TP 7 : Invariants, tableaux multidimensionnels

Informatique Fondamentale (IF121)

25 novembre 2005

1 Matrices

Les matrices sont des tableaux à deux dimensions de nombres réels, on va donc les représenter par des variables de type *double[][]. Rappelons la syntaxe des manipulations de base sur ces tableaux :*

double[][] a ;	*déclaration d'une matrice*
a = new double[m][n] ;	*création d'une matrice à m lignes et n colonnes*
a[i][j]	*accès à l'élément j de la ligne i*
a.length	*nombre de lignes d'une matrice*
a[0].length	*nombre de colonnes d'une matrice*

Les lignes d'une matrice $m \times n$ sont numérotées de 0 à $m-1$ et ses colonnes sont numérotées de 0 à $n-1$.

Exercice 1 (Entrées et sorties)

1. *Écrire une fonction* litMatrice qui demande à l'utilisateur les dimensions d'une matrice et ses coefficients et renvoie le tableau correspondant.
2. Écrire une fonction *afficheMatrice qui affiche une matrice ligne par ligne. Par exemple, la matrice*

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad s'affiche \quad \begin{matrix} 1.0\ 2.0 \\ 3.0\ 4.0 \end{matrix}$$

Exercice 2 (Addition) *Si A et B dont deux matrices à m lignes et n colonnes, dont les éléments sont a_{ij} et b_{ij}, la matrice $A+B$ est la matrice de même taille dont les éléments sont les $a_{ij}+b_{ij}$. Écrire une fonction qui prend en arguments deux matrices (qu'on supposera de même taille) et renvoie leur somme (sous la forme d'une nouvelle matrice, on ne doit pas modifier les matrices données en arguments).*

Exercice 3 (Produit par un scalaire) *Si $A = (a_{ij})$ est une matrice $m \times n$ et x est un nombre réel, la matrice $x\,A$ est la matrice $m \times n$ dont les éléments sont les $x\,a_{ij}$. Écrire une fonction qui prend en arguments un réel* x et une matrice *a et qui renvoie la matrice produit de* a par *x.*

Exercice 4 (Transposée) *Si $A = (a_{ij})$ est une matrice à m lignes et n colonnes, sa transposée ${}^tA$ est la matrice à n lignes et m colonnes dont les éléments sont les a_{ji}. Écrire une fonction qui calcule la transposée d'une matrice.*

Exercice 5 (Produit) *Si $A = (a_{ij})$ est une matrice à m lignes et n colonnes et $B = (b_{jk})$ est une matrice à n lignes et p colonnes, le produit AB est la matrice à m lignes et p colonnes définie par*

$$AB = \begin{pmatrix} c_{0,0} & \cdots & c_{0,p-1} \\ \vdots & \ddots & \vdots \\ c_{m-1,0} & \cdots & c_{m-1,p-1} \end{pmatrix} \qquad avec \qquad c_{ik} = \sum_{j=0}^{n-1} a_{ij} b_{jk}$$

Écrire une fonction qui prend en arguments deux matrices A et B de tailles supposées compatibles (telles que le nombre de colonnes de A est égal au nombre de lignes de B) et renvoie leur produit.

2 Invariants et boucles while

Exercice 6 (La constante de Néper) *La constante de Néper, notée e, est définie par :*

$$e = \lim_{n\to\infty} \sum_{k=0}^{n} \frac{1}{k!}$$

La de terme suite $u_n = \sum_{k=0}^{n} \frac{1}{k!}$ est une suite croissante qui tend vers e. En calculant la suite u_n pour un entier n assez grand, on obtient une bonne valeur approchée de e. La propriété suivante permet de contrôler l'erreur commise : pour tout réel $\epsilon > 0$ on a $u_{n+1} - u_n < \frac{\epsilon}{3} \Rightarrow e - u_{n+1} < \epsilon$

Écrire une méthode neper qui calcule une valeur approchée de e, elle prend en argument un double ϵ qui correspond à la précision désirée sur le résultat.

La méthode comporte une boucle pour laquelle vous préciserez un invariant, afin de prouver la correction du programme.

Exercice 7 (Recherche dichotomique) *Le but est d'écrire une fonction de recherche d'une valeur dans un tableau d'entiers.*

Écrire une méthode recherche qui prend un tableau d'entiers *t et une valeur entière* v et qui retourne un indice *i tel que* `t[i]=v` *(si la valeur n'est pas présente dans le tableau, la méthode retourne -1).*

Dans le cas où le tableau est trié par ordre croissant, il est possible de retrouver plus rapidement la valeur recherchée v, par *dichotomie.* Le principe est de comparer à `v` la valeur stockée au milieu du tableau : si elle est plus grande que `v`, il faut chercher dans la moitié inférieure du tableau, et sinon dans la moitié supérieure. On recommence alors sur la portion du tableau de taille moitié et ainsi de suite jusqu'à trouver la valeur. Écrire la méthode *rechercheDichotomie basée sur cette approche. Identifier un invariant et démontrer la correction du programme.*

Exercice 8 (Tri par sélection) *Le but de cet exercice est d'écrire une fonction qui trie un tableau de nombres entiers, en prouvant la validité du programme.*

1. *Écrire une fonction qui prend en argument un tableau de nombres entiers et renvoie l'indice du plus petit élément. L'expression* minimum(tableau) doit renvoyer un entier *i tel que* tableau[i] soit le plus petit élément du tableau.
2. La fonction précédente utilise forcément une boucle : identifier un invariant pour démontrer que la fonction est correcte.
3. Modifier la fonction *minimum pour qu'elle prenne en argument l'indice où il faut commencer la recherche. Ainsi* minimum(tableau, début) doit renvoyer un entier *i tel que* tableau[i] est le plus petit élément du tableau d'indice supérieur ou égal à *début. Modifier l'invariant pour vérifier la correction de cette nouvelle fonction.*

Le principe du tri par sélection est le suivant : on commence par rechercher le plus petit élément du tableau, puis on l'échange avec le premier élément du tableau. On cherche ensuite le plus petit élément à partir du deuxième indice, puis on l'échange avec le deuxième élément du tableau, et ainsi de suite jusqu'à avoir tout trié.

4. *À l'aide des fonctions précédentes, écrire une fonction qui effectue un tri par sélection sur un tableau d'entiers. Identifier un invariant pour démontrer la correction de cette fonction.*

Exercice 9 (Algorithme d'Euclide) *On s'intéresse au pgcd d'un couple de polynômes $\neq (0,0)$. Attention, dans le cas des polynômes, il n'y a pas unicité du pgcd. Il y a une infinité de pgcd possibles pour un couple de polynômes, qui diffèrent par un coefficient de proportionnalité réel (non nul). On peut donc déduire l'ensemble des pgcd de deux polynômes à partir d'un pgcd quelconque.*

Le but de l'exercice est de programmer l'algorithme d'Euclide (comme celui de l'exercice ***??****) pour les polynômes. La méthode est essentiellement la même que pour les entiers.*

1. *Quel argument permet de démontrer la terminaison de l'algorithme dans le cas des polynômes ?*
2. *Écrire la méthode* pgcd basée sur cet algorithme (elle est construite autour d'une boucle `while`).
3. Exhiber un invariant pour la boucle `while` permettant de justifier la correction du programme.
4. Rédiger un programme complet qui calcule un pgcd de deux polynômes. On choisit d'afficher le pgcd dont le coefficient dominant est égal à 1.