Chapitre 4

Les tables à une dimension

Chapitre 4 Les tables à une dimension

I Introduction par l'exemple

- On veut calculer la moyenne générale d'une promotion.
- De plus on veut connaître le nombre d'étudiants ayant une moyenne supérieure à la moyenne générale.
- Pour pouvoir conserver toutes les moyennes partielles calculées correspondant chacune à la moyenne d'un étudiant, on utilise une structure de donnée qui permet de sauvegarder d'un seul tenant ces moyennes partielles, plutôt que d'utiliser autant de variables que d'étudiants traités.
- Une fois le calcul de la moyenne générale terminée, on pourra définir le traitement à effectuer pour déterminer le nombre de moyennes partielles *supérieures* à la moyenne générale.

Chapitre 4 Les tables à une dimension

Représentation : utilisation d'un système tabulaire ou Table.

Tabmoy :

1	2	3	4	5	….	i	i+1	….	……..
9	8.5	10.3	15.5	11.25	….			….	……..
Moyenne étudiant1	Moyenne étudiant2					Moyenne étudianti	Moyenne étudianti+1		

Table des moyennes

Une case sera définie à l'aide
- d'un nom de table
- d'un numéro ou indice mis entre [ ]

Exemple : Tabmoy[3] = 10.3 Tabmoy[5] = 11.25

Plus généralement : Tabmoy[i] : contenu de la case n° i de la table Tabmoy ⇒ ***moyenne de l'étudiant n° i***

Chapitre 4 Les tables à une dimension

Une table est caractérisée par :

❖sa *taille physique* : nombre de cases mémoires réservées

❖le *type de son contenu* (*c.à.d* le **type** des **éléments** stockés)

❖La **taille** doit être suffisamment **grande** pour contenir les valeurs que l'on veut **stocker** dans la table.

On distinguera la notion de **taille physique** de la table de la notion de **nombre de valeurs effectives** stockées dans la table.

Exemple : Taille physique = 150 cases

Nombre de valeurs stockées: 45

Chapitre 4 Les tables à une dimension

En utilisant la notion de table :

✓ Définition d'un type qui permettra de déclarer la variable table Tabmoy :

T_Tabmoy (**type**) = **table** [1**..**1000] réel

Où :

- borne inférieure : 1
- borne supérieure : 1000
- [1..1000] : intervalle de définition ⇒ *domaine de définition des numéros de cases*

✓ Déclaration de variable dans le type créé :

Tabmoy(T_Tabmoy) : table des moyennes

Chapitre 4 Les tables à une dimension

Avec l'exemple du calcul de la moyenne, on avait :

algo Calcul_moyenne	**Ord**	**Lexique**
ecrire("Moyenne générale : ", moygen)	5	i (entier): compteur
pour i ← 1 **à** nbe **faire**	3	
| ecrire (nom,prenom,moy)	3.5	nbe (entier) : nb étudiants
| Moyenne(noteM, noteF, noteA, moy)	3.3	nom, prenom (chaine)
| lire (nom,prenom)	3.1	moy(reel)
| lire (notea,notem,notef)	3.2	noteM, noteF, noteA (réel)
| sommoy ← sommoy + moy	3.4	sommoy(réel) : somme des moyennes
fpour		
sommmoy ← 0	2	cF(cste/reel = …) : coef français
lire(nbe)	1	
moygen ← sommoy/nbe	4	cM(cste/reel= …) : coef math
		cA(cste/reel = …) : coef français

Chapitre 4 Les tables à une dimension

Utilisation de ce type dans l'algo :

```
algo Calcul_moyenne                                  | Ord | Lexique
pour i ← 1 à Nbe faire                               |  3  | T_Tabmoy(type)=table[1..1000]reel
  |  ecrire (nom, prenom, moy)                       |     | nbe(entier) : nb etudiants
  |  Moyenne(noteM, noteF, noteA, moy)               |     | nom,prenom (chaine)
  |  sommoy ← sommoy + moy                          |     | Tabmoy(T_Tabmoy):table moyennes
  |  Tabmoy[i] ← moy                                |     | moy (réel)
  | + saisie des données                            |     | noteM, noteF, noteaA (réel)
fpour                                                |     | i (entier) : compteur
lire (Nbe)                                           |  1  | sommoy(réel) : somme des moyennes
sommoy ← 0                                           |  2  | Moyenne (module)
moygen ← sommoy/Nbe                                  |  4  | cpt (entier) : compte les
                                                     |     | moyennes > moyenne générale
cpt ← 0                                              |  5  | + cstes
pour i ← 1 à Nbe faire                               |  6  | …..
  |  si Tabmoy[i] > moygen alors
  |  |  cpt ← cpt+1
  | fsi
fpour
ecrire("Nombre de moyennes trouvées = ", cpt)        |  7  |
```

peut être remplacé par Tabmoy[i]

Chapitre 4 Les tables à une dimension

II Définition :

Une table à 1 dimension est une ***fonction*** définie sur un intervalle borné [a,b], pris dans un type **scalaire**, et qui prend ses valeurs dans un type *type_élément* quelconque, où ***a*** est la borne inférieure et ***b*** la borne supérieure de l'intervalle de définition

II.1 Représentation mathématique :

Soit une table T : [a,b] → *type_élément*

✓ A une position de la table on associe une valeur de type *type_élément.*

✓ (a,b) ∈ {(entier, entier), (caractère, caractère), (booléen, booléen)}

✓ type_élément ∈ {entier, caractère, réel, chaîne, booléen, table}

Chapitre 4 Les tables à une dimension

II.2 Définition dans le lexique

- **Syntaxe** : nom_type_table (**type**) = **table** [a**..**b] type_élément

- **Déclaration** : nom_variable1(nom_type_table)
  nom_variable2(nom_type_table)

Remarques :

- Une fois créé, le type table peut être utilisé indéfiniment
- Par convention, le nom d'un type créé commencera par T_.....

Chapitre 4 Les tables à une dimension

II.3 Quelques exemples

a) T_TabCodeAscii (**type**) = **table**['A'**..**'Z'] entier

TabCode(T_TabCodeAscii) : *variable de type table qui contient le code Ascii de chaque lettre majuscule*

'A'	'B'	'C'	….	….			'Y'	'Z'
65	66	67	…	…	…		89	90

Dans la partie définitions on peut faire les affectations suivantes :

TabCode['A'] ← 35

TabCode['B'] ← 36

Chapitre 4 Les tables à une dimension

b) On définit une table pour décrire les jours de la semaine :

n°jour ↔ libellé : on associe à un numéro de jour son nom

T_Jour(**type**) = **table**[1**..**7] chaine

TabJour (T_Jour) : variable *table contenant les noms de jours*

1	2	3	4	5	6	7
''Lundi''	''Mardi''	''Mercredi''	''Jeudi''	''Vendredi''	''Samedi''	''Dimanche''

Dans la partie définitions :

TabJour[3] ← "Mercredi"

Chapitre 4 Les tables à une dimension

c) On veut gérer des températures mensuelles sur 1 année

1ère solution : utiliser 12 variables tables correspondant aux 12 mois de l'année.

T_TabTempMois (**type**) = **table**[1**..**31] réel

TempJanvier(T_TabTempMois) : on y stocke les températures du mois de janvier

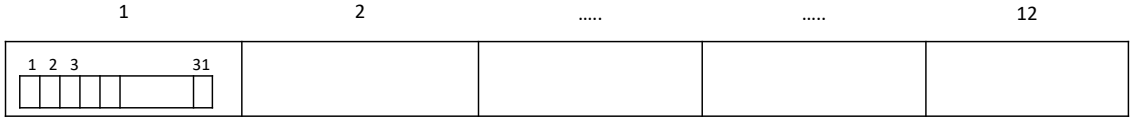
TempJanvier[25] *correpond à la température du 25 janvier*

TempFevrier(T_TabTempMois) : celles du mois de février

…

TempDecembre(T_TabTempMois) : celles du mois de décembre

2nde solution : utiliser une seule variable table pour toutes les températures annuelles.



T_TabTempAnnee (**type**) = **table** [1**..**12] T_TabTempMois

Dans le lexique :

TabTempAnnuelle(T_TabTempAnnee)

La température du 13ème jour du mois de mars : TabTempAnnuelle[3][13]

Remarque : T_TabTempAnnee est un type **table** de **tables** (de **réels**).

II.4 Définition du contenu d'une table

Le contenu d'une case est défini :

- ❑par affectation
- ❑par lecture

Exemples :

- T[i] ← valeur
- lire(T[i])
- T[i] ← T[i-1] + 2
- T[k] ← T[j] + b +2*T[k]

III Utilisation d'une table

Quand utiliser une table ?

Lorsqu'on a besoin d'établir une correspondance entre une valeur de type scalaire et un autre objet dont la valeur ne peut être calculée.

Exemple :

- (no_produit, prix) : la 1ère case va contenir le prix du premier produit, la 2ème le prix du deuxième, …
- (no_etudiant, note) : la 1ère case va contenir la note du premier étudiant, la 2ème la note du deuxième, …
- (lettre, codeAscii) : la case n° 'A' contient son code Ascii, etc ……

Comment utiliser une table ? (*accès aux éléments*)

Soient :

- ✓ TypeTab(**type**) = **table**[a**..**b] typeElement
- ✓ ***Tab***(TypeTab)
- ✓ ***i*** : indice de ***parcours*** des cases de la table **Tab**
- ✓ ***Nbelt*** : nombre de ***valeurs* effectives *stockées*** dans la table **Tab**
- ✓ ***id*** et ***if*** : respectivement indices de la 1ère et de la dernière valeur stockée dans la table

Chapitre 4 Les tables à une dimension

a. Accès direct

Tab[i] : nom de la table + n° de la case à accéder entre []

- *Conditions à vérifier :*

  ➢ si ***i*** ∉ [a,b] => Tab[i] est inaccessible.

On ne peut accéder à une case qui se trouve en dehors des limites physiques de la table.

  ➢ si ***i*** ∉ [id, if] où ***id*** est l'indice de la 1ère valeur stockée et ***if*** l'indice de la dernière valeur stockée => Tab[i] est indéterminée.

Chapitre 4 Les tables à une dimension

b. Accès successif à tous les éléments de la table

➢ Par une *itération* de l'indice de parcours de la table depuis ***id*** (indice de première valeur stockée) à ***if*** (indice de dernière valeur stockée)

et

➢ Par accès direct à chaque valeur

pour i ← id **à** if **faire**
| …Tab[i] …
| …
fpour

c. Accès associatif

L'accès associatif est utilisé pour :

❖Rechercher un élément, ayant une certaine propriété, dans une table

❖Représenter des couples de valeurs où la première n'est pas de type scalaire

Exemple : On veut associer à chaque étudiant sa moyenne => on va manipuler des couples de valeurs (***nom***, ***moyenne***).

Le nom est une chaîne de caractères, type non scalaire ⇒ On ne peut utiliser le nom comme indice

Solution : utiliser 2 tables pour stocker les informations.

T1 : {n° étudiant} → nom étudiant : table qui associe un numéro à un nom d'étudiant

T2 : {n° étudiant} → moyenne : table qui associe un numéro à une moyenne d'étudiant

Réalisation :

- Taille(**cste**/entier=150)
- T_Tabnom(**type**)=table[1..Taille] chaîne
- T_Tabmoy(**type**)=table[1..Taille] réel
- Tabmoy(T_Tabmoy)
- Tabnom(T_Tabnom)
- Nbe, nom

Exemple : Soit un étudiant, donné par son nom. On veut rechercher la moyenne de cet étudiant et l'afficher.

Principe : On parcourt la table des noms. On arrête le parcours lorsqu'on trouve le nom cherché ou lorsqu'on a parcouru tous les noms sans trouver.

NB : on suppose qu'il n'y a pas de problème d'homonymie.

Chapitre 4 Les tables à une dimension

Algorithme :

algo Affiche_Moy	Ord	Lexique
tq (i ≤ nbe) et non trouve faire	6	Taille (cste/entier = 150) : taille physique
| si Tabnom[i] = nom alors		T_Tabmoy (type)=table[1..Taille] reel
| | trouve ← vrai		T_Tabnoms (type)=table[1..Taille]chaine
| | moy ← Tabmoy[i]		nom (chaîne) : nom étudiant
| sinon		Tabnoms (T_Tabnoms)
| | i ←i+1		Tabmoys (T_Tabmoy)
| fsi		i (entier) : indice de parcours de la
ftq	3	table
lire(nom)	1	nbe (entier) : nombre d'étudiants
lire(nbe)	4	traités
i ←1	2	trouve (booleen)
Init_tab(nbe,Tabmoys, Tabnoms)	5	Init_tab (module) : saisie des valeurs
trouve ← faux	7	dans les 2 tables.
si trouve alors		
| ecrire("moyenne de ", nom," est = ", moy)		
sinon		
| ecrire(nom,"n'existe pas")		
fsi		

Chapitre 4 Les tables à une dimension

module Init_tab(↓n:entier; ↑TM:T_tabmoy; ↑TN:T_tabnoms)	Ord	Lexique
pour i ← 1 à n faire		i (entier) : indice de parcours de tables et compteur du nombre d'étudiants
|		
|ecrire("Saisir le nom et la moyenne de l'étudiant ",i)		
|lire(TN[i])		
|lire(TM[i])		
|		
fpour		

A l'appel du module Init_tab chaque **paramètre d'appel** (*paramètre effectif*) va remplacer le **paramètre formel** correspondant (par la position, le type, le mode, le sens).

Remarque : Autre solution ⇨Un module d'initialisation pour chaque table
module Init_tabNoms(↑T : T_tabnoms; ↓n : entier)
module Init_tabMoys(↑T : T_tabmoy; ↓n : entier)

Algorithme :

algo Affiche_Moy	Ord	Lexique
ecrire("Saisir le nombre d'étudiants") lire(nbe) Init_tab(nbe,Tabmoys, Tabnoms) ecrire("Saisir le nom de l'étudiant cherché") lire(nom) i ←1 trouve ← faux **tq** (i ≤ nbe) et non trouve **faire** | **si** Tabnom[i] = nom **alors** | | trouve ← vrai | | moy ← Tabmoy[i] | **sinon** | | i ← i+1 | **fsi** **ftq** **si** trouve **alors** | ecrire("moyenne de ", nom," est = ", moy) **sinon** | ecrire(nom,"n'existe pas") **fsi**		Taille (**cste**/entier = 150) : taille physique T_Tabmoy (**type**)=**table**[1..Taille] reel T_Tabnoms (**type**)=**table**[1..Taille]chaine nom (chaîne) : nom étudiant Tabnoms (T_Tabnoms) Tabmoys (T_Tabmoy) i (entier) : indice de parcours de la table nbe (entier) : nombre d'étudiants traités trouve (booleen) Init_tab (**module**) : saisie des valeurs dans les 2 tables.