPile(P);
      depiler(P);
      g= valeurPile(P);
      depiler(P);
    ftq
fin
```

cellule= structure valeurElement : car ; pointeurSuivant : ^cellule ; pointeurPrécédent : ^cellule ; finstructure		
listeDC_car= structure premier : curseur ; dernier : curseur ; clé : curseur ; finstructure	gestion	fonction new(ref c : ^ cellule) : vide ; fonction delete(ref c : ^ cellule) : vide ;
	accès	fonction valeur(val L : listeDC_car) : objet ; fonction debutListe(ref L : listeDC_car) : vide ; fonction finListe(ref L : listeDC_car) : vide ; fonction suivant(ref L : listeDC_car) : vide ; fonction précédent(ref L : listeDC_car) : vide ; fonction listeVide(val L : listeDC_car) : booléen ; fonction getCléListe(val L : listeDC_car) : curseur ;
	modification	fonction créerListe(ref L : listeDC_car) : vide ; fonction insérerAprès(ref L : listeDC_car, val x : objet) : vide ; fonction insérerEnTête(ref L : listeDC_car, val x : objet) : vide ; fonction supprimerAprès(ref L : listeDC_car) : vide ; fonction supprimerEnTête(ref L : listeDC_car) : vide ; fonction setCléListe(ref L : listeDC_car, val c : curseur) : vide ; fonction détruireListe(ref L : listeDC_car) : vide ;

TAB. 1 – Implémentation du type listeDC par allocation dynamique.

Annexe B