Les tableaux

Pour représenter un grand nombre de données devant toutes rester directement accessibles, on ne peut se servir d'une variable scalaire puisque celle-ci ne peut donner accès qu'à une seule valeur à la fois. On ne peut songer à attribuer une variable à chaque donnée car cela pourrait devenir vite fastidieux aussi bien pour l'écriture que pour l'exécution du programme.
Dans la plupart des langages de programmation, le type *tableau* joue ce rôle. Une variable pourra représenter plusieurs données à la fois qui pourront être manipulées collectivement ou individuellement.

Les tableaux à une dimension

Les *tableaux monodimentionnels* ou *tableaux à une dimension* ou encore à *un indice* (qui seront désignés en fin d'année sous le terme de vecteurs) sont constitués d'*éléments* et d'*indices*.
Il faut donc définir :

- le type de l'élément de base du tableau (ce que contient le tableau)
- le type de l'indice (ce qui nous permet de choisir l'élément qui nous intéresse).

Les indices sont des entiers naturels allant de 1 à *N* si le tableau comporte *N* éléments. Contrairement à certains langages compilés (comme le Turbo-Pascal), ils peuvent être créés dynamiquement (il n'est pas utile de préciser dès le début leur taille).

Création ; plusieurs modes sont envisageables :

- Tableau défini par la liste de ses éléments ; il faut placer cette liste d'éléments, séparés par des virgules, entre crochets :

```
-->v=[2,3,5,9]
 v  =
!  2.    3.    5.    9. !
```

- Tableau dont les éléments constituent une progression arithmétique ; il faut utiliser la notation *n*:*m* ou *n*:*p*:*m* (où *n* est la valeur initiale, *p* le pas d'incrémentation et *m* la valeur finale à ne pas dépasser)

```
-->3:8
 ans  =
!  3.    4.    5.    6.    7.    8. !

-->a=12:-3:-4
 a  =
!  12.    9.    6.    3.    0.  - 3. !
```

- tableaux particuliers

Tableau dont tous les éléments sont nuls : fonction *zeros*

```
-->v=zeros(1,7)          //permet de créer un tableau à une dimension de taille 7, d'éléments nuls
 v  =
!  0.    0.    0.    0.    0.    0.    0. !
```

Tableau vide : [] ; il est défini :
- en écrivant []
- par la commande 3:1 (les entiers à partir de 3 avec un pas de 1 et inférieurs à 1)
- zeros(1,0)

Accès aux éléments

Si temp est une variable de type tableau de 7 entiers correspondant aux températures journalières d'une semaine donnée, les indices possibles sont 1, 2, 3, 4, 5, 6, 7. Les éléments de ce tableau seront des entiers notés temp(1), temp(2), temp(3), temp(4), temp(5), temp(6), temp(7). On peut représenter cette variable sous la forme :

	1	2	3	4	5	6	7
temp	8	6	7	11	6	8	9

L'expression placée entre parenthèse peut être quelconque, à condition qu'elle fournisse un résultat entier compris entre 1 et la taille du vecteur. L'accès à un élément du tableau peut, si k est un entier, se faire par :

Si k est initialisé à 5 temp(k)=6
temp(k+1)=8
temp(k-1)=11

Le calcul de la moyenne des températures de la semaine est :

```
-->somme=0;
-->for k=1:7 do somme=somme+temp(k);
-->end
-->moyenne=somme/7
 moyenne  =
   7.8571429
```

Les fonctions vectorielles

La fonction length fournit le nombre d'éléments d'un tableau

```
-->length([1,2,3,5,8,12])
 ans  =
   6.
-->length(3:2:197)
 ans  =
   98.
```

La variable $ permet de désigner le dernier élément d'un tableau à une dimension :

```
-->v=[2,7,1,-1,2,9];
-->v($)
 ans  =
   9.
-->v(length(v))
 ans  =
   9.
```

La fonction isempty() appliquée à un tableau permet d'indiquer si celui-ci est vide ou non :

```
-->isempty(1:10)
 ans  =
  F
```

```
-->isempty([])
 ans  =
  T
-->isempty(2:-1:8)
 ans  =
  T
```

Tableaux à deux dimensions

La déclaration se fait comme pour les tableaux à une dimension ; les lignes sont séparées entre elles par des points virgules :

```
-->A=[1,5,6,7;5,9,8,4]
 A  =
!  1.    5.    6.    7. !
!  5.    9.    8.    4. !
```

L'opération la plus simple pour l'accès aux éléments d'un tableau à deux dimensions est *M*(*n*,*p*) : cette opération lit l'élément du tableau situé à la ligne *n* et à la colonne *p*.

```
-->M=rand(3,4)      //tableau de taille (3,4) dont les éléments sont des réels aléatoires de [0;1]
 M  =
!   .8833888    .9329616    .3616361    .4826472 !
!   .6525135    .2146008    .2922267    .3321719 !
!   .3076091    .312642     .5664249    .5935095 !
-->M(2,3)
 ans  =
    .2922267
```

La variable $ permet alors de repérer les éléments d'indice maximal en ligne ou en colonne ; en reprenant l'exemple ci-dessus :

```
-->M(1,$)                -->M($,3)                -->M($,$)
 ans  =                   ans  =                   ans  =
    .4826472                 .5664249                 .5935095
```

La fonction size() permet de fournir les dimensions d'un tableau donné en paramètre.

```
-->size([1,2,5,8,14,-9])
 ans  =
!  1.    6. !
-->size([1,1,2;4,-9,-8;2,3,5])
 ans  =
!  3.    3. !
```

L'extraction de sous tableau

L'instruction matrix(*A*,*n*,*p*) permet de restructurer le tableau *A* selon les dimensions *n* et *p*.

```
-->M=matrix(1:2:29,3,5)
 M  =
!  1.    7.    13.   19.   25. !
!  3.    9.    15.   21.   27. !
!  5.    11.   17.   23.   29. !
```

Nous pouvons alors extraire de ce tableau, par exemple les colonnes 1,4 et 5 des lignes 2 et 3 :

```
-->N=M([2,3],[1,4,5])
 N  =
!   3.    21.    27. !
!   5.    23.    29. !
```

Nous pouvons également extraire directement un sous-tableau :

```
-->P=M(2:3,3:5)
 P  =
!   15.    21.    27. !
!   17.    23.    29. !
```

Les fonctions vectorielles

Si vous utilisez les fonctions suivantes vous devez être capable de donner un algorithme permettant de les construire.

max	maximum
min	minimum
sort	tri par ordre croissant
gsort	tri, ordres particuliers
sum	somme
prod	produit
cumsum	sommes cumulées
cumprod	produits cumulés
mean	moyenne
median	médiane
st-deviation	écart type

Exercices

Réversion d'un vecteur

```
function [v]=reversion(u)
// miroir d'un vecteur
v=u
n=length(v);l=int(n/2)
for i=1:l do
j=n-i+1
aux=v(i)
v(i)=v(j)
v(j)=aux
end
```

Dans Scilab :

```
-->reversion([1,2,3,4])
 ans  =
!   4.    3.    2.    1. !
```

Rotation d'un vecteur

Ecrire un programme permettant de réaliser la rotation d'un vecteur c'est-à-dire la permatation circulaire de ses éléments.

```
function [v]=rotation(u,d)
v=u
n=length(v)
for i=1:n do
j=modulo(i+d-1,n)+1
v(i)=u(j)
end
```

```
-->rotation([1,2,3,4,5,6,7],2)
 ans  =
!   3.    4.    5.    6.    7.    1.    2. !
```

Exercice : recherche d'une valeur dans un tableau

Ecrire un programme qui détermine si un tableau de taille 100 contenant des entiers aléatoires entre 0 et 999 contient un nombre entier donné et donne les indices correspondants.

```
function recherche
for indice=1:100 do t(indice)=floor(1000*rand()); disp(t(indice)); end
n=input("Quelle est la valeur à rechercher ? ");
for indice=1:100 do if t(indice)==n then disp(n,' est égal à ',')',indice,'t('); end; end
```

Initialisation et lecture d'un tableau à 2 dimensions

```
n=input('Combien de lignes ? ')
p=input('Combien de colonnes ?')
for i=1:n do
for j=1:p do
disp(j,' et ',i,'Donner l element d indices ')
t(i,j)=input('')
end
end
disp(t)
```

Traingle de Pascal

Ecrire une fonction Pascal(N) fournissant un tableau contenant le triangle de Pascal d'ordre N.

```
t(1,1)=1;for j=2:n do t(1,j)=0; end
for i=2:n do t(i,1)=1;
for j=2:i do t(i,j)=t(i-1,j-1)+t(i-1,j);end
for j=i+1:n do t(i,j)=0; end
end
disp(t)
```

Notes

Un tableau t contient les notes (de 0 à 20) de NE élèves à une série de ND devoirs.
Chaque élève a eu une note à chaque devoir.
On demande de calculer les distributions marginales, c'est-à-dire de compléter le tableaux en donnant la moyenne par élève, la moyenne par devoir et la moyenne globale.

```
n=input('Combien de d''élèves ? ')
p=input('Combien de de devoirs ?')
for i=1:n do
for j=1:p do
disp(j,' et ',i,'Donner l''element d''indices ')
t(i,j)=input('')
end
end
disp(t)

for i=1:n do t(i,p+1)=0; end
for j=1:p do t(n+1,j)=0; end
disp(t)

for i=1:n do
somme=0;
for j=1:p do somme=somme+t(i,j); end
t(i,p+2)=somme/p;
end

for j=1:p do
somme=0;
for i=1:n do somme=somme+t(i,j); end
t(n+2,j)=somme/n;
end

somme=0;
for i=1:n do somme=somme+t(i,p+2); end
t(n+2,p+2)=somme/n;

disp(t)
```