TP 4 : Boucles et tableaux

Informatique Fondamentale (IF1)

20 octobre 2005

Remarques : En TP, lorsqu'un exercice demande d'« écrire une méthode », il faudra aussi la tester en l'incluant dans un programme complet qui l'appelle avec des arguments appropriés.

Ce TP contient un grand nombre d'exercices et pour certains il sera difficile de tous les faire. Il est donc important pour tous d'avoir abordé des exercices des deux premières sections. N'hésitez pas à sauter certains exercices si le temps venait à manquer.

Rappel : Sauf si l'énoncé le précise explicitement, une méthode n'est pas censée lire ou afficher quoi que ce soit. Elle doit *renvoyer* un résultat.

1 Suites numériques **(ne pas utiliser de tableau si ce n'est pas nécessaire, mais il faut ulitiser les boucles !)**

Exercice 1 : *Puissance d'un nombre*

Écrire une méthode qui calcule la puissance $n^{\text{ème}}$ d'un nombre, ou n est un entier positif. Par exemple, `puissance(2.5, 3)` vaut $15,625$.

Modifier la méthode précédente pour traiter le cas où n est un entier quelconque. Par exemple, `puissance(2.5, -3)` vaut $0,064$; `puissance(2.5, 0)` vaut 1.

Exercice 2 : *Factorielle*

On rappelle que la fonction factorielle est définie sur les entiers positifs de la façon suivante :

$$\begin{aligned} &\text{factorielle}(0) = 1 \\ &\text{factorielle}(n) = n \times \text{factorielle}(n-1) \qquad\qquad \text{si } n \geqslant 1 \end{aligned}$$

Écrire une méthode qui calcule la factorielle d'un entier.

Exercice 3 : *Une suite*

On définit la suite $(x_n)_{n\in\mathbb{N}}$ de la manière suivante : $x_0 = a$ et $x_{n+1} = 4x_n(1 - x_n)$. Écrire une méthode qui calcule x_n en fonction de a et de n. Modifier celle-ci pour qu'elle affiche les valeurs successives $x_0, x_1, \ldots, x_n$ de la suite.

Exercice 4 : *Une autre série*

Écrire une méthode qui calcule la somme des carrés des n premiers entiers *impairs*. Par exemple, `sommeCarresImpairs`$(5) = 1^2 + 3^2 + 5^2 + 7^2 + 9^2 = 165$.

Exercice 5 : *Deux suites*

On définit une suite double :

$$\begin{aligned} u_0 &= 1 & u_{n+1} &= (u_n + v_n)/2 \\ v_0 &= 2 & v_{n+1} &= \sqrt{u_{n+1} v_n} \end{aligned}$$

On admet que les suites (u_n) et (v_n) sont adjacentes de limite $\sqrt{27}/\pi$. Écrire un programme qui lit un entier n et affiche l'approximation du nombre π obtenue à partir de v_n.

Rappel : la méthode `Math.sqrt` permet de calculer la racine carrée d'un nombre.

Exercice 6 : *Suite de Fibonacci*

On définit la suite de Fibonacci de la manière suivante :

$$F_0 = F_1 = 1$$
$$F_{n+2} = F_n + F_{n+1}$$

Écrire une méthode qui calcule le $n^{\text{ème}}$ nombre de Fibonacci (c'est-à-dire F_n).

2 Tableaux de nombres

Exercice 7 : *Lecture d'un tableau de nombres*

- Écrire une méthode `int[] litTableau()` qui lit un entier n, puis n nombres réels qu'elle place dans un tableau, et qui renvoie ce tableau.
- Changer la méthode `litTableau` pour que elle prend le tableau comme un parametre et elle le change : `void litTableau1(int[] tableau)`
- Pourquoi ça marche pour le tableau mais pas pour un "int", comme dans la méthode suivante

```
void litEntier(int n){
 n = Deug.readInt();
}
```

Exercice 8 : *Norme, produit scalaire*

Il est naturel de représenter le vecteur de coordonnées $(x_0, ..., x_{n-1})$ (dans un espace vectoriel de dimension n) par le tableau dont ces n nombres sont les éléments.

(a) Écrire une méthode `norme` qui calcule la norme d'un vecteur. On rappelle que la norme du vecteur $\vec{x}$ de coordonnées $(x_0, ..., x_{n-1})$ est donnée par

$$\|\vec{x}\| = \sqrt{\sum_{i=0}^{n-1} x_i^2} = \sqrt{x_0^2 + x_1^2 + \cdots + x_{n-1}^2}$$

(b) Écrire une méthode qui calcule le produit scalaire de deux vecteurs, dont on rappelle l'expression :

$$\vec{x} \bullet \vec{y} = \sum_{i=0}^{n-1} x_i y_i$$

Exercice 9 : *Norme du sup*

On appelle « norme du sup » du vecteur $\vec{x}$ de coordonnées $(x_0, ..., x_{n-1})$ le nombre réel positif donné par

$$\|\vec{x}\|_\infty = \max\{|x_0|, |x_1|, \ldots, |x_n|\}$$

Écrire une méthode qui calcule la norme du sup d'un vecteur.

Exercice 10 : *Concaténation*

Écrire une méthode `concatene` qui prend comme arguments deux tableaux `t1` et `t2` et qui construit et renvoie un tableau `t` contenant les éléments de `t1` suivis des éléments de `t2`.

3 Tableaux triés

Exercice 11 : *Vérification de tri*

Écrire une méthode qui prend comme argument un tableau de nombres et qui vérifie si ce tableau est trié en ordre croissant.

Exercice 12 : *Fusion de deux tableaux triés*

Écrire une méthode qui prend comme arguments deux tableaux de nombres `t1` et `t2`, chacun trié en ordre croissant, et qui construit un nouveau tableau `t` trié contenant les mêmes éléments. Si un nombre apparaît k_1 fois dans `t1` et k_2 fois dans `t2`, il doit apparaître $k_1 + k_2$ fois dans `t`. Par exemple, la fusion de $(3, 3, 6, 7)$ avec $(1, 2, 3, 5, 6)$ est $(1, 2, 3, 3, 3, 5, 6, 6, 7)$.

4 Statistiques

Exercice 13 : *Constitution d'un histogramme*

Un examen est noté sur 10, par points entiers (c'est-à-dire que les notes possibles sont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 et 10). À fins statistiques, on veut savoir, pour chaque note, combien d'élèves l'ont obtenue. On suppose qu'il y a e étudiants, et qu'on dispose d'un tableau `notes` tel que `notes[i]` est la note du $i^{\text{ème}}$ étudiant. Construire un tableau `stats` tel que `stats[n]` contienne le nombre d'étudiants qui ont obtenu la note `n`.

5 Exercices Supplementaires

Exercice 14 : *Methodes Recursive*

Repeter tous les exercices de la section "Suites numériques", mais en remplacant les methodes avec les boucles par les methodes recursives (regarder la fin du TP3)

Exercice 15 : *Tri par la fusion*

Écrire une méthode qui prend comme arguments un tableau de nombres `t1`, et qui construit un nouveau tableau `t` trié contenant les mêmes éléments. L'algorithme est suivante : vous divisez le tableau en parties, et appelez cette methode pour chaque de ces parties. Apres vous pouvez utiliser la fonction de l'exercice 12.