Tableaux et tris

I3 - Algorithmique et programmation

Nicolas Delestre, Nicolas Malandain

Plan...

Architecture des Systèmes d'Information

Rappels : tableau à une dimension 1 / 4

Objectif

- Stocker à l'aide d'une seule variable un ensemble de valeurs de même type

Déclaration

- **Type** Notes = **Tableau[**1..26**] de Reel**
  - définit un nouveau type appelé Notes, qui est un tableau de 26 réels
- a : Notes
  - déclare une variable de type Notes
- b : **Tableau[**1..26**] de Reel**
  - déclare une variable de type tableau de 26 réels
  - a et b sont de même type
- c : **Tableau[**'a'..'z'**] d'**Entier
  - déclare une variable de type tableau de 26 entiers
  - a et c sont de types différents

Rappels : tableau à une dimension 2 / 4

Utilisation

- Ainsi l'extrait suivant d'algortihme :
  tab : **Tableau[**'a'..'c'**] de Reel**
  tab['a'] ← 2.3
  tab['b'] ← -3.6
  tab['c'] ← 4.2
  ... peut être présentée graphiquement par :

tab :

2.3	-3.6	4.2
'a'	*'b'*	*'c'*

Rappels : tableau à une dimension 3 / 4

Caractéristiques

- Un tableau possède un nombre maximal d'éléments défini lors de l'écriture de l'algorithme (les bornes sont des constantes explicites, par exemple MAX, ou implicites, par exemple 10)
  - ce nombre d'éléments ne peut être fonction d'une variable
- Par défaut si aucune initialisation n'a été effectuée les cases d'un tableau possèdent des valeurs aléatoires
- Le nombre d'éléments maximal d'un tableau est différent du nombre d'éléments significatifs dans un tableau
  - Dans l'exemple précédent le nombre maximal d'éléments est de 100 mais le nombre significatif d'élements est référencé par le paramètre nbEleves
- L'accès aux éléments d'un tableau est direct (temps d'accès constant)
- Il n'y a pas conservation de l'information d'une exécution du programme à une autre

Rappels : tableau à une dimension 4 / 4

- L'affectation ($\leftarrow$) **copie** tous les éléments du tableau dans un autre
- Les opérations de bases sur des tableaux **de même type** sont :
  - L'égalité (=) qui permet de savoir si deux tableaux de même type possèdent des éléments de même valeur ($\forall$i, a[i]=b[i])
  - L'inégalité ($\neq$) qui permet de savoir si deux tableaux de même type possèdent au moins un élément différent ($\exists$i, a[i]$\neq$b[i])

Attention

- On ne peut comparer deux tableaux que si ces derniers sont totalement remplis

Architecture des Systèmes d'Information

Les tableaux à deux dimensions 1 / 3

- On peut aussi avoir des tableaux à deux dimensions (permettant ainsi de représenter par exemple des matrices à deux dimensions)
- On déclare une matrice à deux dimensions de la façon suivante :
  - **Tableau[***intervallePremièreDimension***][***intervalleDeuxièmeDimension***] de** *type des éléments*
- On accéde à la i ème ,j ème valeur d'un tableau en utilisant la syntaxe suivante :
  - *nom de la variable***[***i***][***j***]**

Les tableaux à deux dimensions 2 / 3

- Par exemple si *tab* est défini par
  tab : **Tableau[**1..3**][**1..2**] de** Réel
  - tab[2][1] $\leftarrow$ -1.2
    - met la valeur -1.2 dans la case 2,1 du tableau
  - En considérant le cas où *a* est une variable de type Réel, a $\leftarrow$ tab[2][1]
    - met -1.2 dans a

	1	*2*
1	7.2	5.4
2	-1.2	2
3	4	-8.5

Architecture des Systèmes d'Information

Les tableaux à deux dimensions 3 / 3

- Attention, le sens que vous donnez à chaque dimension est important et il ne faut pas en changer lors de l'utilisation du tableau
  - Par exemple, le tableau tab défini de la façon suivante :
    tab : **Tableau[**1..3**][**1..2**] de** Réel
    tab[1][1] ← 2.0 ;tab[2][1] ← -1.2 ;tab[3][1] ← 3.4
    tab[1][2] ← 2.6 ; tab[2][2] ← -2.9 ; tab[3][2] ← 0.5
    . . . peut permettre de représenter l'une des deux matrices suivantes :
    $\begin{pmatrix} 2.0 & -1.2 & 3.4 \\ 2.6 & -2.9 & 0.5 \end{pmatrix} \begin{pmatrix} 2.0 & 2.6 \\ -1.2 & -2.9 \\ 3.4 & 0.5 \end{pmatrix}$

Exercice

- Donner le corps :
  1. d'une fonction additionnant deux matrices
  2. d'une fonction multipliant deux matrices
  3. d'une fonction transposant une matrice

Tableau à n dimensions 1 / 3

Extension

- Par extension on peut définir et utiliser des tableaux à n dimensions
- Leur déclaration est à l'image des tableaux à deux dimensions, c'est-à-dire :
  - **tableau [**$intervalle_1$**][**$intervalle_2$**]. . . [**$intervalle_n$**] de** *type des valeurs*

Exemple à 3 dimensions

Constante LARGEUR = 4
Constante HAUTEUR = 4
Constante PROFONDEUR = 4
Type Contenu = {vide,pionNoir,pionBlanc}
Type Plateau = **tableau[***1..LARGEUR* **][** *1..PROFONDEUR***][***1..HAUTEUR***] de** Contenu

Tableau à n dimensions 2 / 3

Initialiser plateau

procédure initialiserPlateau **(S** p : Plateau**)**
    Déclaration i,j,k : **Naturel**
debut
    pour i ←1 **à** LARGEUR **faire**
        pour j ←1 **à** PROFONDEUR **faire**
            pour k ←1 **à** HAUTEUR **faire**
                p[i][j][k] ← vide
            finpour
        finpour
    finpour
fin

Tableau à *n* dimensions 3 / 3

Jouer

procédure jouerPion **(E/S** p : Plateau,**E** largeur : 1..LARGEUR, profondeur : 1..PROFONDEUR, pion : pionNoir..pionBlanc**)**

⌊**précondition(s)** hauteurColonne(p,largeur,profondeur) $<$ HAUTEUR

debut

p[largeur][profondeur][hauteurColonne(p,largeur,profondeur)+1] $\leftarrow$ pion

fin

Exercice

- Donner le corps de :
  fonction hauteurColonne **(**p : Plateau, largeur : 1..LARGEUR,profondeur : 1..PROFONDEUR**) :** 0..HAUTEUR

Représentation d'un tableau à 2 dimensions avec un tableau en 1D 1 / 2

Retour sur l'exemple des matrices

- Les matrices mathématiques peuvent être de différentes dimensions $\Rightarrow$ plusieurs types ?
  - On veut seulement que pour certaines opérations elles soient compatibles
- On voudrait un seul type de données pour représenter toutes les matrices en optimisant l'espace mémoire
  - Taille donnée M, on veut pouvoir stocker toutes les matrices tels que $n * m \leq M$

Architecture des Systèmes d'Information

Représentation d'un tableau à 2 dimensions avec un tableau en 1D 2 / 2

Solution

- Utiliser un tableau à une dimension pour stocker une information à deux dimensions $\Rightarrow$ création d'un type Matrice
  - **Type** Matrice **=** **Tableau[**1..MAX**] de Reel**
- Stockage des informations par ligne ou par colonne
  - Nécessité d'une formule mathématique pour passer (i, j) à k $\Rightarrow$ fonction *rang*
  - Utilisation de deux informations complémentaires : le nombre de lignes (n) et le nombre de colonnes (m) de la matrice
- Utilisation d'une fonction et d'une procédure pour s'abstraire de cette représentation :
  - **procédure** fixerValeur **(E/S** ma : Matrice,**E** i,j,n,m : **NaturelNonNul**, val : **Reel)**

    ⌊**précondition(s)** n*m≤MAX et i≤ n et j≤m
  - **fonction** obtenirValeur **(**ma : Matrice, i,j,n,m : **NaturelNonNul) : Reel**

    ⌊**précondition(s)** n*m≤MAX et i≤ n et j≤m

Qu'est ce qu'un tri ? 1 / 2

Prérequis

Un tri s'applique sur un tableau à une dimension d'éléments possédant un ordre total :

$$\forall i < j \text{ on a } t[i] \leq t[j] \text{ ou } t[j] \leq t[i]$$

Définition

Un tri est un algorithme qui prend en entrée un tableau et qui donne en sortie ce même tableau avec les éléments ordonnés suivant une relation R donnée

En pratique c'est une procédure qui possède la signature suivante :
procédure trier **(E/S** t : **Tableau[**1..MAX**] d'**Element,**E** nbElements : **Naturel)**

Qu'est ce qu'un tri ? 2 / 2

Attention

Le tri s'effectue par permutations successives des éléments du tableau (utilisation de la procédure échanger)
$\Rightarrow$ on ne compte pas les éléments ! ! !

Dans les tris que nous allons étudier les éléments sont des entiers et la relation d'ordre est $<$ (tri en ordre croissant)

procédure trier **(E/S** t :**Tableau[**1..MAX**] d'Entier**,**E** nb :**Naturel)**

Architecture des Systèmes d'Information

Un premier exemple : le tri à bulles 1 / 3

Principe

- On parcourt le tableau t a trié
- Dès que deux éléments consécutifs $t[i]$ et $t[i+1]$ ne sont pas bien placés, c'est-à-dire que $t[i] > t[i+1]$, alors on les échange
- Si au moins un échange a été effectué, c'est que le tableau n'était pas trié, on le reparcourt de nouveau

Quelques points importants

- On compare deux éléments consécutifs, l'itération doit donc aller de 1 à $nb - 1$
- Il faut savoir si un échange a été effectué, il faut donc utiliser une variable booléenne

Architecture des Systèmes d'Information

Un premier exemple : le tri à bulles 2 / 3

algorithme

```
procédure triABulles (E/S t :Tableau[1..MAX] d'Entier,E nb :Naturel)
    Déclaration   i : Naturel
                  estTrie : Booleen
debut
    repeter
        estTrie ← VRAI
        pour i ←1 à nb-1 faire
            si t[i]>t[i+1] alors
                echanger(t[i],t[i+1])
                estTrie ← FAUX
            finsi
        finpour
    jusqu'a ce que estTrie
fin
```

Architecture des Systèmes d'Information

Un premier exemple : le tri à bulles 3 / 3

Calcul de complexité

- Dans le meilleur des cas, le tri d'un tableau déjà trié est en $\Omega(n)$
- Dans le pire des cas, le tri d'un tableau trié « en sens inverse », il faudra n itérations pour « redescendre » le plus petit élément (celui qui est le plus éloigné de sa position finale). La complexité est alors en $O(n^2)$

Amélioration

- Le tri « shaker »

Exercice

- Donnez l'algorithme du tri « shaker »

Tris itératifs classiques

Principes des tris itératifs classiques

On va parcourir entièrement le tableau, en appliquant à chaque itération i l'une des deux stratégies suivantes :

- on recherche l'élément qui doit se placer à la i^{eme} place
  - **Tri par sélection**
- on recherche la place où va être positionné le i^{eme} élément
  - **Tri par insertion**

Remarque

Après chaque itération, le "sous-tableau" $t[1..i]$ est trié
$\Rightarrow$ À la fin le tableau t sera trié

Architecture des Systèmes d'Information

Le tri par sélection 1 / 3

Principe (par minimum successif)

On parcourt le tableau t (i variant de 1 à $nb - 1$) et à chaque itération i

- on détermine l'indice j du plus petit entier de t sur l'intervalle $[i..nb]$
- on échange $t[i]$ et $t[j]$

Le tri par sélection 2 / 3

algorithme

fonction indiceDuMinimum **(**t :**Tableau[**1..MAX**] d'Entier**, borneInf,borneSup :**Naturel) : Naturel**

  Déclaration i,resultat : **Naturel**

debut
  resultat ← borneInf
  pour i ←borneinf+1 **à** borneSup **faire**
    si t[i]<t[resultat] **alors**
      resultat ← i
    finsi
  finpour
  retourner resultat
fin

procédure triParMinimumSuccessif **(E/S** t :**Tableau[**1..MAX**] d'Entier**,**E** nb :**Naturel)**

  Déclaration i : **Naturel**

debut
  pour i ←1 **à** nb-1 **faire**
    echanger(t[i],t[indiceDuMinimum(t,i,nb)])
  finpour
fin

Le tri par sélection 3 / 3

Exercice

- Quelle est la complexité $T(n)$ dans le meilleur et le pire des cas ?

Le tri par insertion 1 / 4

Principe

On parcourt le tableau t (i variant de 2 à nb) et à chaque itération i

- on détermine l'indice j de la position de t[i] dans l'intervalle $[1..i]$
- on insère $t[i]$ à la j^{eme} place

Le tri par insertion 2 / 4

algorithme

fonction obtenirIndiceDInsertion **(**t :**Tableau[**1..MAX**]**
d'Entier,borneSup :**Naturel**,lEntier :**Entier) : Naturel**

procédure decaler **(E/S** t :**Tableau[**1..MAX**] d'Entier**,**E** borneInf,borneSup :**Naturel)**

procédure triParInsertion **(E/S** t :**Tableau[**1..MAX**] d'Entier**,**E** nb :**Naturel)**

```
    Déclaration   i,j : Naturel
                  temp : Entier
debut
    pour i ←2 à nb faire
        j ← obtenirIndiceDInsertion(t,i,t[i])
        temp ← t[i]
        decaler(t,j,i)
        t[j] ← temp
    finpour
fin
```

Le tri par insertion 3 / 4

Exercices

1. Donner les algorithmes de :
   - **fonction** obtenirIndiceDInsertion **(**t :**Tableau[**1..MAX**] d'Entier**,borneSup :**Naturel**,lEntier :**Entier) : Naturel**
     - avec un algorithme séquentiel puis dichotomique
   - **procédure** decaler **(E/S** t :**Tableau[**1..MAX**] d'Entier**,**E** borneInf,borneSup :**Naturel)**
2. Quelle est la complexité $T(n)$ dans le meilleur et le pire des cas ?

Architecture des Systèmes d'Information

Le tri par insertion 4 / 4

Un autre algorithme

La recherche de l'indice d'insertion et le décalage se font « en même temps » :

procédure inserer **(E/S** t : **Tableau[**1..MAX**] d'Entier**, **E** position : **Naturel)**

procédure triParInsertion **(E/S** t :**Tableau[**1..MAX**] d'Entier**, **E** nb :**Naturel)**
    Déclaration i,j : **Naturel**
        temp : **Entier**
debut
    pour i ←2 **à** nb **faire**
        inserer(t,i)
    finpour
fin

Exercices

- Donnez l'algorithme de la procédure *inserer*
- Quelle est la complexité $T(n)$ dans le meilleur et le pire des cas ?