CHAPITRE I

Les variables indicées

Chapitre I : Les variables indicées

I.1. Objectif de ce chapitre

A l'issu de ce chapitre, l'apprenant sera capable de :

- Définir ce qu'est un tableau et expliquer son utilité en informatique.
- Déclarer, initialiser et manipuler les éléments d'un tableau dans un programme.
- Effectuer des opérations courantes : recherche, tri et calculs sur les éléments d'un tableau.
- Comprendre et utiliser des tableaux multidimensionnels (matrices).
- Appliquer les tableaux pour résoudre des problèmes concrets en programmation.

I.2. Introduction

Imaginons que dans un programme, nous avons besoin d'un grand nombre de variables, il devient difficile de donner un nom à chaque variable.

Exemple :

Ecrire un algorithme permettant de saisir cinq notes et de les afficher après avoir multiplié toutes les notes par trois.

Solution :

Algorithme Note ;
Variables
N1, N2, N3, N4, N5 : réel ;
Début
Ecrire ('Enter la 1ère Note') ;
Lire(N1) ;
Ecrire ('Enter la 2ème Note') ;
Lire(N2) ;
Ecrire ('Enter la 3ème Note') ;
Lire(N3) ;
Ecrire ('Enter la 4ème Note') ;

A. OUARET

```
Lire(N4) ;
Ecrire ('Enter la 5ème Note') ;
Lire(N5) ;
Ecrire('La note 1 multipliée par 3 est : ',N1*3) ;
Ecrire('La note 2 multipliée par 3 est : ',N2*3) ;
Ecrire('La note 3 multipliée par 3 est : ',N3*3) ;
Ecrire('La note 4 multipliée par 3 est : ',N4*3) ;
Ecrire('La note 5 multipliée par 3 est : ',N5*3) ;
```

Fin.

La même instruction se répète cinq fois. Imaginons que si l'on voudrait réaliser cet algorithme avec 100 notes, cela deviendrait fastidieux.

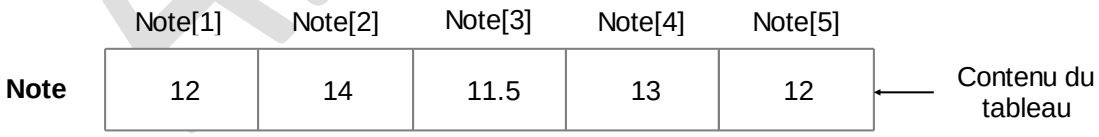
Comme les variables ont des noms différents, on ne peut pas utiliser de boucle, ce qui allonge considérablement le code et le rend très répétitif.

Pour résoudre ce problème, il existe un type de données qui permet de définir plusieurs variables de même type.

I.3. Définition

Un tableau est une suite d'éléments de même type, Il utilise plusieurs cases mémoire à l'aide d'un seul nom. Comme toutes les cases portent le même nom, elles se différencient par un numéro ou un indice.

Nous pouvons représenter schématiquement un tableau nommé Note composé de cinq cases, dans la mémoire comme suit :



Nous disposons alors de cinq variables. Pour les nommer, on indique le nom du tableau suivi de son indice entre crochets ou entre parenthèses : La première s'appelle Note[1], la deuxième Note[2], etc. jusqu'à la dernière Note[5].

Note[4] représente le 4ème élément du tableau **Note** et vaut 13.

A.OUARET

I.4. Tableau à une dimension

I.4.1. Déclaration

La déclaration d'un tableau permet d'associer à un nom une zone mémoire composée d'un certain nombre de cases mémoires de même type.

Syntaxe

Algorithme

```
Variable
Variable identificateur : tableau [Indice_min .. Indice_max] de type ;
```

Pascal

```
Var
Variable identificateur : array [Indice_min .. Indice_max] of type ;
```

Ou bien

```
Algorithme
Constante
MAX = Indice_max ;
Variable
Variable identificateur : tableau [1..MAX] de type;

Pascal
Const
MAX = Indice_max ;
Var
Variable identificateur : Array [1..MAX] of type ;
```

Remarques ☺ :

- Le premier élément d'un tableau porte l'indice zéro ou l'indice 1 selon les langages (1 en langage Pascal).
- La valeur d'un indice doit être un nombre entier.
- La valeur d'un indice doit être inférieure ou égale au nombre d'éléments du tableau. Par exemple, avec le tableau tab[1..20], il est impossible d'écrire tab[0] et tab[21]. Ces expressions font référence à des éléments qui n'existent pas.
- C'est pas obligatoire d'utiliser une constante pour la taille maximale de tableau ; cependant, c'est une bonne pratique dans la programmation

Exemple :

L'instruction suivante déclare un tableau de 30 éléments de type réel :

Algorithme	Pascal
Variable Note : tableau [1..30] de réels ;	**Var** Note : array [1..30] of real ;

Note : c'est le nom du tableau (identificateur).

1 : c'est l'indice du premier élément du tableau.

30 : c'est l'indice du dernier élément du tableau (nombre d'éléments du tableau).

Exercice :

Déclarer deux tableaux nommés A et B composé chacun d'eux de 10 éléments de type chaine.

Solution :

Algorithme	Pascal
Variable A, B : tableau [1..10] de chaine ;	**Var** A, B : array [1..10] of string ;

Remarques ☺ :

- Lorsqu'on déclare un tableau, on déclare aussi de façon implicite toutes les variables indicées qui le constituent. Nous utiliserons souvent la valeur 1 comme premier indice (Pascal) mais on peut aussi utiliser un autre indice minimum, comme 0. Dans ce cas, l'indice maximum sera égal au nombre d'éléments -1.
- Dans le langage C, les cases du tableau sont numérotées à partir de 0. En Visual Basic l'indice minimum est 1.

Exemple 1 :

L'instruction suivante affecte à la variable X la valeur du premier élément du tableau Note :

Algorithme	Pascal
X ← Note [1] ;	X := Note [1] ;

Exemple 2 :

L'instruction suivante affiche le contenu du quatrième élément du tableau Note :

Algorithme	Pascal
Ecrire(Note[4]) ;	write(Note[4]) ;

A.OUARET

Exemple 3 :

L'instruction suivante affecte une valeur introduite par l'utilisateur à l'élément trois du tableau Note :

Algorithme	Pascal
Lire(Note[3]) ;	read(Note[3]) ;

I.4.2. Initialisation

Un tableau peut être initialisé en utilisant deux méthodes :

- En utilisant des affectations ;
- En utilisant la lecture à partir d'un fichier de données ou à partir de clavier.

a) Par affectations : On utilise pour cela l'opération d'affectation.

Exemple :

V[1] := 24.00; *V[2] := 38.25*; *V[3] := 28.00*; *V[4] := 70.25*; *V[5] := 63.00* ; *V[6] := 96.25*;

b) Par lecture : On utilise pour cela l'opération de lecture READ : C'est la méthode la plus commode.

Exemple1 :

Exemple d'algorithme / programme Pascal permettant de lire et d'écrire (afficher) le tableau précédent :

Algorithme	Pascal

algorithme LireEcrireTableau
variables
v:tableau[1..6] de reel
i:entier
Début
ecrire('Introduire V :')
pour i:=1 **à** 6 **faire**
lire(v[i])
finPour
ecrire('Affichage V :')
pour i:=1 **à** 6 **faire**
ecrire(v[i])
finPour
Fin
	program LireEcrireTableau;
	var
	V:**Array**[1..6] **of** real;
	i:integer;
	Begin
	writeln('Introduire V : ');
	for i:=1 **to** 6 **do**
	Read (V[I]);
	
	writeln ('Affichage V : ') ;
	for i:=1 **to** 6 **do**
	Write (V[I]);
	End.

A. OUARET

I.4.3. Manipulation des tableaux à une dimension (vecteurs)

Les tableaux ont une grande importance en informatique, la quasi-totalité des problèmes utilisent des structures de données sous forme de tableaux. On peut en citer le domaine du calcul matriciel, les statistiques, les traitement de gestion, *etc.* La gestion et l'analyse de données nécessitent l'organisation des données dans des tableaux pour rendre possible leur traitement informatique.

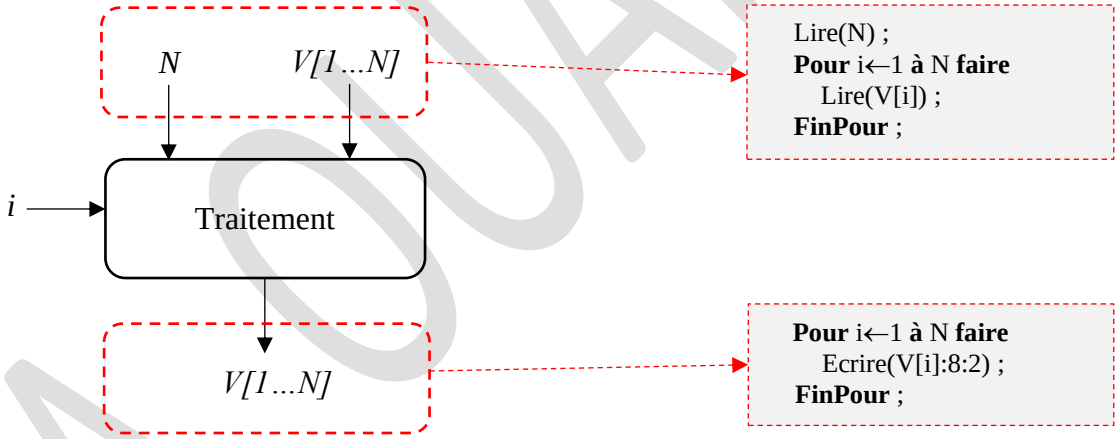
I.5. Exercices avec corrigés

Exercice N°01 : Lecture et Affichage d'un vecteur

Ecrire un algorithme/programme PASCAL qui permet de lire et afficher un vecteur V de n composantes réelles.

Corrigé de l'exercice N°01 :

Les variables d'entrée, variable de sorite et la partie traitement sont présentées sur le schéma ci-dessous:



Remarques ☺

- Il faut noter que ce programme ne réalise pas de traitement, il ne contient que des entrées et des sorties.
- La variable ***i*** est une variable de traitement ou intermédiaire, utilisée pour parcourir le vecteur T.

Algorithme	Pascal
Algorithme Affichage_Vecteur ; **Variables** V : **Tableau** [1..100] **de réel** ;	**Program** Affichage_Vecteur ; **Var** V : **array** [1..100] **of real** ;

A.OUARET

```
   i, N : entier ;
Début
   {-*-*-*- Entrées -*-*-*-}
   Ecrire('Donner la taille du vecteur V : ') ;
   Lire(N) ;
   Ecrire('Donner les composantes du vecteur
           V : ') ;
   Pour i←1 à N faire
      Lire(V[i]) ;
   FinPour ;
   {-*-*-*- Sorties -*-*-*-}
   Ecrire('Affichage du vecteur V : ') ;
   Pour i←1 à N faire
      Ecrire(V[i]:8:2) ;
   FinPour ;
Fin.
```

```
   i, N : integer ;
Begin
   {-*-*-*- Entrées -*-*-*-}
   Write('Donner la taille du vecteur V : ') ;
   Read(N) ;
   Writeln('Donner les composantes du vecteur
           V : ') ;
   For i :=1 to N do
      Read(V[i]) ;

   {-*-*-*- Sorties -*-*-*-}
   Writeln('Affichage du vecteur V : ') ;
   For i :=1 to N do
      Write(V[i]:8:2) ;

End.
```

```
1  Program Affichage_Vecteur ;
2  Var
3     V : array [1..100] of real ;
4     i, N : integer ;
5  Begin
6     {-*-*-*- Entrées -*-*-*-}
7     Write('Donner la taille du vecteur V : ') ;
8     Read(N) ;
9     Writeln('Donner les composantes du vecteur V : ') ;
10    For i :=1 to N do
11       Read(V[i]) ;
12
13    {-*-*-*- Sorties -*-*-*-}
14    Writeln('Affichage du vecteur V : ') ;
15    For i :=1 to N do
16       Write(V[i]:8:2) ;
17
18 End.
```

```
MyPascal V1.20.5 (Exécution) C:\Users\Ah...
Donner la taille du vecteur V : 5
Donner les composantes du vecteur V :
22 1.5 -7 9 5
Affichage du vecteur V :
   22.00    1.50   -7.00    9.00    5.00
```

Après l'exécution

Explication ☺

Ce programme montre comment lire et écrire un vecteur. Pour les deux opérations (lecture et écriture), nous aurons toujours besoin d'une boucle **Pour**. Lors de la déclaration, nous réservons 100 cases réelles (la taille maximale du vecteur), et nous utilisons la variable **n** pour déterminer la taille que nous voulons utiliser (par exemple 5 cases). L'accès à une composante d'indice **i** se fait comme suit : **V[i]**. Ainsi, pour lire la case 2, nous écrivons **Lire(V[2])**, et **Ecrire(T[4])** pour afficher la valeur de la 4ème case.

Exercice N°02 :

Ecrire un algorithme/programme Pascal permettant de saisir 10 notes et de les afficher après avoir multiplié toutes ces notes par un coefficient fourni par l'utilisateur :

Corrigé de l'exercice N°02 :

Algorithme	Pascal

Algorithme tableau_note ;
Variables
Note : tableau [1..10] de réels ;
Coef, i : entier ;
Début
Ecrire('Entrer le coefficient : ') ;
Lire(Coef) ;
{Remplissage du tableau Note}
Pour i ← 1 **à** 10 **faire**
Ecrire('Entrer la valeur de la note ', i,' : ') ;
Lire(Note[i]) ;
Fin-Pour ;
Ecrire('Affichage des notes*Coef : ') ;
Pour i ← 1 **à** 10 **faire**
Ecrire(Note[i]*coef :8:2) ;
Fin-Pour ;
Fin.
	Program tableau_note ;
	Var
	Note : array [1..10] of real ;
	Coef, i : integer ;
	Begin
	write('Entrer le coefficient : ') ;
	read(Coef) ;
	{Remplissage du tableau Note}
	For i := 1 **to** 10 **do**
	Begin
	write('Entrer la valeur de la note ', i,' : ') ;
	read(Note[i]) ;
	End ;
	Write('Affichage des notes*Coef : ') ;
	For i := 1 **to** 10 **do**
	write(Note[i]*coef :8:2) ;
	End.

```
1  Program tableau_note ;
2  Var
3    Note : array [1..10] of real ;
4    Coef, i : integer ;
5  Begin
6    write('Entrer le coefficient : ') ;
7    read(Coef) ;
8    {Remplissage du tableau Note}
9    For i := 1 to 10 do
10     Begin
11       write('Entrer la valeur de la note ', i,' : ') ;
12       read(Note[i]) ;
13     End ;
14   Write('Affichage des notes*Coef : ') ;
15   For i := 1 to 10 do
16     write(Note[i]*coef :8:2) ;
17 End.
```

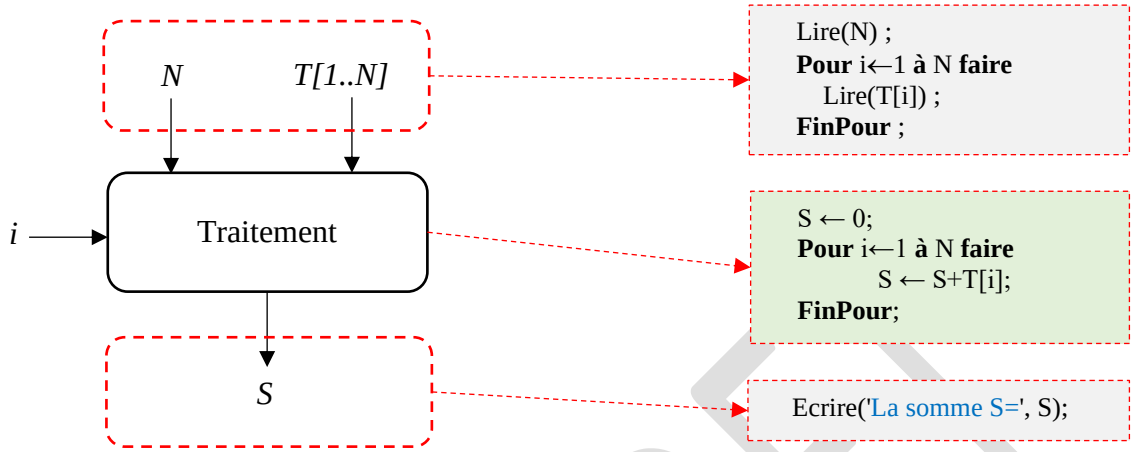
```
MyPascal V1.20.5 (Exécution)
Entrer le coefficient : 3
Entrer la valeur de la note 1 : 12
Entrer la valeur de la note 2 : 4
Entrer la valeur de la note 3 : 5
Entrer la valeur de la note 4 : 5
Entrer la valeur de la note 5 : 7
Entrer la valeur de la note 6 : 7
Entrer la valeur de la note 7 : 5
Entrer la valeur de la note 8 : 3
Entrer la valeur de la note 9 : 9
Entrer la valeur de la note 10 : 6
Affichage des notes*Coef :    36.00   12.00   15.00   15.00
   21.00   21.00   15.00    9.00   27.00   18.00
```

Exercice N°03 :

Ecrire un algorithme/programme Pascal qui calcul la somme des éléments d'un tableau.

Corrigé de l'exercice N°03 :

Les variables d'entrée, variable de sorite et la partie traitement sont présentées dans le schéma ci-dessous :



On suppose que le nombre d'éléments du tableau ne dépasse pas 100.

Algorithme	Pascal

Algorithme Somme_tableau ;
Constante M=100 ;
Variable T : tableau [1..M] de réels ;
N, i : entier ;
S : réel ;
Début
Ecrire('Entrer la taille du tableau : ') ;
Lire(N) ;
Si (N>M) **alors**
N ← M ;
Fin-Si ;
Ecrire('Remplissage du tableau : ') ;
Pour i ← 1 **à** N **faire**
Lire(T[i]) ;
Fin-Pour ;
S ← 0 ;
Pour i ← 1 **à** N **faire**
S ← S+T[i] ;
Fin-Pour ;
Ecrire('La somme des éléments du tableau est : ', S:10:2) ;
Fin.
	Program Somme_tableau ;
	Const M=100 ;
	Var T : array [1..M] of real ;
	N, i : integer ;
	S : real ;
	Begin
	write('Entrer la taille du tableau : ') ;
	read(N) ;
	if (N>M) **then**
	N := M ;
	
	write('Remplissage du tableau : ') ;
	For i := 1 **to** N **do**
	read(T[i]) ;
	
	S := 0 ;
	For i := 1 **to** N **do**
	S := S+T[i] ;
	
	write('La somme des éléments du tableau est : ', S:10:2) ;
	End.

A.OUARET

```
1  Program Somme_tableau ;
2  Const M=100 ;
3  Var T : array [1..M] of real ;
4          N, i : integer ;
5          S : real ;
6  Begin
7    write('Entrer la taille du tableau : ') ;
8    read(N) ;
9    if (N>M) then
10     N := M ;
11
12   write('Remplissage du tableau : ') ;
13   For i := 1 to N do
14     read(T[i]) ;
15
16   S := 0 ;
17   For i := 1 to N do
18     S := S+T[i] ;
19
20   write('La somme des éléments du tableau est : ', S:10:2) ;
21 End.
```

```
MyPascal V1.20.5 (Exécution)
Entrer la taille du tableau : 5
Remplissage du tableau : 2 6 5 0.8 5
La somme des éléments du tableau est :      18.80
```

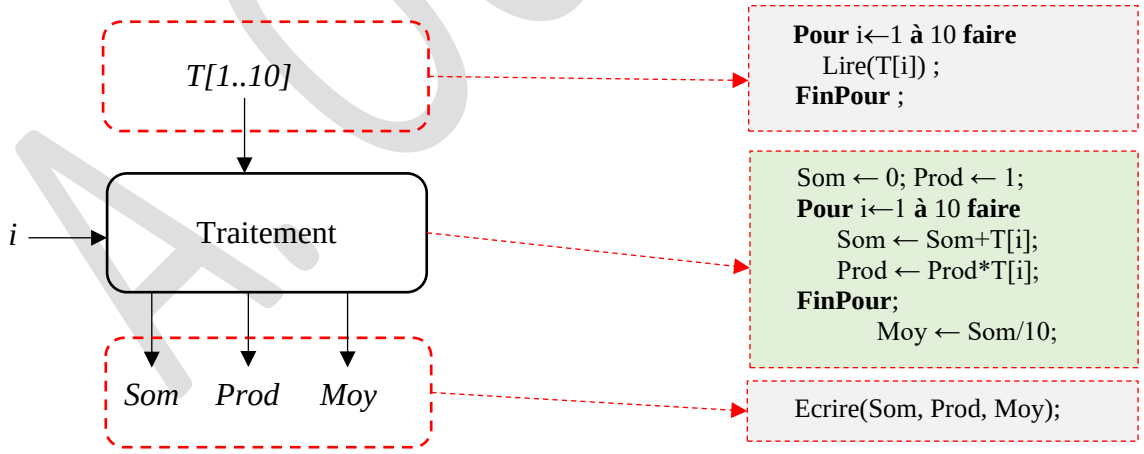
Après l'exécution

Exercice N°04 : (Calcul de la somme, le produit et la moyenne des éléments numériques d'un tableau)

Ecrire un algorithme permettant de calculer la somme, le produit et la moyenne d'un tableau de dix réels. Traduire l'algorithme en Pascal.

Corrigé de l'exercice N°04 :

Les variables d'entrée, variable de sorite et la partie traitement sont présentées dans le schéma ci-dessous :



Algorithme	Pascal
Algorithme Som_Moy_Prod_Tab ; **Variables** T : tableau [1..10] de réel ; i : entier ; Som, Prod, Moy : réel ;	**Program** Som_Moy_Prod_Tab ; **Var** T : array [1..10] of real ; i : integer ; Som, Prod, Moy : real ;

Début	Begin

Ecrire('Donnez les éléments du tableau : ') ;
Pour i ← 1 **à** 10 **faire**
Lire(T[i]) ;
Fin-Pour ;
Som ← 0 ;
Prod ← 1 ;
Pour i ← 1 **à** 10 **faire**
Som ← Som+T[i] ;
Prod ← Prod*T[i] ;
Fin-Pour ;
Moy ← Som/10 ;
 	
Ecrire('La somme = ', Som:10:2) ;
Ecrire('Le produit = ', Prod:10:2) ;
Ecrire('La moyenne = ', Moy:10:2) ;
Fin.
	writeln('Donnez les éléments du tableau : ') ;
	For i := 1 **to** 10 **do**
	read(T[i]) ;
	
	Som := 0 ;
	Prod := 1 ;
	For i := 1 **to** 10 **do**
	Begin
	Som := Som+T[i] ;
	Prod := Prod*T[i] ;
	End ;
	Moy := Som/10 ;
	
	writeln('La somme = ', Som:10:2) ;
	writeln('Le produit = ', Prod:10:2) ;
	writeln('La moyenne = ', Moy:10:2) ;
	End.

```
1  Program Som_Moy_Prod_Tab ;
2  Var
3    T : array [1..10] of real ;
4    i : integer ;
5    Som, Prod, Moy : real ;
6  Begin
7    writeln('Donnez les éléments du tableau : ') ;
8    For i := 1 to 10 do
9      read(T[i]) ;
10
11   Som := 0 ;
12   Prod := 1 ;
13   For i := 1 to 10 do
14     Begin
15       Som := Som+T[i] ;
16       Prod := Prod*T[i] ;
17     End ;
18   Moy := Som/10 ;
19
20   writeln('La somme = ', Som:10:2) ;
21   writeln('Le produit = ', Prod:10:2) ;
22   writeln('La moyenne = ', Moy:10:2) ;
23 End.
```

```
MyPascal V1.20.5 (Exécution)
Donnez les éléments du tableau :
2 3 4 7.5 5 6 7 9 7 4
La somme =      54.50
Le produit = 9525600.00
La moyenne =       5.45
```

Après l'exécution

Exercice N°05 : (La recherche du plus petit et plus grand élément dans un tableau)

Ecrire un algorithme permettant de déterminer la valeur maximale et minimale dans un tableau de dix entiers, avec leurs positions dans le tableau. Traduire l'algorithme en Pascal.

Corrigé de l'exercice N°05 :

Algorithme	Pascal

Algorithme Max_Min_Tab ;
Variables
T : tableau [1..10] d'entier ;
Max, Min, Pos_Max, Pos_Min, i : entier ;
Début
 	
{-*-*-*- Entrées -*-*-*-}
Ecrire('Donnez les éléments du tableau : ') ;
Pour i ← 1 **à** 10 **faire**
Lire(T[i]) ;
Fin-Pour ;
 	
{-*-*-*- Traitement -*-*-*-}
Max ← T[1] ;
Pos_Max ← 1 ;
Pour i ← 2 **à** 10 **faire**
Si (Max<T[i]) **Alors**
Max ← T[i] ;
Pos_Max ← i ;
Fin-Si ;
Fin-Pour ;
 	
Min ← T[1] ;
Pos_Min ← 1 ;
Pour i ← 2 **à** 10 **faire**
Si (Min>T[i]) **Alors**
Min ← T[i] ;
Pos_Min ← i ;
Fin-Si ;
Fin-Pour ;
 	
{-*-*-*- Sorties -*-*-*-}
Ecrire('La valeur maximale = ', Max,
'occupant la position ', Pos_Max) ;
Ecrire('La valeur minimale = ', Min, 'occupant
la position ', Pos_Min) ;
 	
Fin.
	Program Max_Min_Tab ;
	Var
	T : array [1..10] of integer ;
	Max, Min, Pos_Max, Pos_Min, i : integer ;
	Begin
	
	{-*-*-*- Entrées -*-*-*-}
	writeln('Donnez les éléments du tableau : ') ;
	For i := 1 **to** 10 **do**
	read(T[i]) ;
	
	{-*-*-*- Traitement -*-*-*-}
	Max := T[1] ;
	Pos_Max := 1 ;
	For i := 2 **to** 10 **do**
	if (Max<T[i]) **then**
	Begin
	Max := T[i] ;
	Pos_Max := i ;
	End ;
	
	Min := T[1] ;
	Pos_Min := 1 ;
	For i := 2 **to** 10 **do**
	if (Min>T[i]) **then**
	Begin
	Min := T[i] ;
	Pos_Min := i ;
	End ;
	
	{-*-*-*- Sorties -*-*-*-}
	writeln('La valeur maximale = ', Max,
	'occupant la position ', Pos_Max) ;
	writeln('La valeur minimale = ', Min,
	'occupant la position ', Pos_Min) ;
	
	End.

A. OUARET

```
1  Program Max_Min_Tab ;
2  Var
3    T : array [1..10] of integer ;
4    Max, Min, Pos_Max, Pos_Min, i : integer ;
5  Begin
6    {-*-*-*- Entrées -*-*-*-}
7    writeln('Donnez les éléments du tableau : ') ;
8    For i := 1 to 10 do
9      read(T[i]) ;
10   {-*-*-*- Traitement -*-*-*-}
11   Max := T[1] ;
12   Pos_Max := 1 ;
13   For i := 2 to 10 do
14     if (Max<T[i]) then
15       Begin
16         Max := T[i] ;
17         Pos_Max := i ;
18       End ;
19   Min := T[1] ;
20   Pos_Min := 1 ;
21   For i := 2 to 10 do
22     if (Min>T[i]) then
23       Begin
24         Min := T[i] ;
25         Pos_Min := i ;
26       End ;
27 {-*-*-*- Sorties -*-*-*-}
28   writeln('La valeur maximale = ', Max, ' occupant la position ', Pos_Max) ;
29   writeln('La valeur minimale = ', Min, ' occupant la position ', Pos_Min) ;
30 End.
```

```
MyPascal V1.20.5 (Exécution)
Donnez les éléments du tableau :
5 6 7 99 5 3 0 5 7 2
La valeur maximale = 99 occupant la position 4
La valeur minimale = 0 occupant la position 7
```

Après l'exécution

Exercice N°06 :

Que fait l'algorithme suivant ?

Algorithme X ;
Variables
NB : Tableau [1..5] d'entier ;
i : entier ;
Début
Pour i ← 1 **à** 5 **faire**
NB[i] ← i*i ;
Fin-Pour;
Pour i ← 1 **à** 5 **faire**
Ecrire(NB[i]) ;
Fin-Pour;
Fin.

Corrigé de l'exercice N°06 :

Cet algorithme remplit un tableau avec cinq valeurs : 1, 4, 9, 16, 25. Il les affiche ensuite à l'écran.

Voici une simplification :

Algorithme X ;
Variables
NB : Tableau [1..5] d'entier ;
i : entier ;
Début

A. OUARET

```
   Pour i ← 1 à 5 faire
      NB[i] ← i*i ;
      Ecrire(NB[i]) ;
   Fin-Pour ;
Fin.
```

Exercice N°07 :

Que fait l'algorithme suivant ?

```
   Algorithme X ;
   Variables
        N : Tableau [1..6] d'entier ;
        i,k : entier ;
   Début
       N[1] ← 1 ;
       Pour k ← 2 à 6 faire
          N[k] ← N[k-1]+2 ;
       Fin-Pour;
       Pour i ← 1 à 6 faire
           Ecrire(N[i]) ;
       Fin-Pour;
   Fin.
```

Peut-on simplifier cet algorithme pour avoir le même résultat ?

Corrigé de l'exercice N°07 :

Cet algorithme remplit un tableau avec les six valeurs : 1, 3, 5, 7, 9, 11. Il les affiche ensuite à l'écran.

Voici une simplification :

```
  Algorithme X ;
  Variables
       N : Tableau [1..6] d'entier ;
       i,k : entier ;
  Début
      N[1] ← 1 ;
  Ecrire(N[1]) ;
  Pour k ← 2 à 6 faire
     N[k] ← N[k-1]+2 ;
     Ecrire(N[k]) ;
  Fin-Pour;
  Fin.
```

A. OUARET

Exercice N°08 : (Le tri d'un tableau par sélection)

Ecrire un algorithme permettant de trier un tableau de dix entiers en ordre croissant. Traduire l'algorithme en Pascal.

Corrigé de l'exercice N°08 :

Algorithme	Pascal

Algorithme Tri_Tabl_Par_Selection ;
Variables
T : tableau [1..10] d'entier ;
x, i, j : entier ;
Début
{-*-*-*- Entrées -*-*-*-}
Ecrire('Donnez les éléments du tableau : ') ;
Pour i ← 1 **à** 10 **faire**
Lire(T[i]) ;
Fin-Pour ;
{-*-*-*- Traitement -*-*-*-}
Pour i ← 1 **à** 9 **faire**
Pour j ← i+1 **à** 10 **faire**
Si (T[i]>T[j]) **Alors**
x ← T[i] ;
T[i] ← T[j] ;
T[j] ← x ;
Fin-Si ;
Fin-Pour ;
Fin-Pour ;
{-*-*-*- Sorties -*-*-*-}
Ecrire('Voici les éléments du tableau après le tri : ') ;
Pour i ← 1 **à** 10 **faire**
Ecrire(T[i]:3) ;
Fin-Pour ;
Fin.
	Program Tri_Tabl_Par_Selection ;
	Var
	T : array [1..10] of integer ;
	x, i, j : integer ;
	Begin
	writeln('Donnez les éléments du tableau : ') ;
	For i := 1 **to** 10 **do**
	read(T[i]) ;
	{-*-*-*- Traitement -*-*-*-}
	For i := 1 **to** 9 **do**
	For j := i+1 **to** 10 **do**
	if (T[i]>T[j]) **then**
	Begin
	x := T[i] ;
	T[i] := T[j] ;
	T[j] := x ;
	End ;
	{-*-*-*- Sorties -*-*-*-}
	writeln('Voici les éléments du tableau après le tri : ') ;
	For i := 1 **to** 10 **do**
	write(T[i]:3) ;
	End.

A.OUARET

```
1  Program Tri_Tabl_Par_Selection ;
2  Var
3     T : array [1..10] of integer ;
4     x, i, j : integer ;
5  Begin
6     {-*-*-*- Entrées -*-*-*-}
7     writeln('Donnez les éléments du tableau : ') ;
8     For i := 1 to 10 do
9       read(T[i]) ;
10    {-*-*-*- Traitement -*-*-*-}
11    For i := 1 to 9 do
12       For j := i+1 to 10 do
13          if (T[i]>T[j]) then
14             Begin
15                x := T[i] ;
16                T[i] := T[j] ;
17                T[j] := x ;
18             End ;
19    {-*-*-*- Sorties -*-*-*-}
20    writeln('Voici les éléments du tableau après le tri : ') ;
21    For i := 1 to 10 do
22       write(T[i]:3) ;
23 End.
```

```
MyPascal V1.20.5 (Exécution)
Donnez les éléments du tableau :
5 6 7 4 5 3 0 9 8 6
Voici les éléments du tableau après le tri :
  0  3  4  5  5  6  6  7  8  9
```

Après l'exécution

Exercice N°09 :

Ecrire un algorithme permettant de saisir un tableau de N éléments des valeurs réelles non nulles. Une fois la saisie terminée, le programme affichera le nombre de valeurs négatives et le nombre de valeurs positives.

Corrigé de l'exercice N°09 :

Algorithme	Pascal

Algorithme Val_Neg_Val_Pos;
Variables
T : Tableau [1..100] de réels;
N, i, NVP, NVN : entier;
 	
Début
{-*-*-*- Entrées -*-*-*-}
Ecrire('Donner la taille du vecteur T : ');
Lire(N);
Ecrire('Donner les composantes non nulles du vecteur T');
Pour i ← 1 **à** N **faire**
Lire(T[i]);
Fin-Pour ;
{-*-*-*- Traitement -*-*-*-}
NVP ← 0;
NVN ← 0;
Pour i ← 1 **à** N **faire**
Si T[i]>0 **Alors**
NVP ← NVP+1
Sinon
NVN ← NVN+1;
Fin-Si ;
	Program Val_Neg_Val_Pos;
	Var
	T : array [1..100] of real;
	N, i, NVP, NVN : integer;
	
	Begin
	{-*-*-*- Entrées -*-*-*-}
	write('Donner la taille du vecteur T : ');
	read(N);
	writeln('Donner les composantes non nulles du vecteur T');
	For i:=1 **to** N **do**
	read(T[i]);
	
	{-*-*-*- Traitement -*-*-*-}
	NVP:=0;
	NVN:=0;
	for i:=1 **to** N **do**
	if T[i]>0 **then**
	NVP:=NVP+1
	else
	NVN:=NVN+1;

Fin-Pour ; {-*-*-*- Sorties -*-*-*-} Ecrire('Le nombre de valeurs positives est : ', NVP); Ecrire('Le nombre de valeurs négatives est :', NVN); **Fin.**	{-*-*-*- Sorties -*-*-*-} writeln('Le nombre de valeurs positives est : ', NVP); write('Le nombre de valeurs négatives est :', NVN); **End.**

```
Program Val_Neg_Val_Pos;
Var
   T : array [1..100] of real;
   N, i, NVP, NVN : integer;

Begin
   {-*-*-*- Entrées -*-*-*-}
   write('Donner la taille du vecteur T : ');
   read(N);
   writeln('Donner les composantes non nulles du vecteur T');
   For i:=1 to N do
      read(T[i]);

   {-*-*-*- Traitement -*-*-*-}
   NVP:=0;
   NVN:=0;
   for i:=1 to N do
      if T[i]>0 then
         NVP:=NVP+1
      else
   NVN:=NVN+1;

   {-*-*-*- Sorties -*-*-*-}
   writeln('Le nombre de valeurs positives est : ', NVP);
   write('Le nombre de valeurs négatives est :', NVN);
End.
```

```
MyPascal V1.20.5 (Exécution)
Donner la taille du vecteur T : 6
Donner les composantes non nulles du vecteur T
4 -5 8 9 -3 1
Le nombre de valeurs positives est : 4
Le nombre de valeurs négatives est :2
```

Après l'exécution

Exercice N°10 :

Soit T un tableau contenant N entiers (10≤N≤50). On propose d'écrire un programme Pascal qui permet d'éclater T en deux tableaux :

TN (contenant les éléments négatifs de T) et **TP** (contenant les éléments positifs de T).

Exemple :

T	2	7	-4	0	-7	4	3	0	-8	-1	3

TN	-4	-7	-8	-1

TP	2	7	0	4	3	0	3

Corrigé de l'exercice N°10 :

Algorithme	Pascal
Algorithme Eclater_tab ; **Var** T, TN, TP : Tableau [1..50] d'entier ; n, i, j, k : entier ;	**Program** Eclater_tab ; **Var** T, TN, TP : array [1..50] of integer ; n, i, j, k : integer ;

```
Début
  {-*-*-*- Entrées -*-*-*-}
  Répéter
    Ecrire('Saisir un entier') ;
    Lire(n) ;
  Jusqu'à (n>=10) et (n<=50) ;

  Ecrire('Saisir les ', n, ' éléments de T') ;
  Pour i ← 1 à n faire
    Read (T[i]) ;
  Fin-Pour;

  {-*-*-*- Traitement -*-*-*-}
  j ← 0 ;
  k ← 0 ;
  Pour i ← 1 à n faire
    Si T[i] < 0 Alors
        j ← j+1 ;
        TN[j] ← T[i] ;
    Sinon
        k ← k+1 ;
        TP[k] ← T[i] ;
  Fin-Pour;

  {-*-*-*- Sorties -*-*-*-}
  Ecrire('TN : ');
  Pour i ← 1 à j faire
    Ecrire(TN[i]:4) ;
  Fin-Pour;
  Ecrire('TP : ');
  Pour i ← 1 à k faire
    Ecrire(TP[i]:4) ;
  Fin-Pour;
Fin.
```

```
Begin
  {-*-*-*- Entrées -*-*-*-}
  Repeat
    Writeln('Saisir un entier') ;
    Readln(n) ;
  Until (n>=10) and (n<=50) ;

  Writeln ('Saisir les ', n, ' éléments de T') ;
  For i:=1 to n do
    Read (T[i]) ;
 {-*-*-*- Traitement -*-*-*-}
  j := 0 ; k := 0 ;
  For i := 1 to n do
    If T[i] < 0 Then
      Begin
        j := j+1 ;
        TN[j] := T[i] ;
      End
    Else
      Begin
        k := k+1 ;
        TP[k] := T[i] ;
      End ;
  {-*-*-*- Sorties -*-*-*-}
  Write('TN : ');
  For i := 1 to j do
    Write (TN[i]:4) ;
     Writeln;

  Write('TP : ');
  For i := 1 to k do
    Write (TP[i]:4) ;
End.
```

A. OUARET

```
1  Program Eclater_tab ;
2  Var
3  T, TN, TP : array [1..50] of integer ;
4  n, i, j, k : integer ;
5
6  Begin
7    {-*-*-*- Entrées -*-*-*-}
8    Repeat
9      Writeln('Saisir un entier') ;
10     Readln(n) ;
11   Until (n>=10) and (n<=50) ;
12
13   Writeln ('Saisir les ', n, ' éléments de T') ;
14   For i:=1 to n do
15     Read (T[i]) ;
16   {-*-*-*- Traitement -*-*-*-}
17   j := 0 ; k := 0 ;
18   For i := 1 to n do
19     If T[i] < 0 Then
20       Begin
21         j := j+1 ;
22         TN[j] := T[i] ;
23       End
24     Else
25       Begin
26         k := k+1 ;
27         TP[k] := T[i] ;
28       End ;
29   {-*-*-*- Sorties -*-*-*-}
30   Write('TN : ');
31   For i := 1 to j do
32     Write (TN[i]:4) ;
33     Writeln;
34   Write('TP : ');
35   For i := 1 to k do
36     Write (TP[i]:4) ;
37 End.
```

```
MyPascal V1.20.5 (Exécution)
Saisir un entier
11
Saisir les 11 éléments de T
1 3 7 3 5 -4 8 -2 8 -9 88
TN :   -4  -2  -9
TP :    1   3   7   3   5   8   8  88
```

Après l'exécution

Exercice N°11 : (Somme de deux vecteurs T1 et T2)

Ecrire un algorithme permettant de faire la somme de deux vecteurs T1 et T2 de N éléments des valeurs réelles. Traduire l'algorithme en Pascal.

Corrigé de l'exercice N°11 :

Algorithme	Pascal

Algorithme Somme ;
Variables
T1, T2, T : Tableau [1..100] de réel;
N, i : entier;
Début
Ecrire('Donner la taille du vecteur T1 et T2 : ');
Lire (N) ;
Ecrire('Donner les composantes du vecteur T1 : ');
pour i ← 1 **à** N **faire**
	Program Somme;
	Var
	T1, T2, T : array[1..100] of real;
	N, i : integer;
	Begin
	Write('Donner la taille du vecteur T1 et T2 : ');
	Read (N);
	Writeln('Donner les composantes du vecteur T1 : ');
	for i:=1 **to** N **do**
	read (T1[i]);

Lire (T1[i]) ; **FinPour** Ecrire('Donner les composantes du vecteur T2 : '); **pour** i ← 1 **à** N **faire** Lire (T2[i]) ; **FinPour** **pour** i=1 **à** N faire T[i] ← T1[i] + T2[i] ; **FinPour** Ecrire('T1 + T2 = '); **pour** i=1 **à** N **faire** Ecrire (T[i]:10:2) ; **FinPour** **Fin.**	Writeln('Donner les composantes du vecteur T2 : '); **for** i:=1 **to** N **do** read (T2[i]); **for** i:=1 **to** N **do** T[i]:=T1[i]+T2[i] ; Writeln('T1 + T2 = '); **for** i:=1 **to** N **do** Write(T[i]:10:2) ; **End.**

```
1  Program Somme;
2  Var
3    T1, T2, T : array[1..100] of real;
4    N, i : integer;
5  Begin
6    Write('Donner la taille du vecteur T1 et T2 : ');
7    Read (N);
8    Writeln('Donner les composantes du vecteur T1 : ');
9    for i:=1 to N do
10     read (T1[i]);
11   Writeln('Donner les composantes du vecteur T2 : ');
12   for i:=1 to N do
13     read (T2[i]);
14   for i:=1 to N do
15     T[i]:=T1[i]+T2[i] ;
16   Writeln('T1 + T2 = ');
17   for i:=1 to N do
18     Write(T[i]:10:2) ;
19 End.
```

```
MyPascal V1.20.5 (Exécution)
Donner la taille du vecteur T1 et T2 : 5
Donner les composantes du vecteur T1 :
5 6 7 9 0.5
Donner les composantes du vecteur T2 :
5 6 8 3 1
T1 + T2 =
     10.00     12.00     15.00     12.00      1.50
```

Après l'exécution

Exercice N°12 : (Permutation entre les cases d'indice K et L)

Soit T un vecteur de type réel et de taille N, et soient K et L deux positions dans le vecteur T. Écrire un algorithme/Programme PASCAL qui permet de permuter entre les deux éléments du vecteur T, d'indice K et L.

Corrigé de l'exercice N°12 :

Algorithme	Pascal
Algorithme permutation ; **Variables** T : Tableau [1..100] de réel ; i, N, K, L : entier ; Z : **réel** ; **Début** {-*-*-*- Entrées -*-*-*-}	**Program** permutation ; **Var** T : array [1..100] of real ; i, N,K, L : integer ; Z : **real** ; **Begin** {-*-*-*- Entrées -*-*-*-}

```
   Ecrire('Donner la taille du vecteur T : ') ;
   Lire(N) ;
   Ecrire('Donner les composantes du vecteur T : ') ;
   Pour i←1 à N faire
      Lire(T[i]) ;
   FinPour ;

   Ecrire('Donner deux indices K et L entre 1 et',N,' :');
   Lire(K,L) ;

   {-*-*-*- Traitement -*-*-*-}
   Z ← T[K] ;
   T[K] ← T[L] ;
   T[L] ← Z ;

   {-*-*-*- Sorties -*-*-*-}
   Pour i←1 à N faire
      Ecrire(T[i]:8:2) ;
   FinPour ;
Fin.
```

```
   Write('Donner la taille du vecteur T : ') ;
   Read(N) ;
   Writeln('Donner les composantes du vecteur T : ') ;
   For i :=1 to N do
      Read(T[i]) ;

   Write('Donner deux indices K et L entre 1 et',N,' :');
   Read(K,L) ;

   {-*-*-*- Traitement -*-*-*-}
   Z := T[K] ;
   T[K] := T[L] ;
   T[L] := Z ;

   {-*-*-*- Sorties -*-*-*-}
   For i :=1 to N do
      Write(T[i]:8:2) ;
End.
```

```
1  Program permutation ;
2  Var
3     T : array [1..100] of real ;
4     i, N,K, L : integer ;
5     Z : real ;
6  Begin
7     {-*-*-*- Entrées -*-*-*-}
8     Write('Donner la taille du vecteur T : ') ;
9     Read(N) ;
10    Writeln('Donner les composantes du vecteur T : ') ;
11    For i :=1 to N do
12       Read(T[i]) ;
13
14    Write('Donner deux indices K et L entre 1 et ',N,' : ');
15    Read(K,L) ;
16
17    {-*-*-*- Traitement -*-*-*-}
18    Z := T[K] ;
19    T[K] := T[L] ;
20    T[L] := Z ;
21
22    {-*-*-*- Sorties -*-*-*-}
23    For i :=1 to N do
24       Write(T[i]:8:2) ;
25 End.
```

```
MyPascal V1.20.5 (Exécution)
Donner la taille du vecteur T : 6
Donner les composantes du vecteur T :
5 -4 0.5 6 0 3
Donner deux indices K et L entre 1 et  6 : 3 5
    5.00   -4.00    0.00    6.00    0.50    3.00
```

Après l'exécution

Exercice N°13 : (Produit scalaire de deux vecteurs T1 et T2)

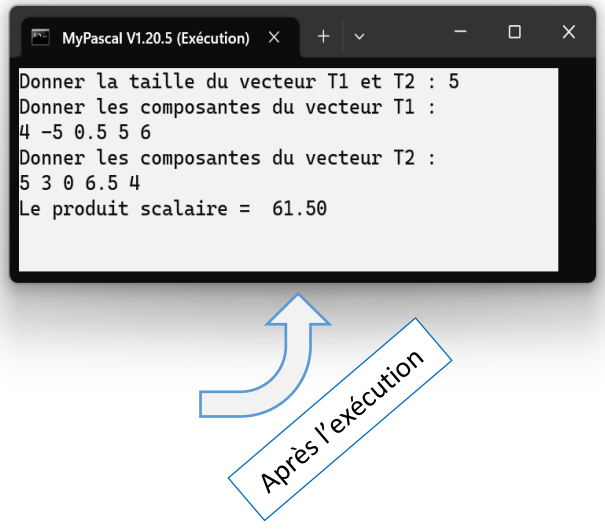
Ecrire un algorithme permettant de calculer et afficher le produit scalaire de deux vecteurs T1 et T2 de N éléments des valeurs réelles. Traduire l'algorithme en Pascal.

Corrigé de l'exercice N°13 :

Algorithme	Pascal

Algorithme Produit_Scalaire ;
Variables
T1, T2 : tableau [1..100] de reel
N, i : entier
PS : reel
Début
Ecrire('Donner la taille du vecteur T1 et T2 : ');
Lire (N) ;
Writeln('Donner les composantes du vecteur T1 : ');
pour i ← 1 **à** N **faire**
Lire (T1[i]) ;
Fin-Pour
Writeln('Donner les composantes du vecteur T2 : ');
pour i ← 1 **à** N **faire**
Lire (T2[i]) ;
Fin-Pour
PS ← 0 ;
pour i ← 1 **à** N faire
PS ← PS + T1[i] * T2[i] ;
Fin-Pour
Ecrire('Le produit scalaire = ', PS:6:2);
Fin
	Program Produit_Scalaire;
	Var
	T1, T2 : array [1..100] of real;
	N, i : integer;
	PS : real;
	Begin
	Write('Donner la taille du vecteur T1 et T2 : ');
	Read (N);
	Writeln('Donner les composantes du vecteur T1 : ');
	for i:=1 **to** N **do**
	read(T1[i]);
	Writeln('Donner les composantes du vecteur T2 : ');
	for i:=1 **to** N **do**
	read(T2[i]);
	PS:=0;
	for i:=1 **to** N **do**
	PS:= PS + T1[i]*T2[i] ;
	Write('Le produit scalaire = ', PS:6:2);
	End.

```
1  Program Produit_Scalaire;
2  Var
3    T1, T2 : array [1..100] of real;
4    N, i : integer;
5    PS : real;
6  Begin
7    Write('Donner la taille du vecteur T1 et T2 : ');
8    Read (N);
9  Writeln('Donner les composantes du vecteur T1 : ');
10   for i:=1 to N do
11   read(T1[i]);
12 Writeln('Donner les composantes du vecteur T2 : ');
13 for i:=1 to N do
14   read(T2[i]);
15 PS:=0;
16 for i:=1 to N do
17   PS:= PS + T1[i]*T2[i] ;
18 Write('Le produit scalaire = ', PS:6:2);
19 End.
```



Exercice N°14 : (Somme des éléments divisible par 3 d'un vecteur T)

Ecrire un algorithme permettant de calculer et afficher la somme des éléments divisible par 3 d'un vecteur T de type réel et de taille N. Traduire l'algorithme en Pascal.

Corrigé de l'exercice N°14 :

Algorithme	Pascal

Algorithme Vecteur;
Variables
T : Tableau [1..100] d'entier ;
N, i, S : entier;
Début
{-*-*-*- Entrées -*-*-*-}
Ecrire('Donner la taille du vecteur T : ');
Lire(N);
Ecrire('Donner les composantes du vecteur T : ');
Pour i←1 **à** N **faire**
Lire(T[i]);
Fin-Pour;
{-*-*-*- Traitements -*-*-*-}
S ← 0;
Pour i←1 **à** N **faire**
Si (T[i] **mod** 3 = 0) **alors**
S ← S+T[i];
Fin-Si;
Fin-Pour;
{-*-*-*- Sorties -*-*-*-}
Ecrire('La somme S=', S);
Fin.
	Program Vecteur ;
	Var
	T : array [1..100] of integer;
	N, i, S : integer;
	Begin
	{-*-*-*- Entrées -*-*-*-}
	Write('Donner la taille du vecteur T : ');
	Read(N);
	Writeln('Donner les composantes du vecteur T : ');
	For i:=1 **to** N **do**
	Read(T[i]);
	{-*-*-*- Traitements -*-*-*-}
	S:=0;
	For i:=1 **to** N **do**
	if (T[i] **mod** 3 = 0) **then**
	S:=S+T[i];
	{-*-*-*- Sorties -*-*-*-}
	Write('La somme S=',S);
	End.

A.OUARET

```
1  Program Vecteur ;
2  Var
3     T : array [1..100] of integer;
4     N, i, S : integer;
5  Begin
6
7     {-*-*-*- Entrées -*-*-*-}
8     Write('Donner la taille du vecteur T : ');
9     Read(N);
10    Writeln('Donner les composantes du vecteur T : ');
11    For i:=1 to N do
12       Read(T[i]);
13
14    {-*-*-*- Traitements -*-*-*-}
15    S:=0;
16    For i:=1 to N do
17       if (T[i] mod 3 = 0) then
18          S:=S+T[i];
19
20    {-*-*-*- Sorties -*-*-*-}
21    Write('La somme S=',S);
22 End.
```

Exercice N°15 : (Inverser les éléments d'un vecteur)

1. Ecrire un algorithme/programme PASCAL qui permet d'inverser les éléments d'un vecteur de type réel T dans un autre vecteur V.
2. Réaliser la même opération dans le même vecteur T (sans utiliser le vecteur V).

Corrigé de l'exercice N°15 :

Question 1 :

Algorithme	Pascal

Algorithme inverser_T_dans_V;
Variables
i, N : **entier**;
V, T : Tableau [1..100] **de réel**;
Début
{-*-*-*- Entrées -*-*-*-}
Ecrire('Donner la taille du vecteur T :');
Lire(N);
Ecrire('Donner les composantes de T :');
Pour i←1 **à** N **faire**
Lire(T[i]);
FinPour
 	
{-*-*-*- Traitement -*-*-*-}
Pour i←1 **à** N **faire**
V[i] ← T[N-i+1];
FinPour
	Program inverser_T_dans_V;
	Var
	i, N : **integer**;
	V, T : **array** [1..100] **of real**;
	Begin
	{-*-*-*- Entrées -*-*-*-}
	Writeln('Donner la taille du vecteur T :');
	Read(N);
	Writeln('Donner les composantes de T :');
	For i := 1 **to** N **do**
	Read(T[i]);
	
	{-*-*-*- Traitement -*-*-*-}
	For i := 1 **to** N **do**
	V[i] := T[N-i+1];
	
	{-*-*-*- Sorties -*-*-*-}
	Writeln('L"inverse de T est V :');

```
  {-*-*-*- Sorties -*-*-*-}
  Ecrire('L"inverse de T est V :');
  Pour i←1 à N faire
    Ecrire(V[i]:8:2);
  FinPour
Fin.
```

```
  For i := 1 to N do
    Write(V[i]:8:2);{Afficher sur 8 espaces avec 2
                     chiffres après la virgule}
End.
```

```
1  Program inverser_T_dans_V;
2   Var
3     i, N : integer;
4     V, T : array [1..100] of real;
5  Begin
6    {-*-*-*- Entrées -*-*-*-}
7    Writeln('Donner la taille du vecteur T :');
8    Read(N);
9    Writeln('Donner les composantes de T :');
10   For i := 1 to N do
11     Read(T[i]);
12
13   {-*-*-*- Traitement -*-*-*-}
14   For i := 1 to N do
15     V[i] := T[N-i+1];
16
17   {-*-*-*- Sorties -*-*-*-}
18   Writeln('L"inverse de T est V :');
19   For i := 1 to N do
20     Write(V[i]:8:2);{Afficher sur 8 espaces avec 2 chiffres après la virgule}
21 End.
```

```
MyPascal V1.20.5 (Exécution)
Donner la taille du vecteur T :
6
Donner les composantes de T :
4 6 5 0 7 5
L'inverse de T est V :
    5.00    7.00    0.00    5.00    6.00    4.00
```

Après l'exécution

Question 2 :

Algorithme	Pascal

Algorithme inverser_T_dans_T;
Variables
i, N : **entier**;
T : Tableau [1..100] **de réel**;
Z : **réel**;
Début
{-*-*-*- Entrées -*-*-*-}
Ecrire('Donner la taille du vecteur T :');
Lire(N);
Ecrire('Donner les composantes de T :');
Pour i←1 **à** N **faire**
Lire(T[i]);
FinPour ;
 	
{-*-*-*- Traitement -*-*-*-}
Pour i←1 **à** (N div 2) **faire**
Z ← T[i];
T[i] ← T[N-i+1];
T[N-i+1] ← Z;
FinPour ;
 	
{-*-*-*- Sorties -*-*-*-}
	Program inverser_T_dans_T;
	Var
	i, N : **integer**;
	T : **array** [1..100] **of real**;
	Z : **real** ;
	Begin
	{-*-*-*- Entrées -*-*-*-}
	Writeln('Donner la taille du vecteur T :');
	Read(N);
	Writeln('Donner les composantes de T :');
	For i := 1 **to** N **do**
	Read(T[i]);
	
	{-*-*-*- Traitement -*-*-*-}
	For i := 1 **to** (N div 2) **do**
	Begin
	Z := T[i];
	T[i] := T[N-i+1];
	T[N-i+1] := Z;
	End;
	
	{-*-*-*- Sorties -*-*-*-}

Ecrire('L"inverse de T est :'); **Pour** i←1 **à** N **faire** Ecrire(T[i]:8:2); **FinPour ;** **Fin.**	Writeln('L"inverse de T est :'); **For** i := 1 **to** N **do** Write(T[i]:8:2) ; **End.**

```
1  Program inverser_T_dans_T;
2  Var
3    i, N : integer;
4    T : array [1..100] of real;
5    Z : real ;
6  Begin
7    {-*-*-*- Entrées -*-*-*-}
8    Writeln('Donner la taille du vecteur T :');
9    Read(N);
10   Writeln('Donner les composantes de T :');
11   For i := 1 to N do
12     Read(T[i]);
13
14   {-*-*-*- Traitement -*-*-*-}
15   For i := 1 to (N div 2) do
16     Begin
17       Z := T[i];
18       T[i] := T[N-i+1];
19       T[N-i+1] := Z;
20     End;
21
22   {-*-*-*- Sorties -*-*-*-}
23   Writeln('L"inverse de T est :');
24   For i := 1 to N do
25     Write( T[i]:8:2) ;
26 End.
```

```
MyPascal V1.20.5 (Exécution)
Donner la taille du vecteur T :
6
Donner les composantes de T :
4 6 5 0 7 5
L'inverse de T est :
    5.00    7.00    0.00    5.00    6.00    4.00
```

Après l'exécution

Exercice N°16 :

Ecrire un algorithme/programme pascal qui permet de lire un tableau de N éléments réels, qui calcule et affiche le nombre d'éléments égaux au dernier élément lu.

Exemple : pour le tableau suivant : le nombre = 5

4	-3	4	4	6	4	12	4	-9	4

Corrigé de l'exercice N°16 :

Algorithme	Pascal

Algorithme Test;
Variables
i, N, NR : entier; {NR : nombre de répétition}
V : tableau [1..50] de réel;
Début
{-*-*-*- Entrées -*-*-*-}
Ecrire('Donner la taille du vecteur V : ');
Lire(N) ;
Ecrire('Donner les composantes de V :');
Pour i ← 1 **à** N **faire**
	Program Test;
	Var
	i, N, NR : integer; {NR : nombre de répétition}
	V : array [1..50] of integer;
	Begin
	{-*-*-*- Entrées -*-*-*-}
	Writeln('Donner la taille du vecteur V : ');
	read(N) ;
	Writeln('Donner les composantes de V :');
	For i:=1 **to** N **do**
	read (V[i]);

```
    Lire(V[i]);
  FinPour

  NR ← 0;
  Pour i ← 1 à (N-1) faire
    Si V[i] = V[N]  alors
      NR:= NR + 1;
    FinSi
  FinPour
  {-*-*-*- Sorties -*-*-*-}
  Ecrire ('Le nombre d''éléments égaux au
          dernier élément lu est : ', NR) ;
Fin.
```

```
  NR:=0;
  For i :=1 to (N-1) do
    If V[i] = V[N]  then
      NR:= NR + 1;
  {-*-*-*- Sorties -*-*-*-}
  Write ('Le nombre d''éléments égaux au
          dernier élément lu est : ', NR) ;
End.
```

```
1  Program Test;
2  Var
3  i, N, NR : integer;  {NR : nombre de répétition}
4  V : array [1..50] of integer;
5  Begin
6   {-*-*-*- Entrées -*-*-*-}
7   Writeln('Donner la taille du vecteur V : ');
8   read(N) ;
9   Writeln('Donner les composantes de V :');
10  For i:=1 to N do
11    read (V[i]);
12
13  NR:=0;
14  For i :=1 to (N-1) do
15    If V[i] = V[N]  then
16      NR:= NR + 1;
17  {-*-*-*- Sorties -*-*-*-}
18  Write ('Le nombre d''éléments égaux au dernier élément lu est : ', NR) ;
19 End.
```

```
MyPascal V1.20.5 (Exécution)
Donner la taille du vecteur V :
10
Donner les composantes de V :
4 -3 4 4 6 4 12 4 -9 4
Le nombre d'éléments égaux au dernier élément lu est : 5
```

Après l'exécution

Exercice N°17 :

Ecrire un algorithme/programme pascal qui permet de lire un tableau T de N éléments réels, qui calcule et affiche le nombre d'éléments égaux au premier élément lu.

Exemple : pour le tableau suivant : le nombre = 5

4	-3	4	4	6	4	12	4	-9	4

Corrigé de l'exercice N°17 :

Algorithme	Pascal
Algorithme Test; **Variables** i, N, NR : entier; {NR : nombre de répétition} V : tableau [1..50] de réel; **Début** {-*-*-*- Entrées -*-*-*-} Ecrire('Donner la taille du vecteur V : ');	**Program** Test; **Var** i, N, NR : integer; {NR : nombre de répétition} V : array [1..50] of integer; **Begin** {-*-*-*- Entrées -*-*-*-} Writeln('Donner la taille du vecteur V : ');

Lire(N) ;
Ecrire('Donner les composantes de V :');
Pour I ← 1 **à** N **faire**
Lire(V[i]);
FinPour
{-*-*-*- Entrées -*-*-*-}
NR ← 0;
Pour i ← 2 **à** N **faire**
Si V[i] = V[1] **alors**
NR:= NR + 1;
FinSi
FinPour
{-*-*-*- Sorties -*-*-*-}
Ecrire ('Le nombre d''éléments égaux au premier élément lu est : ', NR) ;
Fin.
	read(N) ;
	Writeln('Donner les composantes de V :');
	For i:=1 **to** N **do**
	read (V[i]);
	{-*-*-*- Entrées -*-*-*-}
	NR:=0;
	For i :=2 **to** N **do**
	If V[i] = V[1] **then**
	NR:= NR + 1;
	{-*-*-*- Sorties -*-*-*-}
	Write ('Le nombre d''éléments égaux au premier élément lu est : ', NR) ;
	End.

```
1  Program Test;
2  Var
3  i, N, NR : integer;  {NR : nombre de répétition}
4  V : array [1..50] of integer;
5  Begin
6   {-*-*-*- Entrées -*-*-*-}
7    Writeln('Donner la taille du vecteur V : ');
8    read(N) ;
9    Writeln('Donner les composantes de V :');
10   For i:=1 to N do
11       read (V[i]);
12   {-*-*-*- Entrées -*-*-*-}
13   NR:=0;
14   For i :=2 to N do
15       If V[i] = V[1] then
16           NR:= NR + 1;
17   {-*-*-*- Sorties -*-*-*-}
18    Write ('Le nombre d''éléments égaux au premier élément lu est : ', NR) ;
19 End.
```

```
MyPascal V1.20.5 (Exécution)
Donner la taille du vecteur V :
10
Donner les composantes de V :
4 -3 4 4 6 4 12 4 -9 4
Le nombre d'éléments égaux au premier élément lu est : 5
```

Après l'exécution

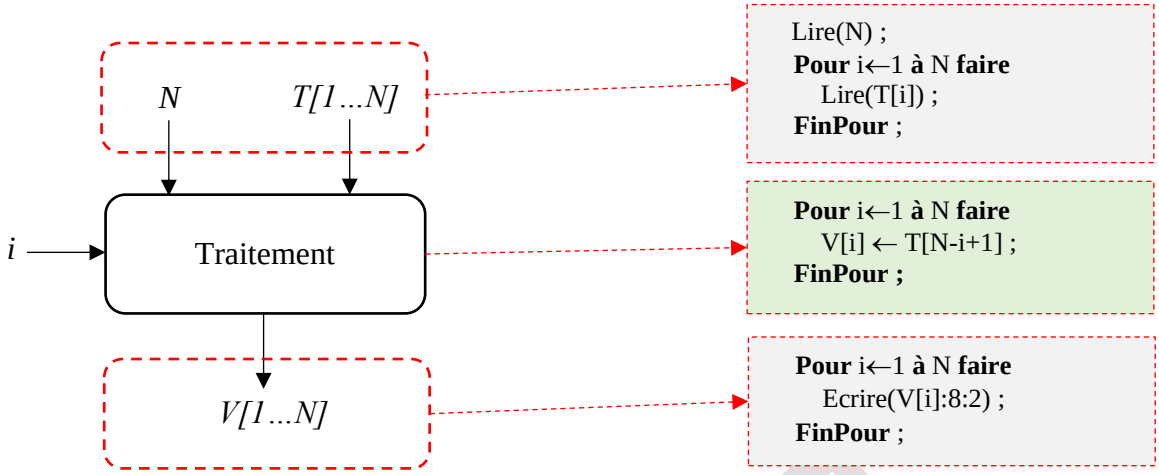
Exercice N°18 : Inverser les éléments d'un vecteur

1. Ecrire un algorithme/programme PASCAL qui permet d'inverser les éléments d'un vecteur de type réel T dans un autre vecteur V.
2. Réaliser la même opération dans le même vecteur T (sans utiliser le vecteur V).

Corrigé de l'exercice N°18 :

Question 01 :

Les variables d'entrée, variable de sorite et la partie traitement sont présentées dans le schéma ci-dessous :



Pour illustrer la partie traitement, nous prenons un exemple :

N = 6, T = [11 13 -8 5 7 22]

Nous devons avoir le vecteur V : V = [22 7 5 -8 13 11]

Ce que nous remarquons : (i allant de 1 à N, avec N=6)

Pour i=1 ➔ V[1]=T[6]
Pour i=2 ➔ V[2]=T[5]
Pour i=3 ➔ V[3]=T[4]
Pour i=4 ➔ V[4]=T[3]
Pour i=5 ➔ V[5]=T[2]
Pour i=6 ➔ V[6]=T[1]

V[1]=T[6-1+1]
V[2]=T[6-2+1]
V[3]=T[6-3+1]
V[4]=T[6-4+1]
V[5]=T[6-5+1]
V[6]=T[6-6+1]

Pour toutes égalités, on peut écrire :

Pour i ←1 **à** N **faire**
V[i] ← T[N-i+1] ;
FinPour ;

Algorithme	Programme PASCAL

Algorithme inverser_T_dans_V;
Variables
i, N : **entier**;
V, T : Tableau [1..100] **de réel**;
Début
{-*-*-*- Entrées -*-*-*-}
Ecrire('Donner la taille du vecteur T :');
Lire(N);
Ecrire('Donner les composantes de T :');
Pour i←1 **à** N **faire**
Lire(T[i]);
FinPour
	Program inverser_T_dans_V;
	Var
	i, N : **integer**;
	V, T : **array** [1..100] **of real**;
	Begin
	{-*-*-*- Entrées -*-*-*-}
	Writeln('Donner la taille du vecteur T :');
	Read(N);
	Writeln('Donner les composantes de T :');
	For i := 1 **to** N **do**
	Read(T[i]);
	
	{-*-*-*- Traitement -*-*-*-}

```
{-*-*-*- Traitement -*-*-*-}
Pour i←1 à N faire
    V[i] ← T[N-i+1];
FinPour

{-*-*-*- Sorties -*-*-*-}
Ecrire('L"inverse de T est V :');
Pour i←1 à N faire
    Ecrire(V[i]:8:2);
FinPour
Fin.
```

```
For i := 1 to N do
    V[i] := T[N-i+1];

{-*-*-*- Sorties -*-*-*-}
Writeln('L"inverse de T est V :');
For i := 1 to N do
    Write(V[i]:8:2);{Afficher sur 8 espaces avec 2
                     chiffres après la virgule}
End.
```

```
1  Program inverser_T_dans_V;
2   Var
3     i,N : integer;
4     V,T : Array [1..100] of real;
5  Begin
6   {-*-*-*- Entrées -*-*-*-}
7   Writeln('Donner la taille du vecteur T :');
8   Read(N);
9   Writeln('Donner les composantes de T :');
10  For i := 1 to N do
11    Read( T[i] );
12
13  {-*-*-*- Traitement -*-*-*-}
14  For i := 1 to N do
15    V[i] := T[N-i+1];
16
17  {-*-*-*- Sorties -*-*-*-}
18  Writeln('L"inverse de T est V :');
19  For i := 1 to N do
20    Write(V[i]:8:2);{Afficher sur 8 espaces avec 2 chiffres après la virgule}
21 End.
```

```
MyPascal V1.20.5 (Exécution) C:\Users\Ahmed\Des...
Donner la taille du vecteur T :
6
Donner les composantes de T :
11  13  -8  5  7 22
L'inverse de T est V :
   22.00    7.00    5.00   -8.00   13.00   11.00
```

Après l'exécution

Question 02 : Inverser le vecteur T dans lui même

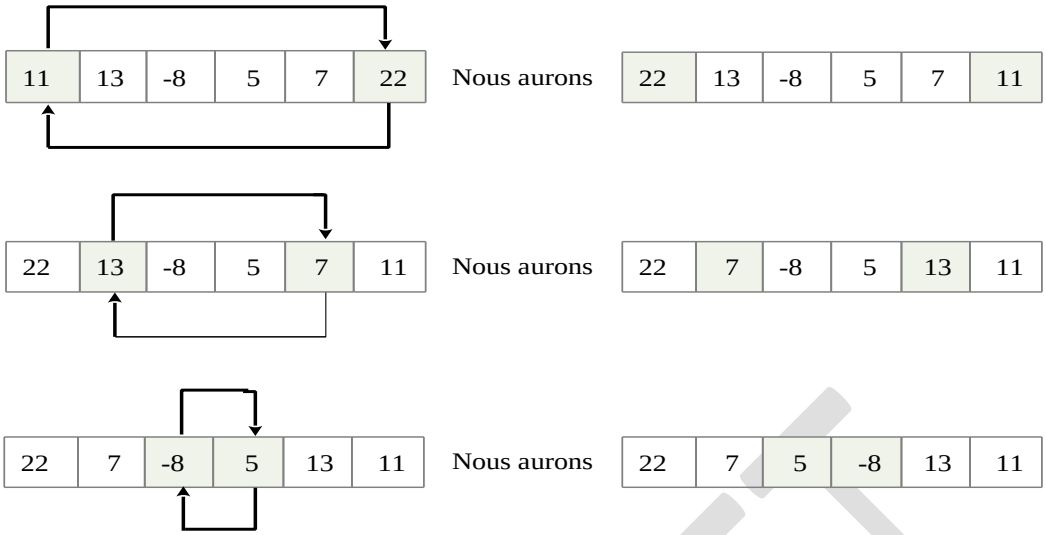
Le même problème que la question 01, sauf que, dans cette deuxième question, nous voulons inverser le vecteur T dans lui-même.

Explication ☺

On reprend le même exemple précédent :

N = 6, T = [11 13 -8 5 7 22]

Pour inverser les éléments de T, dans le même vecteur, nous allons faire les permutations suivantes :



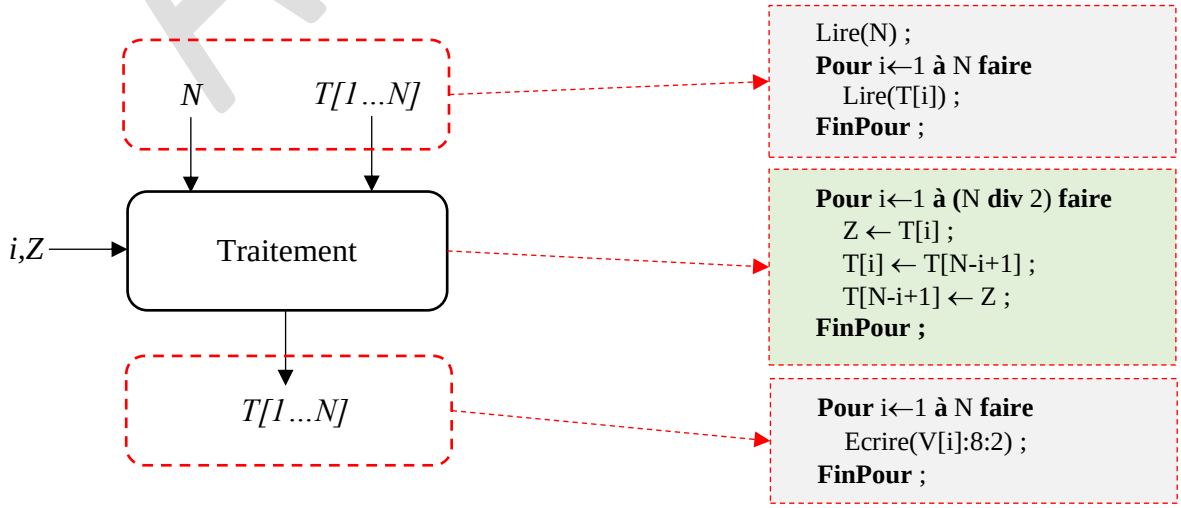
Après la troisième permutation, nous aurons le résultat demandé (inverser le vecteur T dans lui-même).

Remarques à retenir :

- Inverser un vecteur dans lui-même en permutant des cases.
- Le nombre de permutations est la moitié du vecteur.
- Les permutations : (1 avec N), (2 avec N-1), … Jusqu'à (N **div** 2).
- De la question 1, nous déduisons que : la case N° i sera permutée avec la case N° (N-i+1), tel-que i = 1… (N **div** 2).
- Pour permuter entre les cases *i* et (N-i+1), nous utilisons une troisième variable Z, comme suit :

```
Z ←T[i]
T[i] ← T[N-i+1]
T[N-i+1] ← Z
Pour chaque valeur de i allant de 1 à (N div 2)
```

Les variables d'entrée, variable de sorite et la partie traitement sont présentées dans le schéma ci-dessous :



Algorithme	Programme PASCAL

Algorithme inverser_T_dans_T;
Variables
i, N : **entier**;
T : Tableau [1..100] **de réel**;
Z : **réel**;
Début
{-*-*-*- Entrées -*-*-*-}
Ecrire('Donner la taille du vecteur T :');
Lire(N);
Ecrire('Donner les composantes de T :');
Pour i←1 **à** N **faire**
Lire(T[i]);
FinPour ;
 	
{-*-*-*- Traitement -*-*-*-}
Pour i←1 **à** (N div 2) **faire**
Z ← T[i];
T[i] ← T[N-i+1];
T[N-i+1] ← Z;
FinPour ;
 	
{-*-*-*- Sorties -*-*-*-}
Ecrire('L"inverse de T est :');
Pour i←1 **à** N **faire**
Ecrire(T[i]:8:2);
FinPour ;
Fin.
	Program inverser_T_dans_T;
	Var
	i, N : **integer**;
	T : **array** [1..100] **of real**;
	Z : **real** ;
	Begin
	{-*-*-*- Entrées -*-*-*-}
	Writeln('Donner la taille du vecteur T :');
	Read(N);
	Writeln('Donner les composantes de T :');
	For i := 1 **to** N **do**
	Read(T[i]);
	
	{-*-*-*- Traitement -*-*-*-}
	For i := 1 **to** (N div 2) **do**
	Begin
	Z := T[i];
	T[i] := T[N-i+1];
	T[N-i+1] := Z;
	End;
	
	{-*-*-*- Sorties -*-*-*-}
	Writeln('L"inverse de T est :');
	For i := 1 **to** N **do**
	Write(T[i]:8:2) ;
	End.

A.OUARET

```
1  Program inverser_T_dans_T;
2  Var
3    i,N : integer;
4    T : Array [1..100] of real;
5    Z : real ;
6  Begin
7    {-*-*-*- Entrées -*-*-*-}
8    Writeln('Donner la taille du vecteur T :');
9    Read(N);
10   Writeln('Donner les composantes de T :');
11   For i := 1 to N do
12     Read( T[i] );
13
14   {-*-*-*- Traitement -*-*-*-}
15   For i := 1 to (N div 2) do
16     Begin
17       Z := T[i];
18       T[i] := T[N-i+1];
19       T[N-i+1] := Z;
20     End;
21
22   {-*-*-*- Sorties -*-*-*-}
23   Writeln('L''inverse de T est :');
24   For i := 1 to N do
25     Write( T[i]:8:2) ;
26 End.
```

```
MyPascal V1.20.5 (Exécution) C:\Users\Ahmed\Deskt...
Donner la taille du vecteur T :
6
Donner les composantes de T :
11 13 -8 5 7 22
L'inverse de T est :
   22.00    7.00    5.00   -8.00   13.00   11.00
```

Après l'exécution

A. OUA

I.6. Exercices supplémentaires

Exercice supplémentaire 01 : La recherche d'une valeur dans un vecteur.

Soit V un vecteur de type réel de taille N.

Ecrire un algorithme/programme PASCAL qui permet de rechercher si une valeur réelle X existe ou non dans le vecteur V. Dans le cas ou X existe dans V, on affiche aussi sa position.

Exercice supplémentaire 02 : Somme, Produit et compteur d'éléments

Soit V un vecteur de type réel et de taille N.

Écrire un algorithme / Programme PASCAL qui permet de :

– Réaliser la somme des éléments divisibles par 3 et non divisible par 4.

– Réaliser le produit des éléments divisible par 4 et non divisible par 3.

– Compter le nombre d'éléments non-divisibles par 3 et non-divisibles par 4.

Exercice supplémentaire 03 : Convertir un nombre de base 10 vers base 2

Soit Nb un nombre entier positif écrit en base 10.

Ecrire un algorithme / programme PASCAL qui permet de convertir la valeur de Nb en base 2 et d'enregistrer les chiffres binaires de Nb dans un vecteur T.

Exercice supplémentaire 04 :

Ecrire un algorithme permettant de supprimer une case dont la position est lue à partir du clavier dans un tableau de dix entiers. La case supprimée sera substituée par un zéro ajouté à la fin du tableau.

Par exemple, après la suppression de la troisième case du tableau : 3, 15, 8, 18, 34, 1, 0, 4, 5, 21, le tableau devient : 3, 15, 18, 34, 1, 0, 4, 5, 21, 0.

Traduire l'algorithme en Pascal.

Exercice supplémentaire 05 :

Ecrire un algorithme permettant de résoudre le problème suivant :

– Données : un tableau contenant 100 entiers

– Résultat : "vrai" si le tableau est trié du plus petit au plus grand et "faux" sinon

Exercice supplémentaire 06 :

Ecrire un algorithme permettant de saisir 100 valeurs et qui les range au fur et à mesure dans un tableau.

A.OUARET

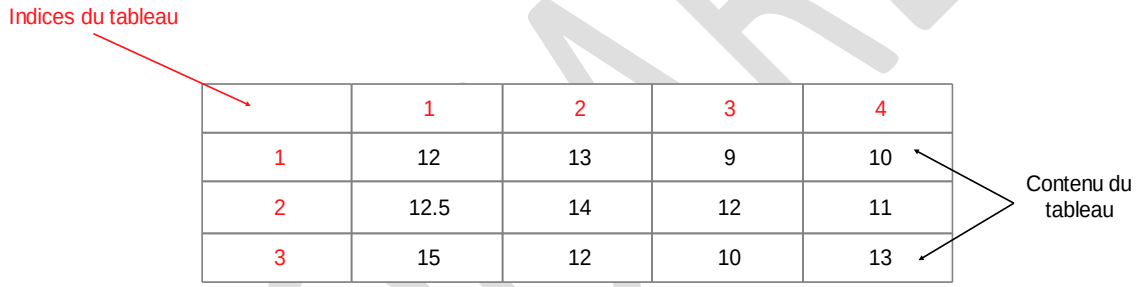
I.7. Tableaux à deux dimensions

Reprenons l'exemple des notes en considérant cette fois qu'un étudiant a plusieurs notes (une note pour chaque matière). Pour quatre étudiants, nous aurions le tableau de relevés des notes suivant :

	Etudiant 1	Etudiant 2	Etudiant 3	Etudiant 4
Informatique	12	13	9	10
Physique	12.5	14	12	11
Mathématiques	15	12	10	13

Les tableaux à deux dimensions se représentent comme une matrice ayant un certain nombre de lignes (première dimension) et un certain nombre de colonnes (seconde dimension).

Nous pouvons représenter schématiquement un tableau de 3 lignes et de 4 colonnes comme suit :



I.7.1. Déclaration

Syntaxe :

Algorithme

```
Variable
Variable identificateur : tableau [1..nb_lignes, 1..nb_colonnes] de type ;
Ou bien
Variable
Variable identificateur : tableau [nb_lignes, nb_colonnes] de type ;
```

Pascal

```
Var
Variable identificateur : array [1..nb_lignes, 1..nb_colonnes] of type ;
Ou bien
Var
Variable identificateur : array [nb_lignes, nb_colonnes] of type ;
```

Exemple :

L'instruction suivante déclare un tableau Note de type réel à deux dimensions composé de 3 lignes et de 4 colonnes :

Algorithme	Pascal
Variable Note : tableau [1..3, 1..4] de réels ;	**Var** Note : array [1..3, 1..4] of real ;

I.7.2. Utilisation

Pour accéder à un élément de la matrice (tableau à deux dimensions), il suffit de préciser, entre crochets, les indices de la case contenant cet élément.

Les éléments de la matrice peuvent être utilisés comme n'importe quelle variable.

Exemple :

L'instruction suivante affecte à la variable X la valeur du premier élément du tableau Note :

Algorithme	Pascal
X ← Note [1,1] ;	X := Note [1,1] ;

I.7.3. Lecture et affichage d'une matrice

L'algorithme (respectivement, le programme) suivant permet de lire et d'écrire une matrice de *n* ligne et *m* colonnes :

Algorithme	Pascal

Algorithme LectureAffichageMatrice
Variables
mat : tableau [1..10, 1..10] de reel
N,M, i,j:entier
Début
Lire (N, M);
pour i=1 **à** N **faire**
Pour j=1 **à** M **faire**
Lire (mat[i, j])
FinPour
FinPour
pour i=1 **à** N **faire**
Pour j=1 **à** M **faire**
Ecrire (mat[i, j])
FinPour
FinPour
Fin.
	Program LectureAffichageMatrice;
	var
	mat : **array** [1..10, 1..10] **of** real;
	N,M, i,j : integer;
	Begin
	Read(N, M);
	For i:=1 **To** N **Do**
	For j:=1 **To** M **Do**
	Read(mat[i, j]);
	For i:=1 **To** N **Do**
	Begin
	For j:=1 **To** M **Do**
	Begin
	Write(mat[i,j],' ');
	End;
	writeln;
	End;
	End.

A.OUARET

I.8. Exercices avec corrigés

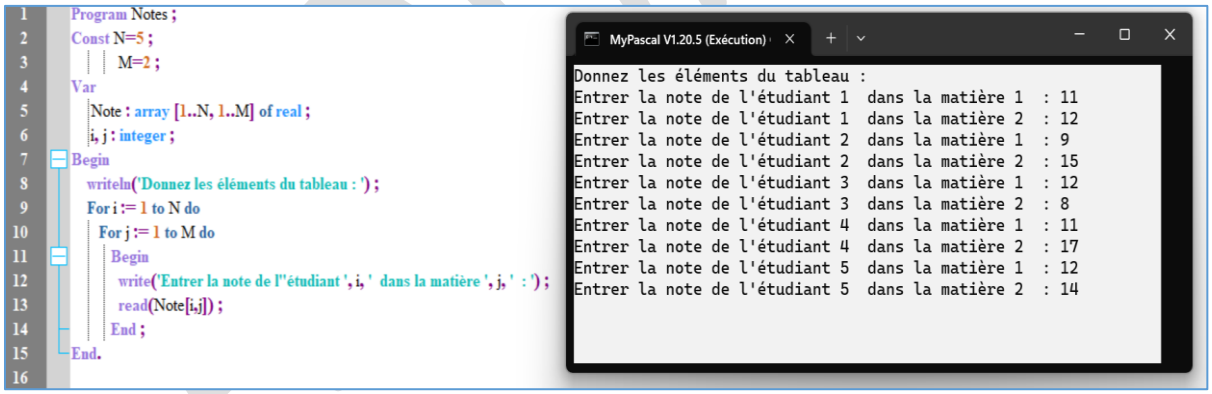
Exercice N°19 :

Ecrire un algorithme permettant la saisie des notes d'une classe de 5 étudiants en 2 matières.

Corrigé de l'exercice N°19 :

Algorithme	Pascal

Algorithme Notes ;
Constante N=5 ;
M=2 ;
Variables
Note : tableau [1..N, 1..M] de réels ;
i, j : entier ;
Début
Ecrire('Donnez les éléments du tableau : ') ;
Pour i ← 1 **à** N **faire**
Pour j ← 1 **à** M **faire**
Ecrire('Entrer la note de l''étudiant ', i, '
dans la matière', j, ' : ') ;
Lire(Note[i,j]) ;
Fin-Pour ;
Fin-Pour ;
Fin.
	Program Notes ;
	Const N=5 ;
	M=2 ;
	Var
	Note : array [1..N, 1..M] of real ;
	i, j : integer ;
	Begin
	writeln('Donnez les éléments du tableau : ') ;
	For i := 1 **to** N **do**
	For j := 1 **to** M **do**
	Begin
	write('Entrer la note de l''étudiant ', i, '
	dans la matière ', j, ' : ') ;
	read(Note[i,j]) ;
	End ;
	End.



Exercice N°20 :

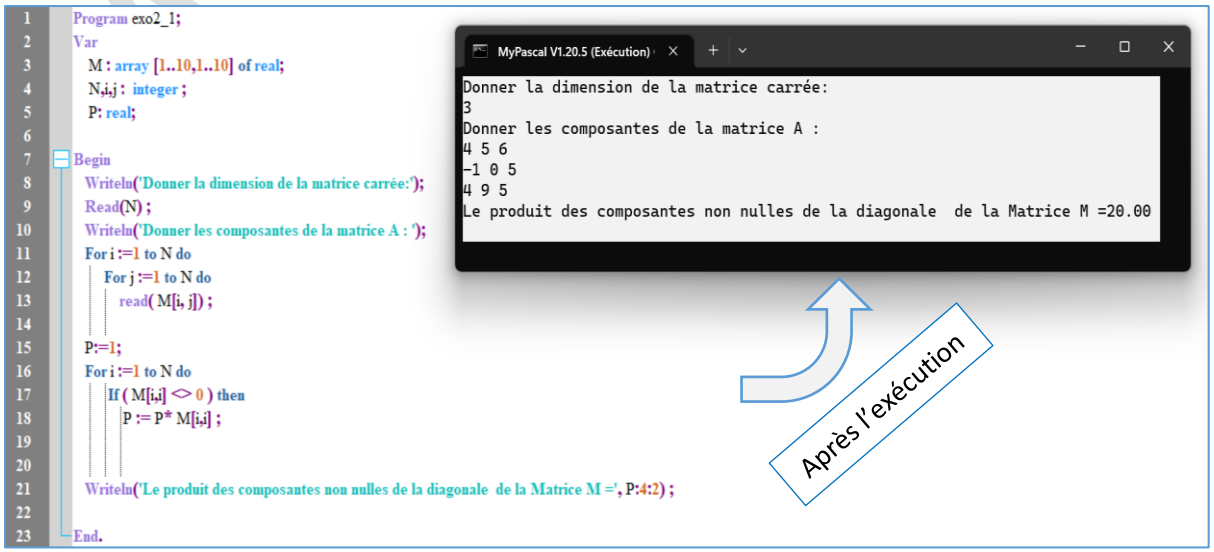
Soit M une matrice carrée de taille N x N et de type réel.

Écrire un programme pascal qui permet de calculer le produit des composantes non nulles de la diagonale principale de la matrice M.

Corrigé de l'exercice N°20 :

Algorithme	Pascal

Algorithme exo2_1;
Variables
M : tableau [1..10,1..10] de réel;
N,i,j : entier ;
P: réel;
 	
Début
Ecrire('Donner la dimension de la matrice carrée:');
Lire(N) ;
Ecrire('Donner les composantes de la matrice A : ');
Pour i ←1 **à** N **faire**
Pour j ←1 **à** N **faire**
Lire(M[i, j]) ;
Fin-Pour;
Fin-Pour;
 	
P ← 1 ;
Pour i←1 **à** N **faire**
Si (M[i,i] <> 0) **alors**
P ← P* M[i,i] ;
Fin-si;
Fin-Pour;
 	
Écrire ('Le produit des composantes non nulles de la diagonale de la Matrice M = ', P:4:2) ;
Fin.
	Program exo2_1;
	Var
	M : array [1..10,1..10] of real;
	N,i,j : integer ;
	P: real;
	
	Begin
	Writeln('Donner la dimension de la matrice carrée:');
	Read(N) ;
	Writeln('Donner les composantes de la matrice A : ');
	For i :=1 **to** N **do**
	For j :=1 **to** N **do**
	read(M[i, j]) ;
	
	P:=1;
	For i :=1 **to** N **do**
	If (M[i,i] <> 0) **then**
	P := P* M[i,i] ;
	
	Writeln('Le produit des composantes non nulles de la diagonale de la Matrice M =', P:4:2) ;
	
	End.



Exercice N°21 :

Soit A une matrice réelle d'ordre N x M.

Écrire un algorithme/programme PASCAL qui permet de rechercher le plus petit élément dans la matrice A ainsi que sa position.

Corrigé de l'exercice N°21 :

Algorithme	Pascal

Algorithme exo2_1
Variables
A : Tableau [1..10, 1..10] **de** Réel
i, j, N, M, imin, jmin : entier
Min : Réel
Début
Ecrire('Donner les dimensions de la Matrice A : ');
Lire(N, M)
Ecrire('Donner les composantes de la matrice A : ');
Pour i ←1 **à** N **faire**
Pour j ←1 **à** M **faire**
Lire(A[i, j])
Fin-Pour
Fin-Pour
min ← A[1, 1]
imin ← 1
jmin ← 1
Pour i ←1 **à** N **faire**
Pour j ←1 **à** M **faire**
Si A[i, j] < min **alors**
min ← A[i, j]
imin ← i
jmin ← j
Fin-Si
Fin-Pour
Fin-Pour
Ecrire('min=', min:3:2, 'et sa position est : ', imin,' ',jmin);
Fin
	Program exo2_1;
	Var
	A : array[1..10, 1..10] **of** Real;
	i, j, N, M, imin, jmin : integer;
	min : real;
	Begin
	Write('Donner les dimensions de la Matrice A : ');
	Read(N, M);
	Writeln('Donner les composantes de la matrice A : ');
	For i:=1 **to** N do
	For j:=1 **to** M **do**
	Read(A[i, j]);
	
	min := A[1, 1]; {On suppose que la première case est le min}
	imin:= 1; {Donc sa position est la ligne 1}
	jmin := 1; {et la colonne 1}
	For i:=1 **to** N **do** {Pour toute ligne i}
	For j:=1 **to** M **do** {Pour toute colonne j}
	if A[i, j] < min **Then**
	Begin
	min := A[i,j]; {On actualise le Min}
	imin := i; {On actualise sa ligne imin}
	jmin := j; {On actualise sa colonne jmin}
	end;
	{*-* Sorties *-*}
	Write('min=', min:3:2, 'et sa position est : ', imin,' ',jmin);
	End.

A.OUARET

```
1  Program exo2_1;
2  Var
3    A : array[1..10, 1..10] of Real;
4    i, j, N, M, imin, jmin : integer;
5    min : real;
6  Begin
7   Write('Donner les dimensions de la Matrice A : ');
8    Read(N, M);
9    Writeln('Donner les composantes de la matrice A : ');
10   For i:=1 to N do
11    For j:=1 to M do
12     Read( A[i, j] );
13   min := A[1, 1]; {On suppose que la première case est le min}
14   imin:= 1; {Donc sa position est la ligne 1}
15   jmin := 1; {et la colonne 1}
16   For i:=1 to N do {Pour toute ligne i}
17    For j:=1 to M do {Pour toute colonne j}
18     if A[i, j] < min Then
19     Begin
20       min := A[i,j]; {On actualise le Min}
21       imin := i;  {On actualise sa ligne imin}
22       jmin := j; {On actualise sa colonne jmin}
23     end;
24   Write('min=', min:3:2, 'et sa position est : ', imin,'    ',jmin);
25   End.
```

```
MyPascal V1.20.5 (Exécution)
Donner les dimensions de la Matrice A : 3 4
Donner les composantes de la matrice A :
2 3 5 6
-4 5 9 7
1 2 3 11
min=-4.00et sa position est : 2    1
```

Après l'exécution

Exercice N°22 :

Écrire un algorithme/programme PASCAL qui permet de calculer la somme de chaque ligne et le produit de chaque colonne.

Corrigé de l'exercice N°22 :

Algorithme	Pascal
Algorithme exol; **Variables** A : Tableau [1..100,1..100] de réels; N,M,i,j : Entier; S:Tableau[1..100] de réels; {On déclare S comme vecteur qui contient la somme de chaque ligne parce que on ne connait pas la taille de la matrice A} P:Tableau[1..100] de réels; {On déclare P comme vecteur qui contient le produite chaque colonne parce que on ne connait pas la taille de la matrice A} **Début** {Début du programme} {*-* Entrées *-*} Ecrire('Donner la taille de la matrice A');	**Program** exol; **Var** A : array [1..100,1..100] of real; N,M,i,j : integer; S:array[1..100] of real; {On déclare S comme vecteur qui contient la somme de chaque ligne parce que on ne connait pas la taille de la matrice A} P:array[1..100] of real; {On déclare P comme vecteur qui contient le produite chaque colonne parce que on ne connait pas la taille de la matrice A} **Begin** {Début du programme} {*-* Entrées *-*} Writeln('Donner la taille de la matrice A');

```
Lire(N,M);
   Ecrire('Donner les composantes de la matrice A');
  Pour i ←1 à N faire
    Pour j ←1 à M faire
      Lire( A[i, j] )
    Fin-Pour
  Fin-Pour
  {*-* Traitement *-*}

  Pour i ←1 à N faire
        S[i] ←0;
          Pour j ←1 à M faire
            S[i] ← S[i]+A[i,j];
          Fin-Pour
  Fin-Pour

  Pour j ←1 à M faire
       P[j] ← 1;
       For i ←1 to N do
            P[j] ←P[j]*A[i,j];
       Fin-Pour
  Fin-Pour
  {*-* Sorties *-*}
  Ecrire('La somme de chaque ligne est : ');
  Pour i ←1 à N faire
    Ecrire(S[i]:6:2);
  FinPour


  Ecrire('Le produit de chaque colonne est : ');
   Pour j ←1 à M faire
       Ecrire(P[j]:6:2);
Fin.
```

```
  Read(N,M);
  Writeln('Donner les composantes de la
          matrice A');
  For i:=1 to N do
     For j:=1 to M do
         Read(A[i,j]);

  {*-* Traitement *-*}
  For i:=1 to N do
     begin
        S[i]:=0;
          For j:=1 to M do
            S[i]:=S[i]+A[i,j];
     End;

        For j:=1 to M do
         begin
            P[j]:=1;
            For i:=1 to N do
              P[j]:=P[j]*A[i,j];
         End;


  {*-* Sorties *-*}
  Writeln('La somme de chaque ligne est : ');
  For i:=1 to N do
  Write(S[i]:6:2);

  Writeln; {J'ai ajouté cette instruction juste
           pour sauter la ligne (ni pas
           obligatoire)}

  Writeln('Le produit de chaque colonne est : ');
     For j:=1 to M do
        Write(P[j]:6:2);
End. {Fin du programme}
```

A.OUARET

Exercice N°23 :

Écrire un algorithme/programme PASCAL qui permet de compter le nombre d'éléments négatifs, positifs et nuls dans une matrice.

Corrigé de l'exercice N°23 :

Algorithme	Pascal

Algorithme elements_P_N_N ;
Variables
mat : tableau [1..10, 1..10] de réel ;
N,M,i,j:entier ;
nbP, nbN, nbNul : entier ;
Début
Ecrire('Donner la dimension de la matrice mat');
Lire (N, M);
Ecrire('Donner les composantes de la matrice mat');
pour i ← 1 **à** N **faire**
Pour j ← 1 **à** M **faire**
Lire (mat[i, j])
FinPour
FinPour
nbP ← 0; nbN ← 0; nbNul ← 0;
pour i ← 1 **à** N **faire**
Pour j ← 1 **à** M **faire**
Si mat[i, j] > 0 **Alors**
nbP ← nbP + 1;
Sinon
Si mat[i,j] < 0 **Alors**
nbN ← nbN + 1;
Sinon
nbNul ← nbNul + 1;
FinSi
FinSi
FinPour
FinPour
Ecrire('Nombre d''éléments positifs : ', nbP) ;
Ecrire('Nombre d''éléments négatifs : ', nbN) ;
Ecrire('Nombre d''éléments nuls : ', nbNul) ;
Fin.
	Program elements_P_N_N ;
	var
	mat :**array** [1..10, 1..10] **of** real;
	N,M,i,j : integer;
	nbP, nbN, nbNul : integer;
	Begin
	Writeln('Donner la dimension de la matrice mat');
	Read(N, M);
	Writeln('Donner les composantes de la matrice mat');
	For i:=1 **To** N **Do**
	For j:=1 **To** M **Do**
	Read(mat[i, j]);
	nbP:=0; nbN:=0; nbNul:=0;
	For i:=1 **To** N **Do**
	For j:=1 **To** M **Do**
	if mat[i,j]>0 **then**
	nbP:=nbP+1
	else
	if mat[i,j]<0 **then**
	nbN:=nbN+1
	else
	nbNul:=nbNul+1;
	Writeln('Nombre d''éléments positifs : ', nbP) ;
	Writeln('Nombre d''éléments négatifs : ', nbN) ;
	Write('Nombre d''éléments nuls : ', nbNul) ;
	End.

A OUARET

```
4    N,M,i,j : integer;
5    nbP, nbN, nbNul : integer;
6  Begin
7    Writeln('Donner la dimension de la matrice mat');
8  Read(N, M);
9  Writeln('Donner les composantes de la matrice mat');
10 For i:=1 To N Do
11   For j:=1 To M Do
12     Read(mat[i, j]);
13 nbP:=0; nbN:=0; nbNul:=0;
14 For i:=1 To N Do
15   For j:=1 To M Do
16     if mat[i,j]>0 then
17       nbP:=nbP+1
18     else
19     if mat[i,j]<0 then
20       nbN:=nbN+1
21     else
22       nbNul:=nbNul+1;
23 Writeln('Nombre d''éléments positifs : ', nbP) ;
24 Writeln('Nombre d''éléments négatifs : ', nbN) ;
25 Write('Nombre d''éléments nuls : ', nbNul) ;
26 End.
```

```
MyPascal V1.20.5 (Exécution)
Donner la dimension de la matrice mat
3 4
Donner les composantes de la matrice mat
1 2 0 7
-4 4 0 1
-4 -1 2 3
Nombre d'éléments positifs : 7
Nombre d'éléments négatifs : 3
Nombre d'éléments nuls : 2
```

Après l'exécution

Exercice N°24 :

Dans le cadre d'étude du comportement thermomécanique d'un alliage métallique, l'étudiant a besoin de déterminer la dureté de chacune de ces **5** éprouvettes. Pour chaque éprouvette, il mesure la dureté **3** fois. Il note ses résultats dans un tableau $(\boldsymbol{T_b})$ en brouillon puis il dessine sur son compte rendu un nouveau tableau $(\boldsymbol{T_f})$ en une seule ligne contenant la moyenne des **3** duretés mesurées pour chacune des éprouvettes.

Ecrire un algorithme qui permet de saisir, traiter et afficher ces deux tableaux $(\boldsymbol{T_b}\ et\ \boldsymbol{T_f})$ de mesure de dureté.

Corrigé de l'exercice N°24 :

Algorithme	Pascal

Algorithme mesure_durete ;
Variables
Tb : tableau [1..5,1..3] de réels ;
Tf : tableau [1..5] de réels ;
S : réel ;
i, j :entier ;
Début
Pour i **de** 1 **à** 5 **faire**
S ← 0 ;
Pour j **de** 1 **à** 3 **faire**
Ecrire('Introduisez la',j, 'ème mesure de la',i, 'ème éprouvette') ;
Lire(Tb[i,j]);
S← S+Tb[i,j];
	program mesure_durete ;
	Var
	Tb : array [1..5,1..3] of real ;
	Tf : array [1..5] of real ;
	S : real ;
	i, j :integer ;
	Begin
	For i := 1 **to** 5 **do**
	Begin
	S := 0 ;
	For j :=1 **to** 3 **do**
	Begin
	write('Introduisez la ',j, ' ème mesure de la',i, ' ème éprouvette :') ;

```
    Fin pour
    Tf[i] ← S/3 ;
  Fin pour
  Ecrire('Le tableau des résultats bruts est le
          suivant : ') ;
  Pour i de 1 à 5 faire
    Pour j de 1 à 3 faire
      Écrire(Tb[i,j]:8:2) ;
    Fin pour
  Fin pour
  Ecrire('Le tableau final des résultats est le
          suivant : ') ;
  Pour i de 1 à 5 faire
    Écrire(Tf[i]:8:2) ;
  Fin pour
Fin.
```

```
      read(Tb[i,j]);
      S := S+Tb[i,j];
    End;
    Tf[i] := S/3 ;
  End ;
  writeln('Le tableau des résultats bruts est le
          suivant : ') ;
  For i :=1 to 5 do
    For j := 1 to 3 do
      write(Tb[i,j]:8:2) ;
  writeln ; {Pour sauter la ligne}
  writeln('Le tableau final des résultats est le
          suivant : ') ;
  For i :=1 to 5 do
    write(Tf[i]:8:2) ;
End.
```

Exercice N°25 :

Ecrire un programme pascal qui permet de calculer et afficher la somme des éléments situés au-dessus de la diagonale d'une matrice carrée.

Corrigé de l'exercice N°25 :

Pascal (méthode 1)

```
Program Somme_elements_dessus_diagonale;
Var
A : array [1..50,1..50] of integer;
Somme, i, j, N  : integer;
Begin
  {-*-*-*- Entrées -*-*-*-}
  read(N) ;
  For i:=1 to N do
    For j:=1 to N do
      read (A[i, j]);
  {-*-*-*- Entrées -*-*-*-}
  Somme:=0;
  For i :=1 to N-1 do
    For j :=i+1 to N do
       Somme:= Somme + A[i,j];
  {-*-*-*- Sorties -*-*-*-}
  Write ('La somme des éléments situés au-
          dessus de la diagonale est : ', Somme)
          ;
End.
```

Pascal (méthode 2)

```
Program Somme_elements_dessus_diagonale;
Var
A : array [1..50,1..50] of integer;
Somme, i, j, N  : integer;
Begin
  {-*-*-*- Entrées -*-*-*-}
  read(N) ;
  For i:=1 to N do
    For j:=1 to N do
      read (A[i, j]);
 {-*-*-*- Entrées -*-*-*-}
  Somme:=0;
  For i :=1 to N do
    For j :=1 to N do
      If (i<j)  then
        Somme:= Somme + A[i,j];
 {-*-*-*- Sorties -*-*-*-}
  Write ('La somme des éléments situés au-
          dessus de la diagonale est : ', Somme)
          ;
End.
```

A. OUARET

Exercice N°26 :

Ecrire un programme pascal qui permet de calculer et afficher la somme des éléments situés en dessous de la diagonale d'une matrice carrée.

Corrigé de l'exercice N°26 :

Pascal (méthode 1)	Pascal (méthode 2)

Program Somme_elements_dessus_diagonale;
Var
A : array [1..50,1..50] of integer;
Somme, i, j, N : integer;
Begin
{-*-*-*- Entrées -*-*-*-}
read(N) ;
For i:=1 **to** N **do**
For j:=1 **to** N **do**
read (A[i, j]);
{-*-*-*- Entrées -*-*-*-}
Somme:=0;
For i :=2 **to** N **do**
For j :=1 **to** i-1 **do**
Somme:= Somme + A[i,j];
{-*-*-*- Sorties -*-*-*-}
Write ('La somme des éléments situés au-dessus de la diagonale est : ', Somme);
End.
	Program Somme_elements_dessous_diagonale;
	Var
	A : array [1..50,1..50] of integer;
	Somme, i, j, N : integer;
	Begin
	{-*-*-*- Entrées -*-*-*-}
	read(N) ;
	For i:=1 **to** N **do**
	For j:=1 **to** N **do**
	read (A[i, j]);
	{-*-*-*- Entrées -*-*-*-}
	Somme:=0;
	For i :=1 **to** N **do**
	For j :=1 **to** N **do**
	If (i>j) **then**
	Somme:= Somme + A[i,j];
	{-*-*-*- Sorties -*-*-*-}
	Write ('La somme des éléments situés au-dessus de la diagonale est : ', Somme) ;
	End.

A.OUARET

I.9. Exercices supplémentaires

Exercice supplémentaire 01 : Somme de deux matrices

Ecrire un algorithme/programme PASCAL qui permet de réaliser la somme de deux matrices réelles A et B d'ordre N x M.

Exercice supplémentaire 02 : Somme, Moyenne et Produit des éléments d'une matrice

Soit une matrice A réelle d'ordre N x M.

1. Ecrire un algorithme/programme PASCAL qui calcule la somme et la moyenne des éléments de la matrice A.
2. Ecrire un algorithme/programme PASCAL qui permet de calculer la somme de chaque ligne et le produit de chaque colonne.

Exercice supplémentaire 03 : Produit de deux matrices

Soit A et B deux matrices carrées d'ordre N.

Ecrire un algorithme/programme PASCAL qui permet de calculer le produit de A et B.

Exercice supplémentaire 04 : La recherche d'une valeur dans une matrice

Soit M une matrice de type réel de taille N x M.

Ecrire un algorithme/programme PASCAL qui permet de rechercher si une valeur réelle X existe ou non dans la matrice M. Dans le cas où X existe dans M, on affiche aussi sa position (numéro de ligne et de colonne).

Exercice supplémentaire 05 : Le Min et le Max dans une matrice et leurs positions

Soit A une matrice réelle d'ordre N x M.

1. Ecrire un algorithme/programme PASCAL qui permet de rechercher le plus petit élément dans la matrice A ainsi que sa position.
2. Ecrire un algorithme/programme PASCAL qui permet de rechercher le plus grand élément dans la matrice A ainsi que sa position.

A.OUARET