

TP Informatique 2

Corrigé de la série de TP N°1 – Tableaux (Vecteurs & Matrices)

Partie A :

Exercice N°01 : Algorithme → Programme PASCAL

Soit l'algorithme suivant :

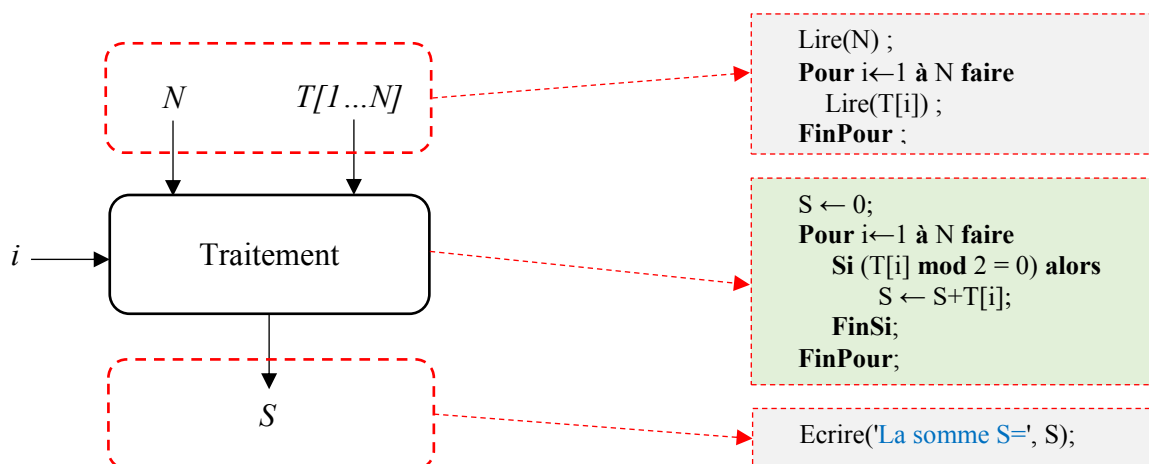
```

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]);
    FinPour;
    {-*-*- Traitements -*-*-}
    S ← 0;
    Pour i←1 à N faire
        Si (T[i] mod 2 = 0) alors
            S ← S+T[i];
        FinSi;
    FinPour;
    {-*-*- Sorties -*-*-}
    Ecrire('La somme S=', S);
Fin.
    
```

- Questions :**
- 1- Traduire l'algorithme en Programme PASCAL.
 - 2- Compiler et exécuter le programme pour :
N = 4 et T=[14, 3, 8, 22].
 - 3- Dérouler l'algorithme pour les valeurs de N et T ci-dessus ?
 - 4- Déduire ce que fait l'algorithme ?
 - 5- Ré-écrire le programme en remplaçant la boucle *Pour* par la boucle *Tant-que* dans la partie **traitements**.
 - 6- Ré-écrire le programme en remplaçant la boucle *Pour* par la boucle *Répéter* dans la partie **traitements**.

Solution :

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



Remarque :
 La variable *i* est une variable de traitement ou intermédiaire, utilisée pour parcourir le vecteur T.


1- Traduire l'algorithme en programme PASCAL

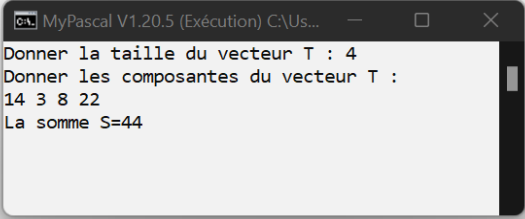
Algorithme	Programme Pascal
Algorithme Vecteur ; Variables T : Tableau [1..100] d'entier ; N, i, S : entier; Début {-*-*- Entrées -*-*-} Ecrire('Donner la taille du vecteur T : '); Lire(N); Ecrire('Donner les composantes du vecteur T : '); Pour i←1 à N faire Lire(T[i]); Fin-Pour ; {-*-*- Traitements -*-*-} S ← 0; Pour i←1 à N faire Si (T[i] mod 2 = 0) alors S ← S+T[i]; FinSi ; FinPour ; {-*-*- 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 {Lecture des éléments du vecteur T} Read(T[i]); {-*-*- Traitements -*-*-} S:=0; For i:=1 to N do {recherche les multiples de 2} if (T[i] mod 2 = 0) then S:=S+T[i]; {La somme des multiples de 2} {-*-*- Sorties -*-*-} Write('La somme S=',S); {Affichage de la somme des multiples de 2} End.

2- Compiler et exécuter le programme pour : N = 6 et T=[14, 3, 8, 22].

```

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 {Lecture des éléments du vecteur T}
12     Read(T[i]);
13
14  {-*-*- Traitements -*-*-}
15  S:=0;
16  For i:=1 to N do {recherche les multiples de 2}
17     if (T[i] mod 2 = 0) then
18        S:=S+T[i]; {La somme des multiples de 2}
19
20  {-*-*- Sorties -*-*-}
21  Write('La somme S=',S); {Affichage de la somme des multiples de 2}
22 End.
```





3- Dérouler l'algorithme pour : N = 6 et T=[14, 3, 8, 22].

Dérouler un algorithme (ou un programme) consiste à exécuter manuellement les instructions de cet algorithme et à visualiser l'impact de ces instructions sur les variables.

Autrement dit, dérouler un algorithme permet de visualiser les changements des valeurs des variables (évolution des valeurs).

Instructions	Variables				Affichage
	N	i	T	S	
Ecrire('Donner la taille du vecteur T : ');	/	/	/	/	Donner la taille du vecteur T :
Lire(N)	4	/	/	/	//
Ecrire('Donner les composantes du vecteur T : ');	//	/	/	/	Donner les composantes du vecteur T :
Pour i←1 à N faire Lire(T[i]) Fin-Pour	//	1 2 3 4	1 2 3 4 [14, 3, 8, 22]	/	//
S ← 0	//	/	//	0	//
Pour i=1 Si T[1] mod 2 = 0 Alors 14 mod 2 = 0 True S ← S+T[i]=S+T[1] S ← 0+14=14	//	1	//	14	//
Pour i=2 Si T[2] mod 2 = 0 Alors 3 mod 2 = 0 False	//	2	//	//	//
Pour i=3 Si T[3] mod 2 = 0 Alors 8 mod 2 = 0 True S ← S+T[i]=S+T[3] S ← 14+8=22	//	3	//	22	//
Pour i=4 Si T[4] mod 2 = 0 Alors 22 mod 2 = 0 True S ← S+T[i]=S+T[4] S ← 22+22=44	//	4	//	44	//
Ecrire('La somme S = ', S);	//		//	//	La somme S = 44

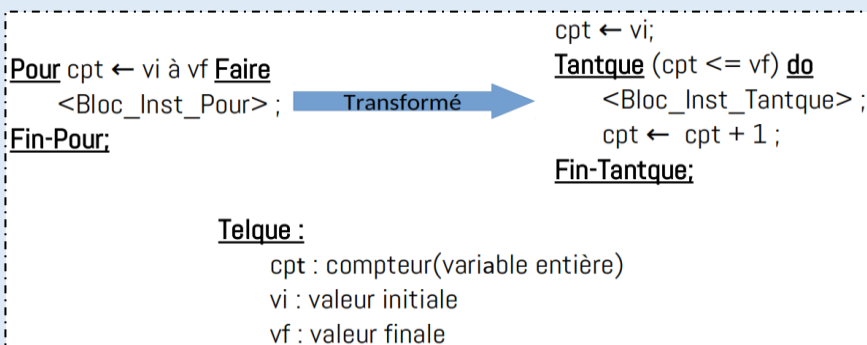
4- Dédire ce que fait l'algorithme

L'algorithme permet de réaliser la somme des composantes du vecteur T divisibles par 2 (la somme des nombres pairs).

5- Boucle Tant-que (While)

Rappel :

La boucle **Pour** possède un compteur, une valeur initiale et une valeur finale. Par contre, la boucle **Tant-que** possède une condition.



Algorithme/Programme PASCAL

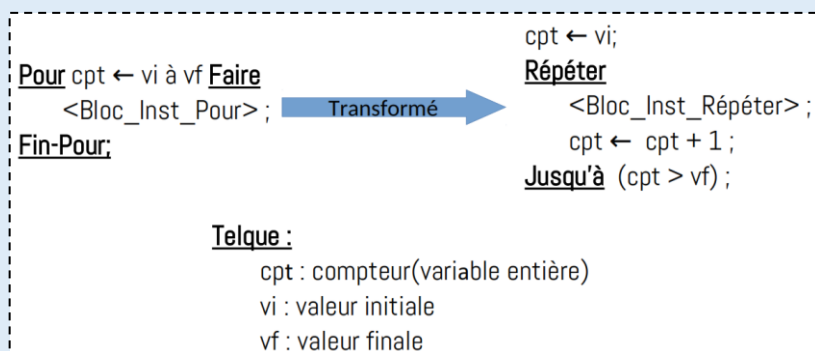
Algorithme	Programme PASCAL
<p>Algorithme Vecteur ;</p> <p>Variables T : Tableau [1..100] d'entier ; N, i, S : entier;</p> <p>Début</p> <p style="color: red;">{--*-*- Entrées --*-*-}</p> <p>Ecrire('Donner la taille du vecteur T : '); Lire(N); Ecrire('Donner les composantes du vecteur T : '); i ← 1 ;</p> <p>Tant-que (i<=N) faire Lire(T[i]); i ← i+1 ;</p> <p>Fin-Tant-que;</p> <p style="color: red;">{--*-*- Traitements --*-*-}</p> <p>S ← 0; i ← 1;</p> <p>Tant-que (i<=N) faire Si (T[i] mod 2 = 0) alors S ← S+T[i]; FinSi; i ← i+1;</p> <p>Fin-Tant-que;</p> <p style="color: red;">{--*-*- Sorties --*-*-}</p> <p>Ecrire('La somme S=', S);</p> <p>Fin.</p>	<p>program Vecteur ;</p> <p>Var T : array [1..100] of integer ; N, i, S : integer ;</p> <p>Begin</p> <p style="color: red;">{--*-*- Entrées --*-*-}</p> <p>Write('Donner la taille du vecteur T : '); Read(N); Writeln('Donner les composantes du vecteur T : '); i:=1;</p> <p>While (i<=N) do {Lecture des éléments du vecteur T} Begin Read(T[i]); i:=i+1; End;</p> <p style="color: red;">{--*-*- Traitements --*-*-}</p> <p>S:=0; i:=1;</p> <p>While (i<=N) do {Recherche des divisibles de 2} Begin if (T[i] mod 2 = 0) then S:=S+T[i]; {La somme des divisibles de 2} i:=i+1; End;</p> <p style="color: red;">{--*-*- Sorties --*-*-}</p> <p>Write('La somme S=',S); {Affichage de la somme des divisibles de 2}</p> <p>End.</p>

6- Boucle Répéter (Repeat)

Rappel :

De la même façon que la boucle **Tant-que**, la boucle **Répéter** possède une condition. Sauf que pour cette dernière (la boucle répéter), la condition représente la condition d'achèvement de la boucle.

Voici le modèle de transformation de la boucle *Pour* vers la boucle *Répéter* :



Algorithme/Programme PASCAL

Algorithme	Programme PASCAL
<p>Algorithme Vecteur ;</p> <p>Variabes T : Tableau [1..100] d'entier ; N, i, S : entier ;</p> <p>Début</p> <p>{-*-*- Entrées -*-*-}</p> <p>Ecrire('Donner la taille du vecteur T : '); Lire(N); Ecrire('Donner les composantes du vecteur T : '); i ← 1 ;</p> <p>Répéter Lire(T[i]); i ← i+1 ;</p> <p>Jusqu'à (i>N);</p> <p>{-*-*- Traitements -*-*-}</p> <p>S ← 0; i ← 1;</p> <p>Répéter Si (T[i] mod 2 = 0) alors S ← S+T[i]; FinSi; i ← i+1;</p> <p>Jusqu'à (i>N);</p> <p>{-*-*- Sorties -*-*-}</p> <p>Ecrire('La somme S=', S);</p> <p>Fin.</p>	<p>program Vecteur ;</p> <p>Var T : array [1..100] of integer ; N, i, S : integer ;</p> <p>Begin</p> <p>{-*-*- Entrées -*-*-}</p> <p>Write('Donner la taille du vecteur T : '); Read(N); Writeln('Donner les composantes du vecteur T : '); i:=1;</p> <p>Repeat {Lecture des éléments du vecteur T} Read(T[i]); i:=i+1;</p> <p>Until (i>N) ;</p> <p>{-*-*- Traitements -*-*-}</p> <p>S:=0; i:=1;</p> <p>Repeat {Recherche des divisibles de 2} if (T[i] mod 2 = 0) then S:=S+T[i]; {La somme des divisibles de 2} i:=i+1;</p> <p>Until (i>N) ;</p> <p>{-*-*- Sorties -*-*-}</p> <p>Write('La somme S=',S); {Affichage de la somme des divisibles de 2}</p> <p>End.</p>

Remarques :

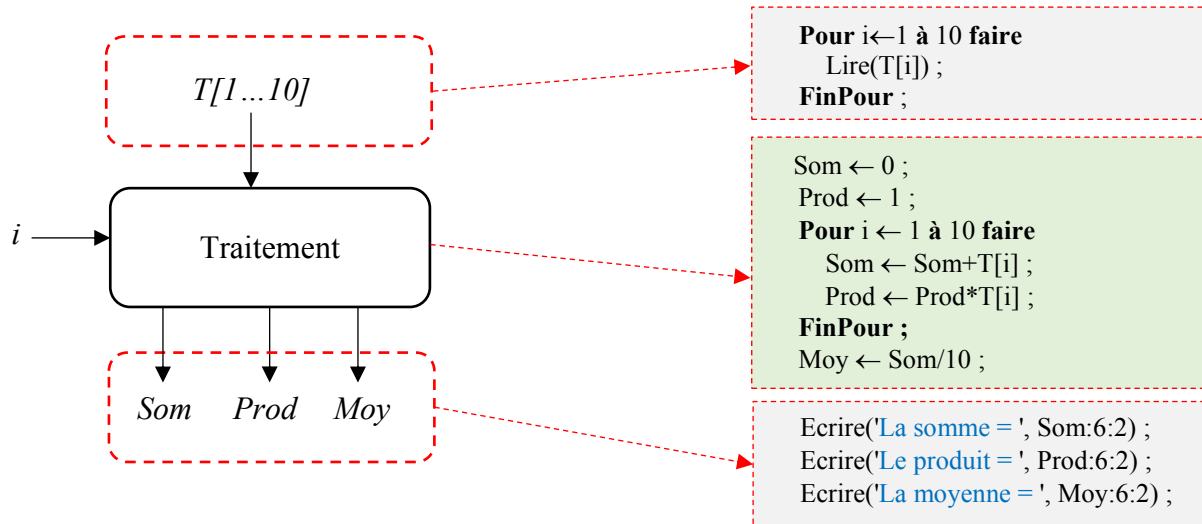
- Un vecteur (Tableau à une dimension) est une suite de cases mémoires adjacentes. Ces cases mémoires définissent des variables de même types.
- La déclaration suivante : **T : Tableau [1..100] d'entier;**
Signifie que nous réservons 100 cases de type entier. 100 est la taille maximale du vecteur T. Chaque case est accessible par un indice qui peut prendre les valeurs 1 à 100.
- Pendant l'exécution, nous n'utiliserons pas les **100** cases du vecteur, nous utiliserons **n** cases, où **n** est une variable entière qui doit être introduite (lue) pendant l'exécution.
- Pour lire un vecteur T, il faut lire la taille que l'on veut utiliser (la variable **n**) et lire toutes les cases **V[i]** tel-que **i** allant de **1** à la valeur **n**. Même chose pour l'affichage.

Exercice N°02 : La somme, le produit et la moyenne des éléments d'un tableau

Ecrire un algorithme/programme PASCAL qui permet de calculer la somme, le produit et la moyenne des éléments d'un vecteur V de dix réels.

Solution :

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

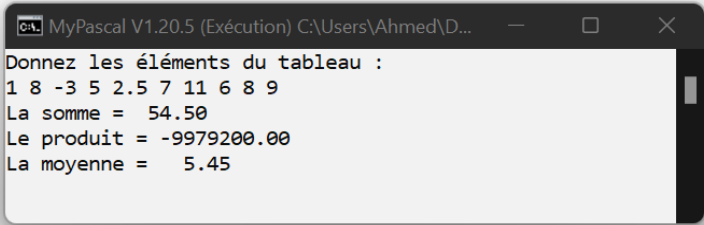


Algorithme	Programme PASCAL
<p>Algorithme Som_Moy_Prod_Tab ;</p> <p>Variabes T : Tableau [1..10] de réel ; i : entier ; Som, Prod, Moy : réel ;</p> <p>Début Ecrire('Donnez les éléments du tableau : '); Pour i ← 1 à 10 faire Lire(T[i]); FinPour ; Som ← 0 ; Prod ← 1 ; Pour i ← 1 à 10 faire Som ← Som+T[i] ; Prod ← Prod*T[i] ; FinPour ; Moy ← Som/10 ;</p> <p>Ecrire('La somme = ', Som:6:2) ; Ecrire('Le produit = ', Prod:6:2) ; Ecrire('La moyenne = ', Moy:6:2) ;</p> <p>Fin.</p>	<p>Program Som_Moy_Prod_Tab ;</p> <p>Var T : array [1..10] of real ; i : integer ; Som, Prod, Moy : real ;</p> <p>Begin writeln('Donnez les éléments du tableau : '); For i := 1 to 10 do read(T[i]) ;</p> <p>Som := 0 ; Prod := 1 ; For i := 1 to 10 do Begin Som := Som+T[i] ; Prod := Prod*T[i] ; End ; Moy := Som/10 ;</p> <p>writeln('La somme = ', Som:6:2) ; writeln('Le produit = ', Prod:6:2) ; writeln('La moyenne = ', Moy:6:2) ;</p> <p>End.</p>

```

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:6:2) ;
21   writeln('Le produit = ', Prod:6:2) ;
22   writeln('La moyenne = ', Moy:6:2) ;
23 End.

```



Explication

Pour calculer la somme des nombres contenus dans le tableau, il faut ajouter un à un le contenu des cases depuis la première jusqu'à la dernière, en utilisant une variable initialisée à zéro.

Pour calculer le produit des nombres contenus dans le tableau, il faut multiplier un par un le contenu des cases depuis la première jusqu'à la dernière, en utilisant une variable initialisée à un.

Pour calculer la moyenne, il suffit de diviser la somme par le nombre de cases du tableau.

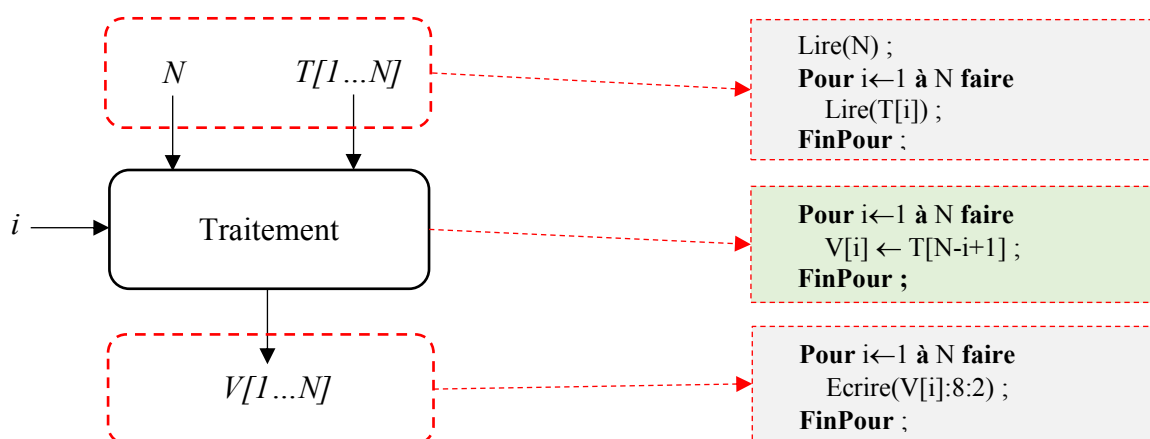
Exercice 03 : Inverser les éléments d'un vecteur

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).

Solution :

Question 01 :

Les variables d'entrée, variable de sortie 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 $i=1 \rightarrow V[1]=T[6]$

Pour $i=2 \rightarrow V[2]=T[5]$

Pour $i=3 \rightarrow V[3]=T[4]$

Pour $i=4 \rightarrow V[4]=T[3]$

Pour $i=5 \rightarrow V[5]=T[2]$

Pour $i=6 \rightarrow V[6]=T[1]$

$V[1]=T[6-1+1]$

$V[2]=T[6-2+1]$

$V[3]=T[6-3+1]$

$V[4]=T[6-4+1]$

$V[5]=T[6-5+1]$

$V[6]=T[6-6+1]$

Pour toutes égalités, on peut écrire :

Pour $i \leftarrow 1$ à N faire
 $V[i] \leftarrow T[N-i+1];$
FinPour ;

Algorithme/Programme PASCAL

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 $i \leftarrow 1$ à N faire Lire(T[i]); FinPour {*-***- Traitement -***-} Pour $i \leftarrow 1$ à N faire $V[i] \leftarrow T[N-i+1];$ FinPour {*-***- Sorties -***-} Ecrire('L'inverse de T est V :'); Pour $i \leftarrow 1$ à N faire Ecrire(V[i]); FinPour Fin.	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 :'); For $i := 1$ to N do Write(V[i]:8:2); {Afficher sur 8 espaces avec 2 chiffres après la virgule} End.


```

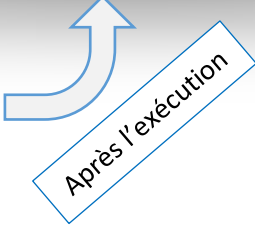
1 Program inverser_T_dans_V;
2 Var
3   i,N : integer;
4   V,T : Array [1..100] of real;
5 Begin
6   {*-**- Entrées -*-}
7   Writeln('Donner la taille du vecteur T :');
8   Read(N);
9   Writeln('Donner les composantes de T :');
10  For i := 1 to N do
11    Read(T[i]);
12  .....
13  {*-**- Traitement -*-}
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



Question 02 : Inverser le vecteur T dans lui même

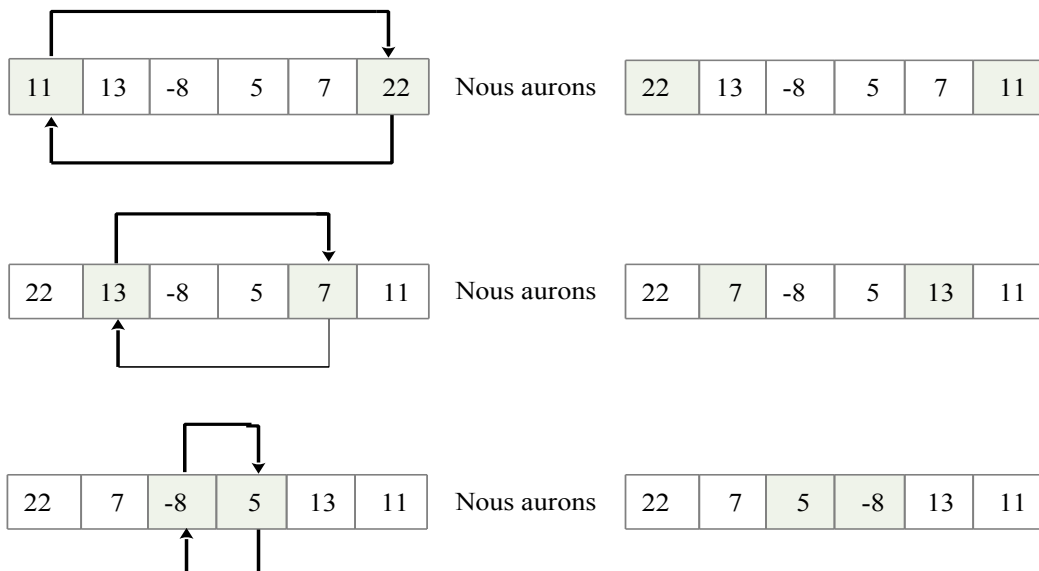
Le même problème que la question 01, sauf que, dans cette deuxième question, nous voulons inverser le vecteur T dans lui-même.

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 :

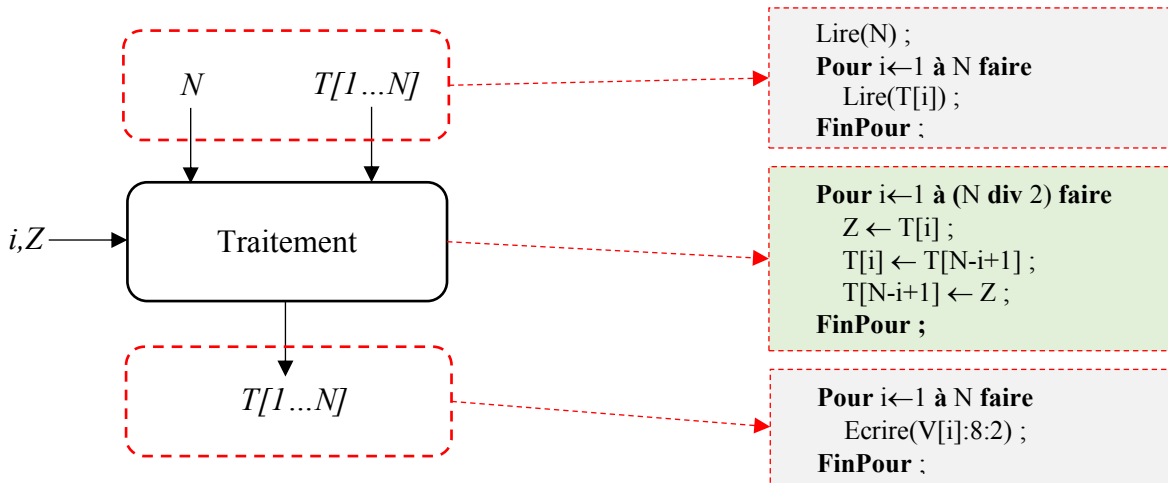
- Inverser un vecteur dans lui-même en permutant des cases.
- Le nombre de permutations est la moitié du vecteur.
- Les permutations : (1 avec N), (2 avec N-1), ... Jusqu'à (N div 2).
- De la question 1, nous déduisons que : la case $N^o \ i$ sera permutée avec la case $N^o \ (N-i+1)$, tel-que $i = 1 \dots (N \text{ div } 2)$.

- 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  $(N \text{ div } 2)$ 
    
```

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



Algorithme/Programme PASCAL

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

```

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.

```

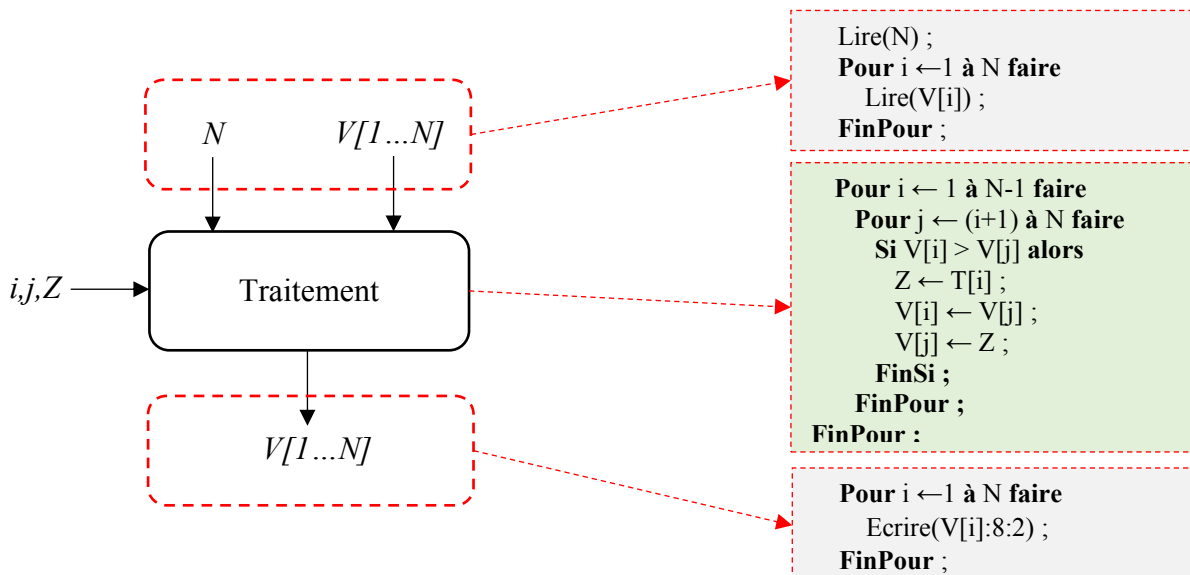
Exercice N°04 : Trier un vecteur avec un ordre croissant

Soit V un vecteur de type réel de taille N.

Ecrire un algorithme/programme PASCAL qui permet de trier (ordonner) les éléments du vecteur V avec un ordre croissant.

Solution :

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

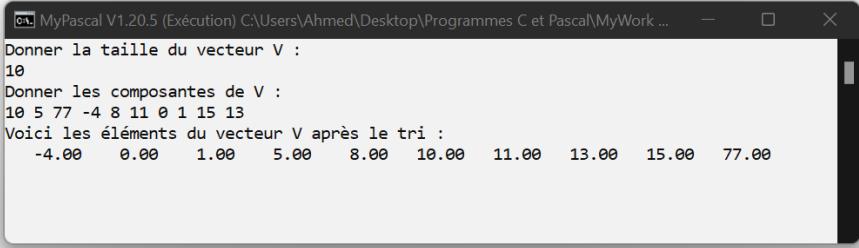


Algorithme/Programme PASCAL

Algorithme	Programme PASCAL
<p>Algorithme Tri_Simple</p> <p>Variables V : Tableau [1..100] de réel ; i, j, N : entier ; Z : réel ;</p> <p>Début</p> <p>{-*-*- Entrées -*-*-}</p> <p>Ecrire('Donner la taille du vecteur V :'); Lire (N) ; Ecrire('Donner les composantes de V :');</p> <p>Pour i ← 1 à N faire Lire (V[i]) FinPour ;</p> <p>{-*-*- Traitement -*-*-}</p> <p>Pour i ← 1 à N-1 faire Pour j ← (i+1) à N faire Si V[i] > V[j] alors Z ← T[j] ; V[i] ← V[j] ; V[j] ← Z ; FinSi ; FinPour ; FinPour ;</p> <p>{-*-*- Sorties -*-*-}</p> <p>Ecrire('Voici les éléments du vecteur V après le tri :');</p> <p>Pour i ← 1 à N faire Ecrire (V[i]:8:2) ; FinPour ;</p> <p>Fin.</p>	<pre> Program Tri_Simple; Var V : array[1..100] of real; i, j, N : integer; Z : real; 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]); {-*-*- Traitement -*-*-} For i:=1 to (N-1) do For j:=(i+1) to N do if V[i] > V[j] then Begin Z:=V[i]; V[i]:=V[j]; V[j]:=Z; End; {-*-*- Sorties -*-*-} Writeln('Voici les éléments du vecteur V après le tri :'); For i:=1 to N do Write(V[i]:8:2); End. </pre>

```


1  Program Tri_Simple;
2  Var
3    V : array[1..100] of real;
4    i, j, N : integer;
5    Z:real;
6  Begin
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   For i:=1 to (N-1) do
13     For j:=(i+1) to N do
14       if V[i] > V[j] then
15         Begin
16           Z:=V[i];
17           V[j]:=V[i];
18           V[i]:=Z;
19         End;
20   Writeln('Voici les éléments du vecteur V après le tri :');
21   For i:=1 to N do
22     Write(V[i]:8:2);
23   End.
        
```



MyPascal V1.20.5 (Exécution) C:\Users\Ahmed\Desktop\Programmes C et Pascal\MyWork ...

```

Donner la taille du vecteur V :
10
Donner les composantes de V :
10 5 77 -4 8 11 0 1 15 13
Voici les éléments du vecteur V après le tri :
-4.00  0.00  1.00  5.00  8.00  10.00  11.00  13.00  15.00  77.00
        
```



Après l'exécution

Explication

Il existe plusieurs stratégies possibles pour trier les éléments d'un tableau ; nous en verrons la méthode de tri par sélection (tri simple).

Dans ce cas, le tri d'un tableau consiste à mettre en bonne position l'élément numéro 1, c'est-à-dire le plus petit, puis, on met en bonne position l'élément suivant, et ainsi de suite, jusqu'au dernier.

Par exemple, si l'on part de : 10, 5, 77, -4, 8, 11, 0, 1, 15, 13.

On prend l'élément occupant la première case, et on le compare avec le reste du tableau (à partir de la deuxième case, jusqu'à la dixième).

Chaque fois où cet élément est supérieur, on le permute avec l'élément comparé. Le tableau devient ainsi : -4, 10, 77, 5, 8, 11, 0, 1, 15, 13.

On prend maintenant l'élément occupant la deuxième case, et on le compare avec le reste du tableau (à partir de la troisième case, jusqu'à la dixième). Chaque fois où cet élément est supérieur, on le permute avec l'élément comparé. Le tableau devient ainsi : -4, 0, 77, 10, 8, 11, 5, 1, 15, 13.

Remarquons qu'à ce niveau les deux plus petits éléments (-4 et 0) ont pris leurs positions dans le tableau.

On prend maintenant l'élément occupant la troisième case, et on le compare avec le reste du tableau (à partir de la quatrième case, jusqu'à la dixième). Chaque fois où cet élément est supérieur, on le permute avec l'élément comparé. Le tableau devient ainsi : -4, 0, 1, 77, 10, 11, 8, 5, 15, 13.

Cette opération se répète jusqu'à arriver au neuvième élément, le comparer avec le dixième, s'il est supérieur, alors permuter ces deux derniers éléments.

Le processus de tri d'un tableau par sélection consiste en deux boucles imbriquées :

- Une boucle principale : allant du premier élément du tableau, puis le second, etc. jusqu'à l'avant dernier.
- Une boucle secondaire : allant du compteur de la boucle principale plus un, jusqu'au dernier élément du tableau. On compare l'élément du tableau (ayant comme indice la valeur courante du compteur de la boucle principale) avec l'élément du tableau (ayant comme indice la valeur courante du compteur de la boucle secondaire). Si le premier est supérieur au deuxième, on permute les deux éléments.