TP Informatique 2

Corrigé de la série de TP N°1 – Tableaux à une dimension - Vecteurs

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 ?

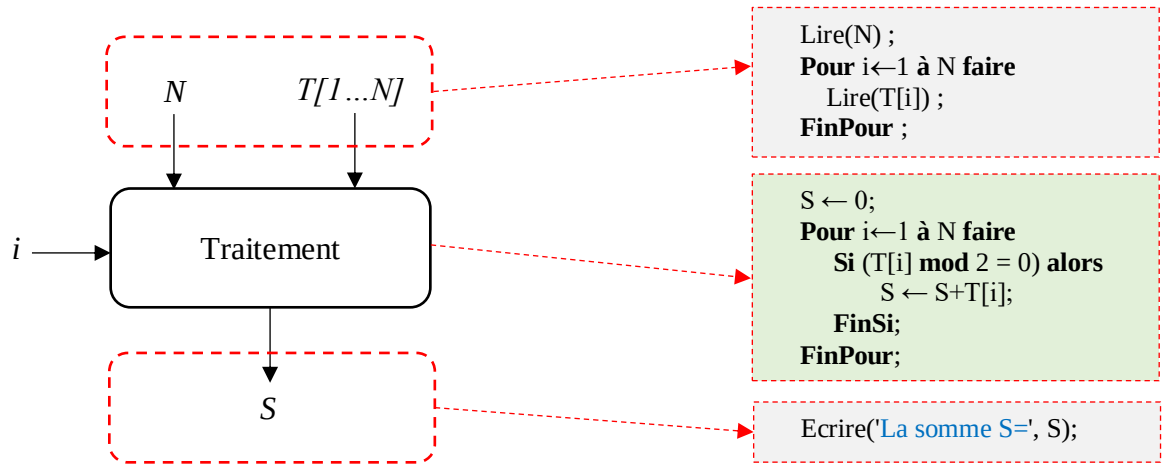
4- Déduire ce que fait l'algorithme ?

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 sorite et la partie traitement sont présentées dans le schéma ci-dessous :



Remarque :

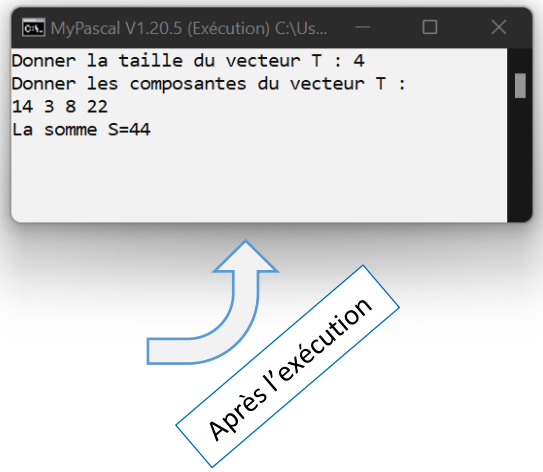
La variable ***i*** est une variable de traitement ou intermédiaire, utilisée pour parcourir le vecteur T.

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

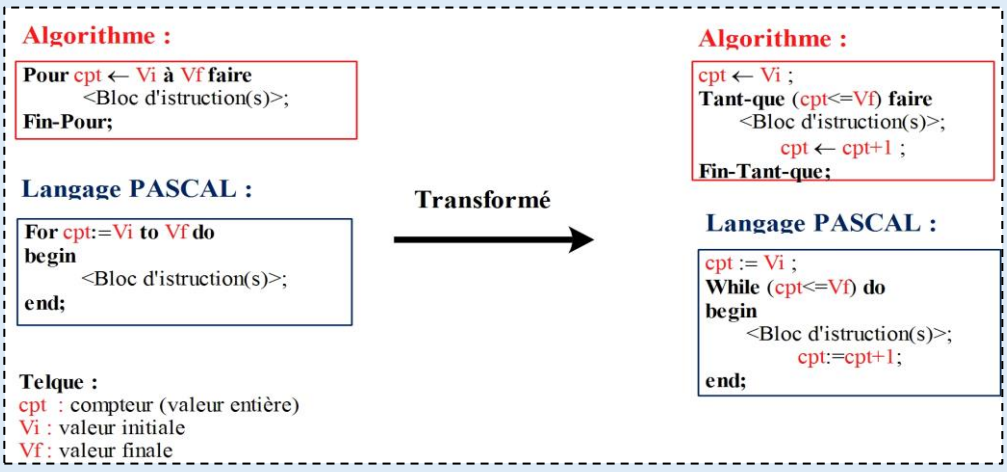
4- Déduire ce que fait l'algorithme

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

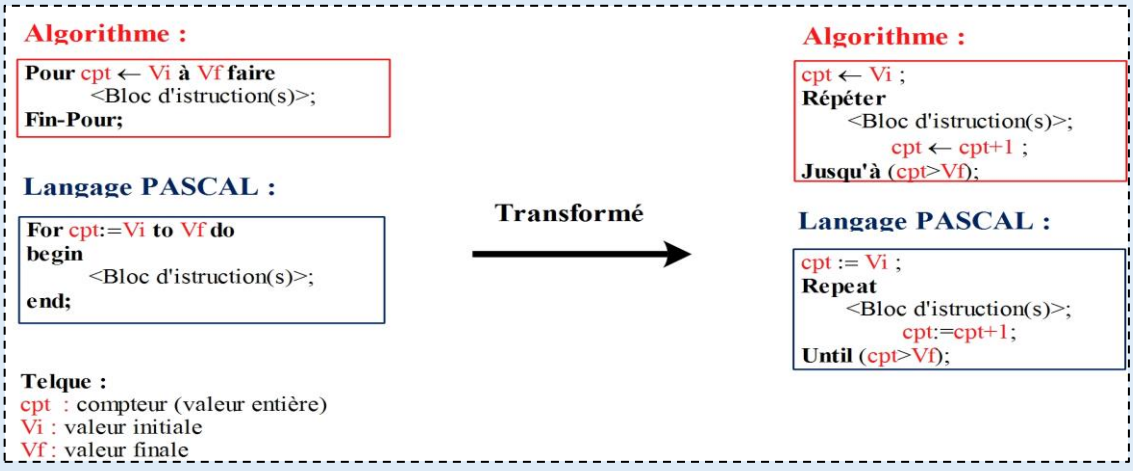
Algorithme Vecteur ;
Variables
T : Tableau [1..100] **d'entier** ;
N, i, S : **entier**;
Début
{-*-*-*- Entrées -*-*-*-}
Ecrire('Donner la taille du vecteur T : ');
Lire(N);
Ecrire('Donner les composantes du vecteur T : ');
Pour i←1 **à** N **faire**
Lire(T[i]);
Fin-Pour;
{-*-*-*- Traitements -*-*-*-}
S ← 0;
i ← 1;
Tant-que (i<=N) **faire**
Si (T[i] **mod** 2 = 0) **alors**
S ← S+T[i];
FinSi;
i ← i+1;
Fin-Tant-que;
{-*-*-*- 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;
	i:=1;
	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;
	{-*-*-*- Sorties -*-*-*-}
	Write('La somme S=',S); {Affichage de la somme des divisibles de 2}
	End.

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

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;
i ← 1;
Répéter
Si (T[i] **mod** 2 = 0) **alors**
S ← S+T[i];
FinSi;
i ← i+1;
Jusqu'à (i>N);
{-*-*-*- 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;
	i:=1;
	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;
	Until (i>N) ;
	{-*-*-*- Sorties -*-*-*-}
	Write('La somme S=',S); {Affichage de la somme des divisibles de 2}
	End.

Remarques :

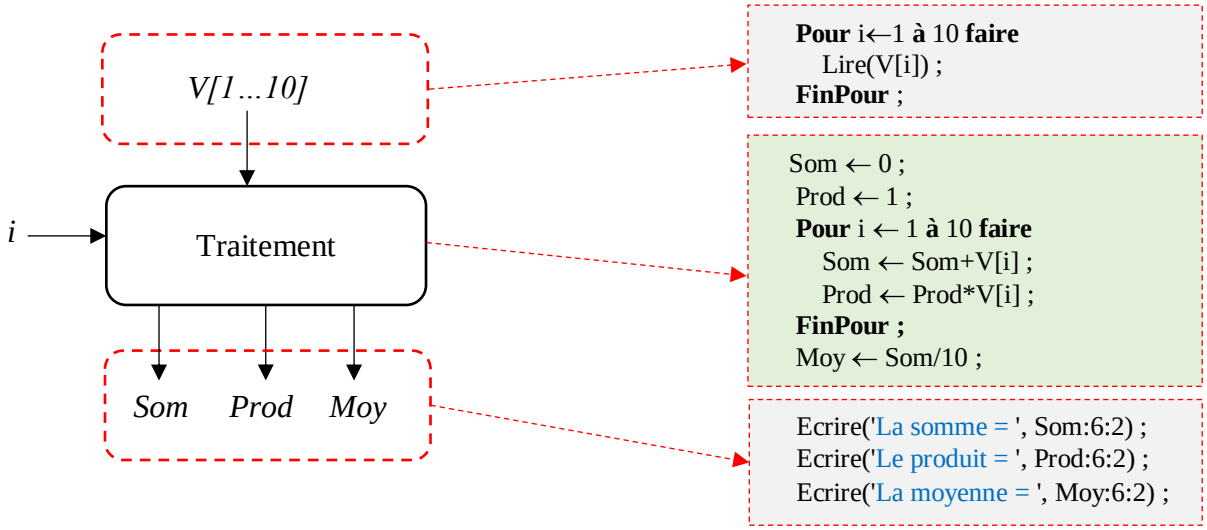
- Un vecteur (Tableau à une dimension) est une suite de cases mémoires adjacentes. Ces cases mémoires définissent des variables de même types.
- La déclaration suivante : T : **Tableau** [1..100] d'entier;
  Signifie que nous réservons 100 cases de type entier. 100 est la taille maximale du vecteur T. 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 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 **T[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 sorite et la partie traitement sont présentées dans le schéma ci-dessous :



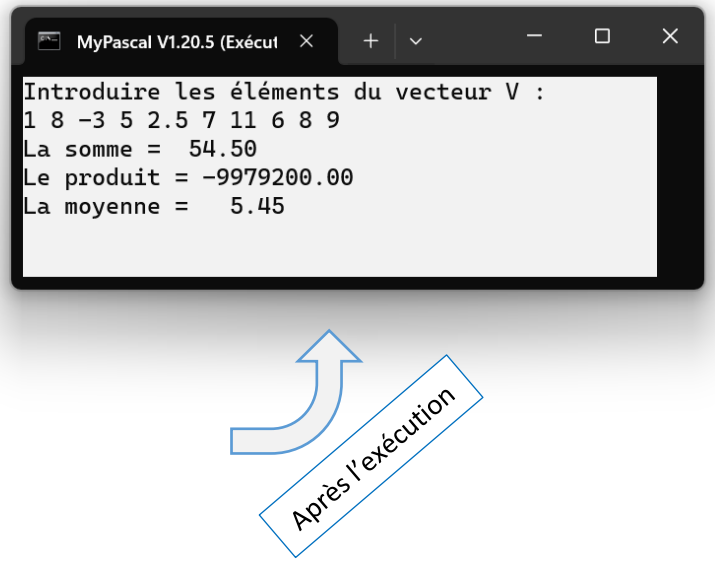
Algorithme	Programme PASCAL

Algorithme Som_Moy_Prod_Tab ;
Variables
V : Tableau [1..10] **de réel** ;
i : **entier** ;
Som, Prod, Moy : **réel** ;
Début
Ecrire('Introduire les éléments du vecteur V : ') ;
Pour i ← 1 **à** 10 **faire**
Lire(V[i]) ;
FinPour ;
Som ← 0 ;
Prod ← 1 ;
Pour i ← 1 **à** 10 **faire**
Som ← Som+V[i] ;
Prod ← Prod*V[i] ;
FinPour ;
Moy ← Som/10 ;
 	
Ecrire('La somme = ', Som:6:2) ;
Ecrire('Le produit = ', Prod:6:2) ;
Ecrire('La moyenne = ', Moy:6:2) ;
Fin.
	Program Som_Moy_Prod_Tab ;
	Var
	V : **array** [1..10] **of real** ;
	i : **integer** ;
	Som, Prod, Moy : **real** ;
	Begin
	writeln('Introduire les éléments du vecteur V : ') ;
	For i := 1 **to** 10 **do**
	read(V[i]) ;
	
	Som := 0 ;
	Prod := 1 ;
	For i := 1 **to** 10 **do**
	Begin
	Som := Som+V[i] ;
	Prod := Prod*V[i] ;
	End ;
	Moy := Som/10 ;
	
	writeln('La somme = ', Som:6:2) ;
	writeln('Le produit = ', Prod:6:2) ;
	writeln('La moyenne = ', Moy:6:2) ;
	End.

```pascal
Program Som_Moy_Prod_Tab ;
Var
  V : array [1..10] of real ;
  i : integer ;
  Som, Prod, Moy : real ;
Begin
  writeln('Introduire les éléments du vecteur V : ') ;
  For i := 1 to 10 do
    read(V[i]) ;

  Som := 0 ;
  Prod := 1 ;
  For i := 1 to 10 do
    Begin
      Som := Som+V[i] ;
      Prod := Prod*V[i] ;
    End ;
  Moy := Som/10 ;

  writeln('La somme = ', Som:6:2) ;
  writeln('Le produit = ', Prod:6:2) ;
  writeln('La moyenne = ', Moy:6:2) ;
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.

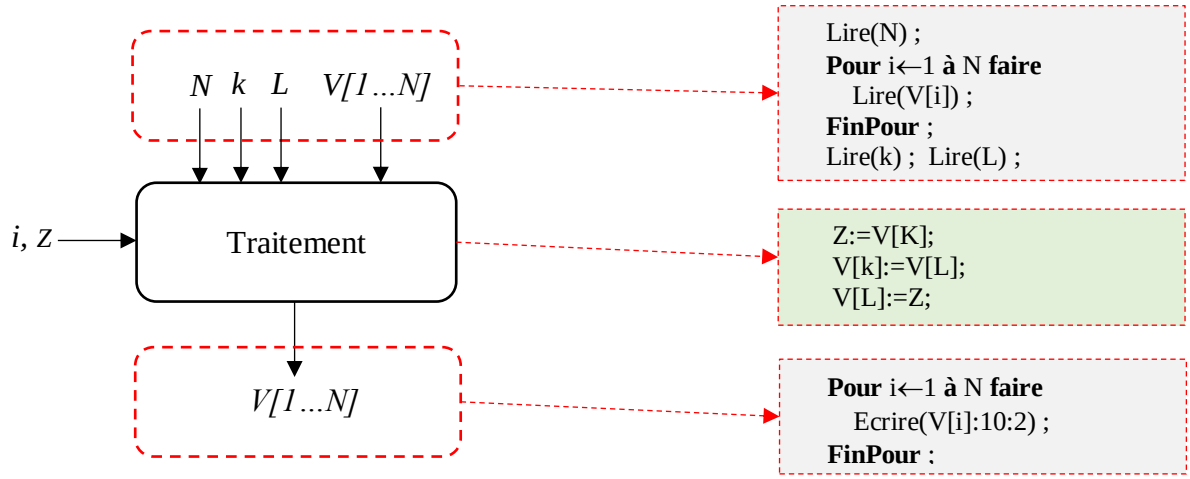
Exercice N°03 : Permutation entre les cases d'indice K et L

Soit V un vecteur de type réel et de taille N, et soient K et L deux positions dans le vecteur V.

Écrire un algorithme/Programme PASCAL qui permet de permuter entre les deux éléments du vecteur V, d'indice K et L.

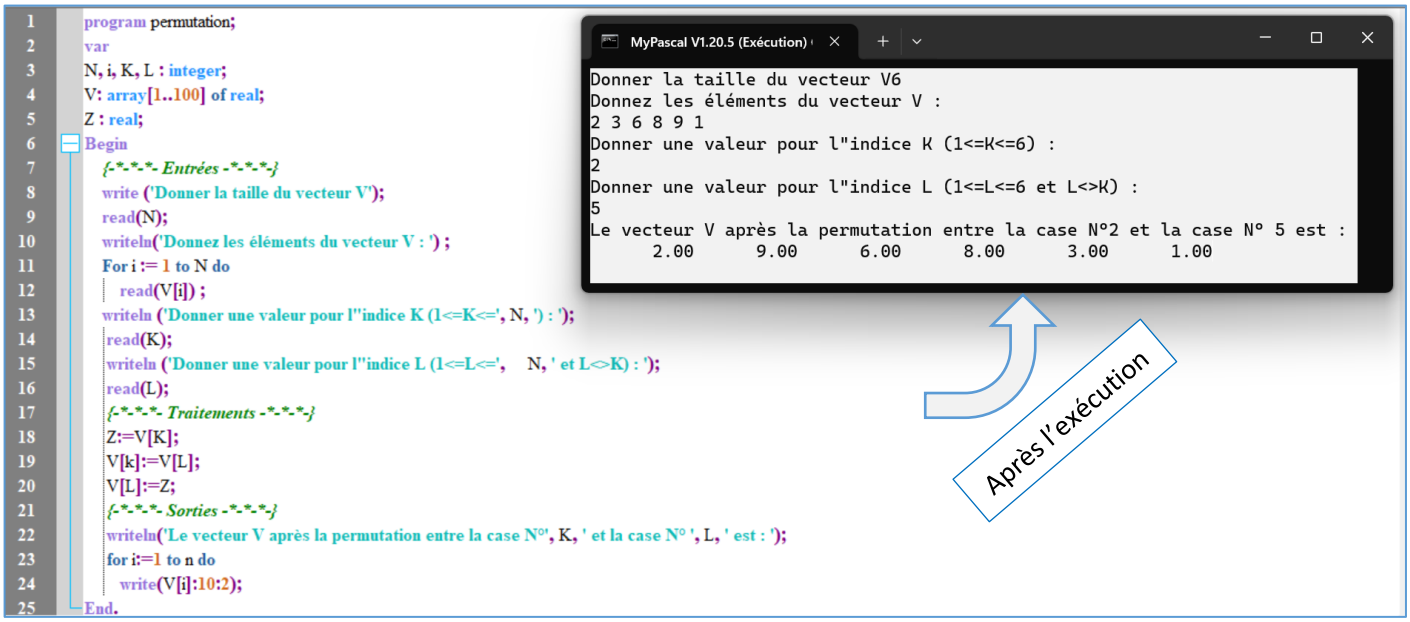
Solution :

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



Algorithme	Programme PASCAL

Algorithme permutation;
variables
N, i, K, L : **entiers**;
V: Tableau [1..100] de **réels**;
Z : **réel**;
 	
Début
{-*-*-*- Entrées -*-*-*-}
Ecrire('Donner la taille du vecteur V');
Lire(N);
Ecrire('Donnez les éléments du vecteur V : ') ;
Pour i ← 1 **à** N **faire**
Lire(V[i]) ;
FinPour ;
Ecrire('Donner une valeur pour l''indice K (1<=K<=', N, ') : ');
Lire(K);
 	
Ecrire('Donner une valeur pour l''indice L (1<=L<=', N, ' et L<>K) : ');
Lire(L);
 	
{-*-*-*- Traitements -*-*-*-}
Z ← V[K];
V[k] ← V[L];
V[L] ← Z;
 	
{-*-*-*- Sorties -*-*-*-}
Ecrire('Le vecteur V après la permutation entre la case N°', K, ' et la case N° ', L, ' est : ');
Pour i ← 1 **à** N **faire**
Ecrire(V[i]:10:2);
FinPour ;
Fin.
	program permutation;
	var
	N, i, K, L : **integer**;
	V: **array**[1..100] of **real**;
	Z : **real**;
	
	Begin
	{-*-*-*- Entrées -*-*-*-}
	write ('Donner la taille du vecteur V');
	read(N);
	writeln('Donnez les éléments du vecteur V : ') ;
	For i := 1 **to** N **do**
	read(V[i]) ;
	
	writeln ('Donner une valeur pour l''indice K (1<=K<=', N, ') : ');
	read(K);
	
	writeln ('Donner une valeur pour l''indice L (1<=L<=', N, ' et L<>K) : ');
	read(L);
	
	{-*-*-*- Traitements -*-*-*-}
	Z:=V[K];
	V[k]:=V[L];
	V[L]:=Z;
	
	{-*-*-*- Sorties -*-*-*-}
	writeln('Le vecteur V après la permutation entre la case N°', K, ' et la case N° ', L, ' est : ');
	for i:=1 **to** N **do**
	write(V[i]:10:2);
	
	End.



Explication☺

pour permuter les éléments d'indices K et L dans un vecteur V de taille N en Pascal, vous pouvez suivre ce principe :

Sauvegardez la valeur de V[K] dans une variable temporaire Z pour ne pas la perdre.

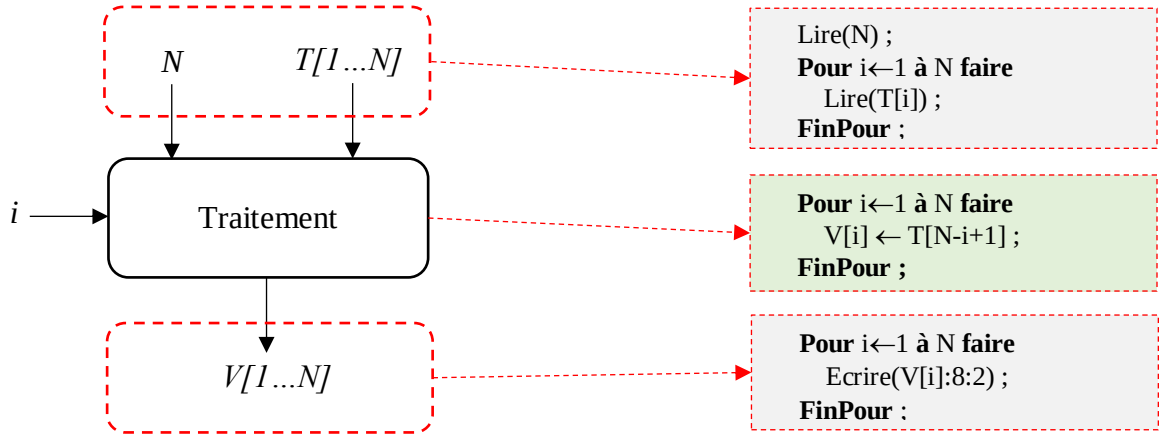
Remplacez V[K] par V[L] et ensuite, utilisez la variable temporaire Z pour attribuer sa valeur à V[L].

Exercice 04 : Inverser les éléments d'un vecteur

Ecrire un algorithme/programme PASCAL qui permet d'inverser les éléments d'un vecteur de type réel T dans un autre vecteur V.

Solution :

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	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
 	
{-*-*-*- 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.
	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
```

Après l'exécution