Tableaux

I2 - Algorithmique et Programmation structurée

Julien Saunier, Alexandre Pauchet, Pierrick Tranouez

Plan...

Tableaux à 1 dimension

Les tableaux

Rappel

Un tableau est un type de données permettant de représenter plusieurs valeurs dans une même variable.

Propriétés

Un tableau a :

- une taille déterminée
- un unique type de données pour toutes ses valeurs

Opérateurs

- l'opérateur `[ ]` permet d'accéder à une valeur spécifique du tableau
- l'affectation copie toutes les valeurs d'un tableau dans un autre
- les opérateurs sur les valeurs du tableau sont ceux du type de données

Déclaration d'un tableau en pseudo-code

Pseudo-code

- tab : **Tableau[**1..26**] de Reel**
  - déclare une variable de type tableau de 26 réels
  - les indices valides vont de 1 à 26
- tab2 : **Tableau[**'a'..'f'**] de Caractere**
  - déclare une variable de type tableau de 6 caractères
  - les indices valides vont de 'a' à 'f'

Définition de type

La déclaration d'un type spécifique facilite l'écriture

- **Type** Notes = **Tableau[**1..26**] de Reel**
  - définit un nouveau type appelé Notes, qui est un tableau de 26 réels
- tab3 : Notes
  - déclare une variable de type Notes (tableau de 26 réels)
  - les indices valides vont de 1 à 26

Exemple : somme de nombres, taille utilisée

Déclaration:
Constante MAX = 100
Type Notes = **Tableau**[1..MAX] **de Reel**
lesNotes : Notes
i, nbEleves : **Naturel**
somme : **Reel**
debut
    repeter
        ecrire('Nombre d''eleves (maximum ', MAX, ') : ')
        lire(nbEleves)
    jusqu'a ce que (nbEleves $>$ 0) et (nbEleves $\leq$ MAX)
    pour i $\leftarrow$1 **à** nbEleves **faire**
        ecrire('Note de l''eleve numero ', i, ' :')
        lire(lesNotes[i])
    finpour
    somme $\leftarrow$ 0
    pour i $\leftarrow$1 **à** nbEleves **faire**
        somme $\leftarrow$ somme + lesNotes[i]
    finpour
    ecrire('La moyenne est de : ', somme/nbEleves)
fin

Dépassement de tableau

Dépassement

Le dépassement de tableau se produit, **à l'exécution**, lors de l'accès à un indice n'existant pas dans un tableau

Déclaration: t : **Tableau['A'..'L'] de Reel**
debut
    t['M'] $\leftarrow$ 10
fin

Effet

L'effet d'un dépassement de tableau peut

- provoquer une erreur à l'exécution (arrêt du programme)
- modifier de manière aléatoire d'autres variables
- n'avoir aucun effet

Complexité

Notion de complexité

Complexité

La complexité est un des critères principaux de qualité d'un programme. Elle chercher à estimer l'efficacité du programme, soit en calcul, soit en espace, indépendamment de la puissance de calcul / espace mémoire de la machine.

Calcul de complexité

- dans le pire des cas (O),
- en moyenne (Ω),
- dans le meilleur des cas (Θ)

Valeurs classiques

- $O(1)$ - constante (ex. une suite séquentielle d'instructions)
- $O(log\ n)$ - logarithmique (ex. boucle qui divise n à chaque itération)
- $O(n)$ - linéaire (ex. parcours complet d'un tableau)
- $O(n^2)$ - quadratique (ex. boucle dans une boucle)
- $O(2^n)$ - combinatoire (ex. test de toutes les permutations possibles)

Ordres de grandeurs

	n= 10	n=50
$O(1)$	1 s	1 s
$O(log\ n)$	2s	2 s
$O(n)$	10 s	50 s
$O(n^2)$	100 s	250 s
$O(2^n)$	17 minutes μs	13031248921 jours

Algorithmique sur tableaux

Algorithmes simples sur les tableaux

Manipulations classiques :

- Parcours de tableaux
- Insertion/suppression de valeurs
- Recherche

Parcours de tableau

Parcours séquentiel

L'objectif est de parcourir les éléments du tableau pour y effectuer une opération donnée (modification, affichage, lecture...)

- Il est possible de ne pas considérer tous les éléments
- Le parcours peut être interrompu avant la fin

Exemples : Affichage de l'ensemble des notes d'un groupe, calcul de la moyenne, . . .

Complexité ?

Ajout/Retrait d'éléments 1 / 4

Ajout en fin de tableau

Il y a, au plus, deux variables à modifier

- La valeur après le dernier indice utilisé pour un ajout, celle du dernier indice utilisé pour un retrait
- la taille du tableau

```
procédure ajouterEnFin (E val : Entier, E/S tab : Tableau[1 .. MAX] de Entier ;
taille : Naturel)
debut
    si (taille < MAX) alors
        taille ← taille + 1
        tab[taille] ← val
    finsi
fin
```

Complexité ?

Ajout/Retrait d'éléments 2 / 4

Retrait en fin de tableau

Il y a, au plus, une variable à modifier

- la taille du tableau

```
procédure supprimerEnFin (E/S taille : Naturel)
debut
    si (taille > 0) alors
        taille ← taille - 1
    finsi
fin
```

Complexité ?

Ajout/Retrait d'éléments 3 / 4

à une position autre que la dernière

Pour modifier le tableau à une position *pos* différente de la dernière utilisée, il faut :

- modifier la taille du tableau ;
- décaler toutes les valeurs situées entre *pos* et la fin du tableau
- éventuellement ajouter la nouvelle valeur à *pos*

```
procédure ajouter (E val : Entier, pos : Naturel, E/S tab : Tableau[1 .. MAX] de Entier ;
taille : Naturel)
    Déclaration  i : Naturel
debut
    si (taille < MAX) et (pos ≥1) et (pos ≤ taille) alors
        pour i ←taille à pos pas de -1 faire
            tab[i+1] ← tab[i]
        finpour
        tab[pos] ← val
        taille ← taille + 1
    finsi
fin
```

Ajout/Retrait d'éléments 4 / 4

Retirer à une position autre que la dernière

Pour modifier le tableau à une position *pos* différente de la dernière utilisée, il faut

- modifier la taille du tableau ;
- décaler toutes les valeurs situées entre *pos* et la fin du tableau

```
procédure supprimer (E pos : Naturel, E/S tab : Tableau[1 .. MAX] de Entier ;
taille : Naturel)
    Déclaration  i : Naturel
debut
    si (pos ≥1) et (pos < taille) alors
        pour i ←pos à taille-1 faire
                tab[i] ← tab[i+1]
        finpour
        taille ← taille - 1
    finsi
fin
```

Recherche dans un tableau 1 / 3

Type de recherche

- par indice
- par valeur
- par critères

Recherche par indice

C'est le plus simple. On accède directement à la valeur de l'indice souhaité par l'opérateur [ ].
Ex : pour accéder à l'opération d'indice 3
`operations[3]`

Complexité ?

Recherche dans un tableau 2 / 3

Recherche par valeur

On souhaite trouver le dernier indice d'une valeur donnée

```
fonction rechercheValeur (val : Entier ; tab : Tableau[1 .. MAX] de Entier ; taille : Naturel) : Naturel
    Déclaration  i,resultat : Naturel
debut
    resultat ← 0
    pour i ←1 à taille faire
        si tab[i] = val alors
            resultat ← i
        finsi
    finpour
    retourner resultat
fin
```

Complexité ?

Recherche dans un tableau 3 / 3

Recherche par critères

On souhaite trouver le dernier indice d'une valeur qui satisfait un critère de recherche

```
fonction rechercheMin (tab : Tableau[1 .. MAX] de Entier ; taille : Naturel) : Naturel

    Déclaration  i,resInd : Naturel
debut
    resInd ← 1
    pour i ←2 à taille faire
        si tab[i] < tab[resInd] alors
            resInd ← i
        finsi
    finpour
    retourner resInd
fin
```

Complexité ?

Exercice 1 / 2

Énoncé

On souhaite calculer la moyenne en I1 de chaque groupe. Le programme a besoin de 3 types de tableau :

- un tableau qui fait correspondre à un numéro d'élève sa note en I1 ;
- un tableau qui fait correspondre à un numéro d'élève la lettre ; de son groupe ;
- un tableau qui fait correspondre à une lettre de groupe la moyenne du groupe.

On supposera avoir 50 élèves répartis en 5 groupes (de A à E) de tailles égales.
Définissez les 3 types correspondants puis écrivez une procédure qui calcule la moyenne par groupe (on supposera que les valeurs des deux premiers tableaux sont préalablement saisies).

Exercice 2 / 2

Type de données

Type Notes = **Tableau**[1..50] **de Reel**
Type Groupes = **Tableau**[1..50] **de Caractere**
Type Moyennes = **Tableau**['A'..'E'] **de Reel**

Une solution

Type Notes = **Tableau[**1..50**] de Reel**
Type Groupes = **Tableau[**1..50**] de Caractere**
Type Moyennes = **Tableau[**'A'..'E'**] de Reel**

procédure moyenne **(E** noteEleve : Notes ; groupeEleve : Groupes ; **S** moyGrp : Moyennes ;**)**
    Déclaration grp : char ; i : integer ; score : real ;
debut
    pour grp ←'A' **à** 'E' **faire**
        score ← 0
        pour i ←1 **à** 50 **faire**
            si groupeEleve[i] = grp **alors**
                score ← score + noteEleve[i]
            finsi
        finpour
        moyGrp[grp] ← score / 10
    finpour
fin

Une solution plus efficace

Type Notes = **Tableau**[1..50] **de Reel**
Type Groupes = **Tableau**[1..50] **de Caractere**
Type Moyennes = **Tableau**['A'..'E'] **de Reel**

procédure moyenne (**E** noteEleve : Notes ; groupeEleve : Groupes ; **S** moyGrp : Moyennes ;)
    Déclaration grp : char ; i : integer ;
debut
    pour grp ←'A' **à** 'E' **faire**
        moyGrp[grp] ← 0
    finpour
    pour i ←1 **à** 50 **faire**
        moyGrp[groupeEleve[i]] ← moyGrp[groupeEleve[i]] + noteEleve[i]/10
    finpour
fin

Tableaux triés

Tableaux triés

Principe

Un tableau trié est un tableau dont les valeurs sont stockées dans un ordre qui correspond à celui de leur type de données. Cet ordre peut être croissant ou décroissant.

initialement :

5	-3	4	8	-1

exécution d'une procédure de tri

```
procedure tri(E/S tab : Tableau, E taille: naturel)
```

après exécution :

-3	-1	4	5	8

Tri par insertion 1 / 2

Principe

- On considère que le tableau est initialement trié (cela peut correspondre au cas d'un tableau vide)
- Chaque fois qu'une valeur doit être ajoutée, on cherche dans le tableau l'indice i d'une valeur immédiatement supérieure
- Si une telle valeur n'existe pas, on ajoute la nouvelle valeur à la fin du tableau
- Si elle existe, on l'ajoute à la position i

Tri par insertion 2 / 2

procédure ajouterOrdreCroissant **(E** val : **Entier**, **E/S** tab : **Tableau[**1 .. MAX**] de Entier** ; taille : **Naturel)**

Déclaration pos : **Naturel**

debut

si (taille $<$ MAX) **alors**

pos $\leftarrow$ taille $+1$

tant que (pos $>$ 1) et (tab[pos - 1] $>$ val) **faire**

tab[pos] $\leftarrow$ tab[pos - 1]

pos $\leftarrow$ pos - 1

fintantque

tab[pos] $\leftarrow$ val

taille $\leftarrow$ taille $+$ 1

finsi

fin

Complexité ?

Recherche par dichotomie 1 / 2

Principe

Le propriété de tri est utilisée pour diviser l'espace de recherche par 2 après chaque test

Déroulement

1. On cherche une valeur x dans un tableau de N valeurs triées
2. On compare x à la valeur située à l'indice $N/2$
3. Tant que x est différent de cette valeur, on recommence la recherche mais sur la première ou la seconde moitié du tableau (selon que x est plus petit ou plus grand que la valeur testée)
4. L'algorithme s'arrête quand x a été trouvé ou quand l'espace de recherche n'est plus divisible

Recherche par dichotomie 2 / 2

```
fonction rechercheDichotomie (val : Entier ; tab : Tableau[1 .. MAX] de Entier ; taille : Naturel) : Naturel
    Déclaration  min,max,pos,res : Naturel
debut
    res ← 0
    min ← 0
    max ← taille
    tant que max > min faire
        pos ← (min + max+1) div 2
        si (tab[pos] < val) alors
            min ← pos
        sinon
            si (tab[pos] > val) alors
                max ← pos-1
            sinon
                res ← pos
                max ← min-1
            finsi
        finsi
    fintantque
    retourner res
fin
```

Complexité ?

Tableau à 2 dimensions

Combiner des indices dans un tableau à 1 dimension 1 / 1

Exemple fil rouge

Exemple de représentation d'un ensemble de notes pour un ensemble d'élèves : chaque note est un réel et doit être stockée dans un tableau avec deux valeurs d'indice : le numéro de l'élève et le numéro du cours.

$$\begin{pmatrix} 12.5 & 4.0 & 16.5 & 11.0 \\ 16.0 & 7.0 & 18.0 & 15.0 \\ 6.5 & 0.5 & 6.0 & 9.0 \end{pmatrix}$$

Déclaration d'un tableau à 2 dimensions

Tableau 2D

- Un tableau peut être déclaré avec 2 dimensions (matrice).
- L'accès à un élément d'un tableau à 2 dimensions se fait grâce à 2 indices.

Exemple

- On déclare un tableau à deux dimensions de la façon suivante :
  - **Tableau** [IntPremièreDimension] [IntDeuxièmeDimension]
- On accède à la i^{eme}, j^{eme} valeur d'un tableau à 2 dimensions par :
  - *nomdelavariable*[*i*][*j*]

Pseudo-code : déclaration d'un tableau à 2 dimensions

- tab : **Tableau** [1..20][1..30] **de Reel**
  - déclare une variable tab de type tableau à 2 dimensions de $20 * 30 = 600$ réels.
  - les indices de la première dimension vont de 1 à 20 et ceux de la deuxième dimension de 1 à 30.
- tab2 : **Tableau** [1..15][1..20] **de Caractere**
  - déclare une variable tab2 de type tableau à 2 dimensions de $15 * 20 = 300$ caractères.
  - les indices de la première dimension vont de 1 à 15 et ceux de la deuxième dimension de 1 à 20.

Accès et affectation 1 / 2

- $tab[2][1] \longleftarrow -1.2$
  - met la valeur -1.2 dans la case $2, 1$ du tableau
- En considérant une variable a de type réel, $a \longleftarrow tab[2][1]$
  - met la valeur -1.2 dans a
- **lire(tab**$[10][7]$**)**
  - met l'entier saisi par l'utilisateur dans la case correspondante au 10^{eme} ligne et au 7^{eme} colonne
- **ecrire(tab[**$10][7]$**)**
  - affiche la valeur de la case correspondante au 10^{eme} ligne et au 7^{eme} colonne

Accès et affectation 2 / 2

- Ainsi l'extrait suivant de l'algorithme :
  tab : **Tableau** [1..2][1..3] **de Reel**
  $tab[1][1] \longleftarrow -4$
  $tab[1][2] \longleftarrow 2$
  $tab[1][3] \longleftarrow 1.3$
  $tab[2][1] \longleftarrow -1.5$
  $tab[2][2] \longleftarrow 9$
  $tab[2][3] \longleftarrow 2.3$
  peut être représenté graphiquement (ou sous forme matricielle) par :

-4	2	1.3
- 1.5	9	2.3

L'ordre des dimensions est important !

- Attention, l'ordre 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 Reel**
    $tab[1][1] \longleftarrow 2.0$; $tab[2][1] \longleftarrow -1.2$; $tab[3][1] \longleftarrow 3.4$
    $tab[1][2] \longleftarrow 2.6$; $tab[2][2] \longleftarrow -2.9$; $tab[3][2] \longleftarrow 0.5$
    peut permettre de représenter l'un des deux tableaux suivants :
    $$\begin{pmatrix} 2.0 & -1.2 & 3.4 \\ 2.6 & -2.9 & 0.5 \end{pmatrix} \quad \begin{pmatrix} 2.0 & -2.6 \\ -1.2 & -2.9 \\ 3.4 & 0.5 \end{pmatrix}$$

Exercice

Soit un tableau à deux dimensions, afficher le contenu du tableau dans le terminal.

Exercice : somme de tableaux 2D

Faire la somme de tableaux 2D (3×3)

On veut réaliser un programme qui nous permet d'additionner chaque valeur de 2 tableaux à 2 dimensions (chacune de taille 3).

Un exemple de somme

$$\begin{pmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 5.0 & 6.0 \\ 7.0 & 8.0 & 9.0 \end{pmatrix} + \begin{pmatrix} 1.0 & 2.0 & 3.0 \\ 2.0 & 3.0 & 1.0 \\ 3.0 & 1.0 & 2.0 \end{pmatrix} = \begin{pmatrix} 1.0+1.0 & 2.0+2.0 & 3.0+3.0 \\ 4.0+2.0 & 5.0+3.0 & 6.0+1.0 \\ 7.0+3.0 & 8.0+1.0 & 9.0+2.0 \end{pmatrix}$$

$$= \begin{pmatrix} 2.0 & 4.0 & 6.0 \\ 6.0 & 8.0 & 7.0 \\ 10.0 & 9.0 & 11.0 \end{pmatrix}$$

Représentation du problème

Type de données

Constante TAILLE = 3

Type Tab33 = **Tableau**[1..TAILLE][1..TAILLE] **de** **Reel**

Sous-programme demandé

- **Procedure** somme(E m1, m2 : Tab33, S m3 : Tab33)

Solution : somme de deux tableaux 2D

Somme de deux tableaux 3 × 3

procédure somme **(E** m1, m2 : **Tab33**, **S** m3 : **Tab33)**

Déclaration i,j : **Naturel**

debut

pour i ←1 **à** 3 **faire**

pour j ←1 **à** 3 **faire**

m3[i][j] ← m1[i][j] + m2[i][j]

finpour

finpour

fin

Complexité ?

Algorithmes sur des tableaux à 2 dimensions

1. Parcours par dimension
2. Insertion de valeurs dans un tableau à 2 dimensions
3. Suppression de valeurs dans un tableau à 2 dimensions

Tableaux à 2 dimensions : l'exemple

On considère un tableau permettant de représenter la note (un réel) obtenue par chaque élève à chaque cours

- Première dimension : le numéro des élèves
- Seconde dimension : le numéro des cours

Taille des dimensions

Pour conserver une écriture indépendante du nombre réel d'élèves et de cours, il est utile de manipuler deux tailles par dimension

- une taille maximale réservée en mémoire pour le tableau (MAX_ELEVES, MAX_COURS)
- le nombre d'indices réellement utilisés dans chaque dimension (nbEleves, nbCours)

Constante MAX_ELEVES = 400
Constante MAX_COURS = 20
Type Notes = **Tableau**[1..MAX_ELEVES][1..MAX_COURS] **de** **Reel**

Parcourir un tableau

Afficher un tableau

```
procédure afficherTableau (E lesNotes : Notes ; nbEleves, nbCours : Naturel)
    Déclaration   i,j : Naturel
debut
    pour i ←1 à nbEleves faire
        ecrire('Elevre no', i, ' :')
        pour j ←1 à nbCours faire
            ecrire(' Cours no', j, ' : ',lesNotes[i][j])
        finpour
        ecrire(Saut de ligne)
    finpour
fin
```

Complexité ?

Insertion/Suppression sur 2 dimensions

Comme pour les tableaux à une dimension, il faut distinguer deux catégories d'ajout ou de suppression

- en fin de tableau
- à une autre position

Mais, pour conserver une représentation matricielle, il faudra

- ajouter/supprimer une ligne complète
- ajouter/supprimer une colonne complète

Insertion d'une colonne en fin

exemple

On souhaite insérer les notes d'un nouveau cours (fournies dans `tab`).

Type tabReelsC = **Tableau**[1..MAX_ELEVES] **de Reel**

procédure ajouteCoursEnFin (**E** tab : tabReelsC, nbEleves : **Naturel**, **E/S** lesNotes : Notes, nbCours : **Naturel**)

```
    Déclaration   i : Naturel
debut
    si nbCours < MAX_COURS alors
        nbCours ← nbCours + 1
        pour i ←1 à nbEleves faire
            lesNotes[i][nbCours] ← tab[i]
        finpour
    finsi
fin
```

Suppression d'une ligne en fin

exemple

On souhaite supprimer les notes du dernier élève.

```
procédure supprimeDernierEleve (E/S nbEleves :Naturel)
debut
    si nbEleves > 0 alors
        nbEleves ← nbEleves - 1
    finsi
fin
```

Complexité ?

Insertion d'une ligne dans le tableau

exemple

On souhaite insérer les notes d'un nouvel élève à placer à l'indice pos.

```
Type tabReelsE = Tableau[1..MAX_COURS] de Reel


procédure ajouteEleve (E tab : tabReelsE, pos, nbCours : Naturel, E/S lesNotes : Notes, nbEleves : Naturel)
    Déclaration   i,j : Naturel
debut
    si (nbEleves < MAX_ELEVES) et (pos >= 1) et (pos <= nbEleves) alors
        nbEleves ← nbEleves + 1
        pour j ←nbEleves à pos+1 pas de -1 faire
            pour i ←1 à nbCours faire
                lesNotes[j][i] ← lesNotes[j-1][i]
            finpour
        finpour
        pour i ←1 à nbCours faire
            lesNotes[pos][i] ← tab[i]
        finpour
    finsi
fin
```

Complexité ?

Suppression d'une colonne dans le tableau

exemple

On souhaite supprimer les notes du cours situé à l'indice pos.

```
procédure supprimeCours (E pos, nbEleves : Naturel, E/S lesNotes : Notes, nbCours : Naturel)
    Déclaration   i,j : Naturel
debut
    si (nbCours > 0) et (pos >= 1) et (pos <= nbCours) alors
        nbCours ← nbCours - 1
        pour i ←pos à nbCours faire
            pour j ←1 à nbEleves faire
                lesNotes[j][i] ← lesNotes[j][i+1]
            finpour
        finpour
    finsi
fin
```

Représentation mémoire 1 / 3

1 dimension

- Un tableau de N lignes et M colonnes contient $N \times M$ valeurs qui peuvent donc être stockées dans un tableau de taille $N \times M$
- les M premières valeurs du tableau représentent la 1ère ligne du tableau
- les valeurs comprises entre les indices $M + 1$ et $2 \times M$ représentent la 2ème ligne du tableau ...
- les valeurs comprises entre les indices $(N - 1) \times M + 1$ et $N \times M$ représentent la dernière ligne du tableau

```
lesNotes : Tableau[1..12] de Réel
```

1	2	3	4	5	6	7	8	9	10	11	12
12.5	4.0	16.5	11.0	16.0	7.0	18.0	15.0	6.5	0.5	6.0	9.0

Représentation mémoire 2 / 3

Accéder à une valeur $N_{i,j}$

Constante MAX = 1000
Type Notes1D = **Tableau[**1..MAX**] de Reel**

fonction aPourValeur **(**notes : Notes1D, nbCols, lig, col : **Entier) : Reel**
debut
    retourner notes[(lig-1)*nbCols+col]
fin

Représentation mémoire 3 / 3

Modifier une valeur $N_{i,j}$

procédure modifierValeur **(E/S** notes : Notes1D, **E** nbCols, lig, col : **Entier**, val : **Reel)**
debut
    si (lig $\geq$ 1) et (lig*nbCols $\leq$ MAX) et (col $\geq$ 1) et (col $\leq$ nbCols) **alors**
        notes[(lig-1)*nbCols+col] $\leftarrow$ val
    finsi
fin

Tableau à *n* dimensions

Tableau à *n* dimensions

Définition

Par extension, on peut définir des tableaux à *n* dimensions !

Intervalle et indice

Les tableaux à *n* dimensions sont déclarés de la même façon que les tableaux à 2 dimensions, mais cette fois en précisant *n* intervalles (1 intervalle par dimension).

Pseudo-code

tableau [*int01*][*int02*]...[*intN*] **de** *TypeContenu*
tableau [*int01*,*int02*,...,*intN*] **de** *TypeContenu*

Exemple de tableau à *n* dimensions 1 / 4

Historique de parties de Mastermind

Supposons que nous voulions représenter dans un tableau un historique des combinaisons proposées sur plusieurs parties de Mastermind. Nous pourrions définir un tableau avec 3 dimensions :

- Une dimension représentant le numéro de la partie
- Une dimension représentant le numéro de la proposition dans une partie
- Une dimension représentant la couleur d'un pion dans une combinaison

Exemple de tableau à *n* dimensions 2 / 4

Les données

Constante MAX_PARTIES = 100

Type Couleur = {jaune, orange, rose, rouge, vert, bleu}
Type Combinaison = Tableau[1..4] de Couleur

{*type du tableau représentant l'ensemble de l'historique*}
Type Historique = **tableau[***1..MAX_PARTIES* **][** *1..10***][***1..4***] de** Couleur

{*type du tableau représentant le nombre de coups joués dans chaque partie*}
Type NbCoups = **tableau[***1..MAX_PARTIES***] de** Naturel

Exemple de tableau à *n* dimensions 3 / 4

Créer une nouvelle partie

```
procédure creerNouvellePartie (E/S tabNbCoups : NbCoups ; E/S
nbParties : Naturel)
debut
    si nbParties < MAX_PARTIES alors
        nbParties ← nbParties + 1
        tabNbCoups[nbParties] ← 0
    finsi
fin
```

Exemple de tableau à *n* dimensions 4 / 4

Ajouter une combinaison

procédure ajouterCombinaison **(E** numPartie : Naturel ; **E** combi : Combinaison ; **E/S** tabNbCoups : NbCoups ; **E/S** histo : Historique**)**

```
    Déclaration  i : Naturel
debut
    si tabNbCoups[numPartie] < 10 alors
        tabNbCoups[numPartie] ← tabNbCoups[numPartie] + 1
        pour i ←1 à 4 faire
            histo[numPartie][tabNbCoups[numPartie]][i] ← combi[i]
        finpour
    finsi
fin
```

Énoncé du problème

Enoncé

Le système de gestion des présences est un logiciel développé pour l'assiduité des élèves tous les jours dans leurs écoles ou instituts. Le système facilite l'accès et la vérification sur une année et précise la présence d'un élève en particulier dans une classe particulière pendant une séance particulière.

Élément important

Le système propose de noter la présence pour chaque étudiant par :

- année d'étude ($1^{\text{ère}}$ à la $5^{\text{ème}}$) ;
- cours ;
- semaine dans le calendrier annuel.

Représentation du problème

Les années d'étude (énumération)

Type AnneeEtude = {STPI1, STPI2, DEPART3, DEPART4, DEPART5}

Numéro de l'étudiant, identifiant du cours et semaine

On représente :

- le numéro de l'étudiant par un Naturel compris entre 1 et 1500,
- l'identifiant du cours par un Naturel compris entre 1 et 50,
- la semaine par un Naturel compris entre 1 et 28.

Type de données

Type TableauDePrésence =
Tableau[*STPI1...DEPART5***][**1...1500**][**1...50**][**1...28**] de** *Booléen*

Méthodes demandées

Sous-programmes

- **procédure** fixerPresence(**E/S** t : **TableauDePresence**, **E** annee : **AnneeEtude**, **E** idEtudiant, idCours, semaine : **Naturel**, **E** present : **Booléen**)
- **Fonction** aÉtéPresent(t : **TableauDePresence**, annee : **AnneeEtude**, idEtudiant, idCours, semaine : **Naturel**) : **Booléen**

Méthode demandée

Fonction aÉtéPresent(t : **TableauDePresence**, annee : **AnneeEtude**, idEtudiant, idCours, semaine : **Naturel**) : **Booléen**

Solutions

Vérifier la présence

fonction aÉtéPresent **(**t : **TableauDePresence**, annee : **AnneeEtude**,idEtudiant, idCours, semaine : **Naturel) : Booleen**
debut
    retourner t[annee][idEtudiant][idCours][semaine]
fin

Tableaux à N dimensions en Pascal

Définition/Déclaration

- Il y a deux façon de déclarer des tableaux à *n* dimensions :
  1. en considérant que l'on a un tableau de tableau ... de tableau

```
array [indiceDebut1..indiceFin1] of array[
    indiceDebut2..indiceFin2]
of ... of array[indiceDebutN..indiceFinN] of
    TypeDesElements
```

  2. en spécifiant directement *n* intervalles séparés par une , :

```
array [indiceDebut1..indiceFin1, indiceDebut2..indiceFin2,
    ... ,indiceDebutN..indiceFinN] of TypeDesElements
```