Université Mohammed V-Agdal
Faculté des Sciences de Rabat
Département d'Informatique

Année Universitaire 2013/2014
MODULE M14 (SMI-S4)

Travaux Dirigés de Structures de Données
[TD n°2 : "Diviser pour régner" & Types Abstraits de Données]

Objectifs : *- Initier à la conception d'algorithmes utilisant la stratégie "Diviser pour régner" ;*
- Initier à la manipulation des types abstraits de données (TADs).

Exercice 1

En utilisant **la stratégie "*Diviser pour régner*" :**

a)- Ecrire un algorithme (*ou une fonction C*), **`puissance`**, pour calculer la puissance n-ième d'un nombre entier a.

b)- Ecrire un algorithme (*ou une fonction C*), **`min_tab`**, pour déterminer le plus petit entier d'un tableau contenant n nombres entiers.

Exercice 2

a)- Appliquer l'algorithme du tri rapide sur l'exemple suivant : 12, 3, 9, 2, 6, 8, 5, 13, 11, 7, 6, 18.

b)- L'algorithme du tri rapide est il stable ? Expliquer.

Exercice 3

On désire implanter, en langage C, **le type abstrait de données Vecteur**, défini dans le cours. On rappelle, ci-dessous, une spécification minimale du **TAD Vecteur :**

```
Type Vecteur
Utilise Entier, Elément
Opérations
     vect         : Entier → Vecteur
     changer_ième: Vecteur x Entier x Elément → Vecteur
     ième         : Vecteur x Entier → Elément
     taille       : Vecteur → Entier
Préconditions
     vect(i) est_défini_ssi i ≥ 0
     ième(v,i) est_défini_ssi 0 ≤ i < taille(v)
     changer_ième(v,i,e) est_défini_ssi 0 ≤ i < taille(v)
Axiomes
     Soit, i, j : Entier, e : Elément, v : Vecteur
     si 0 ≤ i < taille(v) alors ième(changer_ième(v,i,e),i) = e
     si 0 ≤ i < taille(v) et 0 ≤ j < taille(v) et i ≠ j
           alors ième(changer_ième(v,i,e),j) = ième(v,j)
     taille(vect(i)) = i
     taille(changer_ième(v,i,e)) = taille(v)
```

Il existe plusieurs méthodes pour représenter les vecteurs. Parmi celles-ci une méthode qui consiste à représenter un vecteur sous la forme d'un tableau. Plus précisément, un vecteur est une structure regroupant un entier, pour la taille, et un pointeur sur les éléments du vecteur.

a)- Donner la déclaration complète, en C, de la structure de données représentant le TAD Vecteur d'entiers, c-à-d :

- Donner les déclarations nécessaires pour définir **le type Vecteur d'entiers** ;
- Donner les déclarations, en C, **des primitives** du TAD Vecteur d'entiers.

b)- Définir les primitives (*ou opérations de base*) du TAD Vecteur d'entiers, c'est à dire : pour chaque primitive, écrire la fonction C correspondante.

c)- ***Question facultative :*** Ajouter au TAD Vecteur d'entiers les deux opérations auxiliaires suivantes:

- **`produit_scalaire`** qui calcule le produit scalaire de deux vecteurs d'entiers.
- **`decalage_circulaire`** qui prend un vecteur et retourne le vecteur dans lequel tous les éléments sont décalés d'une place vers la droite (*le dernier est inséré à la 1ère place*)

Exercice 4

On s'intéresse au **TAD Pile**, défini dans le cours. Les primitives du TAD Pile sont supposées déjà définies.

a)- Ecrire un algorithme (*ou une fonction C*), **`taille_pile`**, qui calcule le nombre d'éléments dans une pile donnée.

b)- Ecrire un algorithme (*ou une fonction C*), **`copie_pile`**, qui retourne une copie d'une pile donnée.

Exercice 5

On utilise le **TAD Pile**, défini dans le cours.

a)- Ecrire un algorithme (*ou une fonction C*), **`tri_pile`**, de tri par insertion d'un ensemble de nombres entiers. Les données sont stockées dans une pile P1 et non un tableau. L'algorithme (ou la fonction) doit retourner une pile P2 contenant ces nombres triés, avec le minimum au sommet de la pile. (***indication :*** *utiliser trois piles, P1, P2 et une 3ème pile d'aide P3*).

b)- Tester avec la pile P1 = {4, 3, 2, 5, 8, 2, 6, 9, 3}, où l'entier 3 le plus à droite est l'élément au sommet de P1.

c)- En déduire un algorithme (*ou une fonction C*), **`tri_tab`**, pour trier un tableau contenant n nombres entiers.