(4) Exemple de variables structurées

On a vu des exemples où l'ordinateur devait gérer un grand nombre de valeurs (numériques ou autres). Un des intérêts fondamentaux du traitement informatique est de pouvoir manipuler une quantité importante de données, ce que ne permettent pas seules les variables simples. Nous allons donc être amenés a utiliser un nouveau type de variable : les tableaux (=variables indicées). Ce type de données est dit structuré car il est composé d'un assemblage de plusieurs données élémentaires. Un tableau est composé d'un nombre fixe de données du même type ; il existe d'autres types structurés qui sont composés de données de types différents (voir les structures RECORD) ou/et d'un nombre non prédéterminé de données (voir les pointeurs).

Il est aussi possible de définir de nouveaux types. Cette possibilité notament est utile dans la manipulation de tableau quand on veut que deux tableaux aient de même type sans avoir à recopier toute la déclaration (source d'erreurs). On peut donc définir des types d'objets plus proches de la réalité et abstrayant les types programmés.

1.) Variables à un indice

Exemp1e : on veut saisir puis conserver 10 valeurs réelles afin de les traiter (moyenne, écart-type).

```
program dixvaleurs ;
(* déclaration des variables *)
var T : array[1..10] of real ;
    i :  integer ;
    moyenne, variance  : real ;
BEGIN
  (* saisie des données *)
  writeln('donnez-moi 10 valeurs') ;
  for i:=1 to 10 do
    begin
    write(i,'-ième valeurs ? ') ;
    readln(T[i]) ;
    end ;
  (* calcul et affichage de la moyenne *)
  moyenne:=0 ;
  for i:=1 to 10 do moyenne:=moyenne+T[i] ;
  moyenne:=moyenne/10 ;
  writeln('moyenne des dix valeurs : ',moyenne) ;
  (* calcul et affichage de la variance *)
  (* ... *)
END.
```

Application : compléter ce programme pour calculer la variance. Var($\{x_i\}$)=Somme((x_i-moyenne)^2).

On a donc définit ici un nouveau type de données : un tableau, indicé de 1 à 10, de 10 réels. La variable T est donc en fait un groupe de 10 variables distinguées par leur indice. On retrouve ce mécanisme d'indiçage dans les suites : U1, U2, U3, ...Un, à la différence fondamentale que la suite a un indiçage potentiellement infini alors qu'une variable indicée contiendra toujours un nombre fini (et même prédéfini) de valeurs.

1.1.) déclaration

Le format de la déclaration est :

`ARRAY[`*début*`..`*fin*`] OF` *type* `;`

où *début* et *fin* sont des constantes entières telles que *début*<*fin* et *type* est un type de données quelconque (appelé [*type*] *support* du tableau). l'écriture *début*`..`*fin* représente un intervalle scalaire (En Pascal, on peut donc utiliser comme indices des caractères et commencer à n'importe quel indice).

Ce qu'on peut faire :

```
VAR T : array[-5..13] of integer ;

    U : array[-9..-2] of string[10] ;

    V : array['a'..'z'] of real ;


    W : array[1..10] of
            array[8..20] of real ;
      (* voir var. à 2 indices *)
```

Ce qu'on ne peut pas faire :

```
VAR   T : array[9..1] of integer ;
        (* debut<fin *)
U : array[0..1000] of string[255] ;
        (* >64kB en Turbo Pascal *)
V : array['oui'..'non'] of real ;
W : array[3.5 .. 3.6] of real ;
        (* indices non scalaires *)


X : array[1..'z'] of real ;
    (* types différents d'indices *)
Y : array[p..q] of real ;
 (* p et q sont des noms variables *)
```

1.2.) affectation et opérations

Ce qu'on peut faire :

```
V[?]:=T[?] ;
```

Ce qu'on ne peut normalement pas faire :

```
V:=T ;          (* quels que soient les tableaux T et U *)
T[0]:=U[0] ;    (* types de variables incompatibles *)
```

Il en va de même pour les opérations : elles sont celles autorisées par le "type support" du tableau.
On ne peut bien évidemment pas faire appel à un indice dépassant la limite du tableau. Par exemple, après la déclaration `var T : array[11..20] of integer ;` on a accès (lecture et écriture) aux 10 variables `T[11]` à `T[20]` mais pas aux variables `T[10]`, `T[21]`, ...

Exemple : on ne sait pas toujours à l'avance la longueur d'un tableau. Dans le cas suivant, on aura accès possible à T[1], ..., T[1000], même si on n'utilise qu'une partie du tableau (distinction entre déclaration et usage : "sémantisation").

```
PROGRAM nvaleurs ;
CONST taillemax = 1000 ;
VAR   T : array[1..taillemax] of real ;
      n  : integer ;
BEGIN
(* Déterminer la taille de T *)
write('Combien de valeurs ? ') ; readln(n) ;
if n>taillemax then
      begin
      n:=taillemax ;
      writeln('désolé, pas plus de ',taillemax,' valeurs !') ;
      end ;
(* saisir les valeurs du tableau *)
for i: =1 to n do
      begin
      write('? ') ;
      readln(T[i]) ;
      end ;
(* traitement sur T *)
...
END.
```

On peut ainsi gérer quasi-dyamiquement la taille du tableau : si la valeur 1000, ne suffit pas, on peut changer le programme source et le recompiler. Cependant, après compilation, on ne pourra pas dépasser cette taille maximale.

N.B. : la directive de compilation {$R+}, placée en début de partie exécutive, permet au compilateur d'ajouter au programme des contrôles de dépassement de bornes par les indices (mais cela ralentit le programme et doit donc surtout être utilisé pour la mise au point du programme).

1.3.) application : le tri dans un tableau de n valeurs

Première utilisation de boucles imbriquées.

Tri par recopie brutale dans un autre tableau (éventuellement)

Algorithme COPIERTRIER
Donnée : un tableau T de n valeurs distinctes (réelles par exemple).
Résultat : un tableau U contenant les mêmes valeurs triées par ordre décroissant.
Procédure : prendre le maximum de T le mettre en U[1]
prendre le "sous-maximum" de T et le mettre en U[2]
... et ainsi de suite jusqu'à U[n].

Programmation de la première étape :

```
max:=T[1] ;
for j:=2 to n do if max>T[j] then max:=T[j] ;
U[1]:=max ;
```

Programmation inductive du tri complet :

```
(* remplir U *)
for i:=1 to n do
    (* i-ième étape du remplissage *)
    begin
    max:=T[i] ;
    for j:=i+1 to n do
        if max<T[j] then max:=T[j] ;
    U[i]:=max ;
    end ;
```

Trace pour T=(1, 5, -10) : ...

Vérification de la procédure (effectivité et complexité) : ... $O(n^2)$.

Tri brutal avec un seul tableau
On peut procéder comme précédemment en déterminant l'indice i du maximum de T puis en échangeant T[1] et T[i] , etc. Plusieurs réalisations sont possibles. Le travail est moins brutal si on possède selon le principe du "tri à bulles".

Algorithme TRIERBRUTAL
Donnée : un tableau T de n valeurs distinctes (comparables par >).
Résultat : le tableau T trié par ordre décroissant.
Procédure :
parcourir T pour localiser le maximumT[j], puis placer ce maximum en T[1] (T[1] vient en T[j] en échange)
recommencer pour placer T[2]
etc.

Algorithme TRIABULLE
Donnée : un tableau T de n valeurs distinctes (comparables par >).
Résultat : le tableau T trié par ordre décroissant.
Procédure :
pour placer le maximum en T[1] (placé en bas de la colonne) :
prendre T[1] et le "remonter" la bulle dans le tableau jusqu'à trouver t[j]<T[1],
échanger les deux valeurs et "remonter" la nouvelle bulle T[j] jusqu'à trouver ...etc. jusqu'à j=n.
placer de même T[2], T[3]..., T[n-1].

Programmation :

```
for i:=1 to n-1 do
  for j:=i+1 to n do
    if T[i]<T[j] then
      begin
      tmp:=T[i] ;
      T[i]:=T[j] ;
      T[j]:=tmp ;
      end ;
```

Exemple :

8	5
7	3
6	6
5	12
4	7
3	1
2	4
1	2

8	5
7	3
6	6
5	7
4	4
3	1
2	2
1	12

8	5
7	3
6	6
5	4
4	2
3	1
2	7
1	12

8	5
7	3
6	4
5	2
4	1
3	6
2	7
1	12

8	4
7	3
6	2
5	1
4	5
3	6
2	7
1	12

8	3
7	2
6	1
5	4
4	5
3	6
2	7
1	12

8	2
7	1
6	3
5	4
4	5
3	6
2	7
1	12

8	1
7	2
6	3
5	4
4	5
3	6
2	7
1	12

Un exemple de tri à bulles (odre décroissant)

Exercices

1)
Construire un programme qui saisit les coordonnées de deux vecterus de l'espace puis calcule :
a) leur somme.
b) leur produit scalaire. Scal((x1,y1,z1),(x2,y2,z2))=x1x2+y1y2+z1z2.

2)
Construire un programme qui écrit dans un tableau les n premières valeurs de la suite (Un).
a) (Un) définie par Uo=2 et U(n+1)=SQRT(U(n)^2-1), pour n=10.
b) (Un) définie par Uo='ba' et U(n+1)=Un+Un, pour n=6. Quels problèmes rencontrera-t-on pour n=100 ?

3)
Ecrire un programme qui décale les valeurs d'un tableau de deux cases vers la gauche :
a) en effaçant les premières valeurs.
b) en replaçant les premières valeurs en fin de tableau, dans le même ordre (permutation circulaire).

4) Ecrire un programme qui vérifie si un tableau est "palindromique". (N.B. réutilisation de programme.)

2.) Variables à deux indices

On a vu qu'on pouvait créer un tableau de tableau :

```
array[m..n] of arry[p..q] of ...
```

Ce type de variable correspond en fait à une variable à deux indices.

Exemple : Relever les notes en maths, physique et informatique de 40 élèves de S1, puis calculer leurs moyennes simples.

```
PROGRAM tableaudenotes ;
CONST max=40 ;
VAR   T : array[1..max,1..max] of real ;
      nom : array[1..max] of string[20] ;
      i, j : integer ;
      moyenne : array[1..max] of real ;
BEGIN
(* pour chaque étudiant *)
for i:=1 to max do
      begin
      write('nom de l'étudiant ? ') ; readln(nom[i]) ;
      (* note de maths *)
      write('note de maths ? ') ; readln(T[i,1]) ;
      (* note de physique *)
      write('note de maths ? ') ; readln(T[i,2]) ;
      (* note d'informatique *)
      write('note d'informatique ? ') ; readln(T[i,3]) ;
      end ;
...
for i:=1 to max do
      begin
      moyenne[i]:=0 ;
      for j:=1 to 3 do moyenne[i]:=moyenne[i]+T[i,j] ;
      end ;
...
for i:=1 to max do writeln('nom : ', nom[i], ' moyenne : ',moyenne[i]) ;
END.
```

Attention à la présentation d'une variable à deux indices : premier et deuxième indice pour lignes et colonnes (deux conventions sont possibles). Par exemple, pour le tableau T ci-dessus, lignes=n°étudiant et colonnes=note ou bien l'inverse. Un choix est donc à faire au moins pour l'affichage.

Exercice :
a) Ecrire un programme qui détermine la plus grande valeur contenue dans une matrice 10x10.
b) Ecrire un programme qui saisit les valeurs de deux matrices carrées 10x10, puis calcule leur somme.

2.1.) déclaration

pour un tableau à deux indices,

le format de la déclaration est :	ce qui est équivalent à :
nom : ARRAY[m..n,p..q] OF *type* ;	*nom* : ARRAY[m..n] OF ARRAY[p..q] OF *type* ;

On peut, bien évidemment, créer des tableaux à trois indices ou plus.

2.2.) (macro-)affectation et opérations

a) affectations globales (hors programme)

Contrairement à ce qui a été écrit précédemment, on peut affecter directement deux tableaux de même type :
déclaration : T,U : array[1..10] of string ;

affectation : T:=U ;
Cette affectation revient à écrire : for i:=1 to 10 do T[i]:=U[i] ;
Attention : ceci n'est possible que si les deux variables ont même type : même support, mêmes indices.

b) affectation semi-globales (hors programme)

Comme un tableau à plusieurs indices est considéré comme un tableau de tableau, on peut écrire :
déclaration : A : array[1..3,10..19,1..40] or real ; B : array[10..19,1..40] of real ;
affectation : A[1]:=B ; (ou B:=A[1] ;)
Aux mêmes conditions que précédemment.

On a donc intérêt à prédéfinir de nouveaux types sur le modèle donné dans le paragraphe suivant.

Les opérations sont les mêmes que pour les variables à un indice et dépendent du type support.

3.) Déclaration de types

Entre la rubrique CONST et la rubrique VAR, on peut placer une rubrique TYPE définissant de nouveaux types.

Exemple :

```
CONST taille=40 ;
TYPE etudiant = string[20] ;
     S1 = array[1..taille] of etudiant ;
     note : 0..20 ;
     lettre : 'a'..'z' ;
VAR groupe1 : S1 ;
    Nom : etudiant ;
    n :note ;
...
```

Le modèle est toujours le même : [TYPE [NomduNouveauType = DéclarationduType ;]+]?

Conseils : il ne s'agit pas d'utiliser à tout propos les déclarations de types mais plutôt de simplifier et d'alléger la tâche de programmation selon les besoins : abstraction des données, correspondance de types entre variables.

Exercices

1)
Contruire un tableau correspondant à la table de multiplication des nombres de 1 à 10.

2)
Construire un programme qui transpose un tableau à deux indices 10x10 de réels en un tableau à un indice (1..100).
Quel choix fait : k=10(i-1)+j ou k=i+10(j-1) ?

3)
Construire un programme qui remplit une matrice carrée 15x15 de caractères, en plaçant "0" sur l'interdiagnonale, "+" sur les reste des diagonales et "-" sur le reste du tableau. Copier ce tableau dans un tableau de 15 chaînes de 15 caractères. Afficher la matrice puis le tableau.