Sommaire

Série 1 : Type Tableau (Vecteurs et Matrices)

Exercice 1 : Lecture et Affichage d'un vecteur

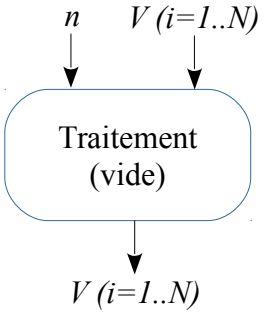
Écrire un algorithme/programme PASCAL qui permet de lire et afficher un vecteur V de N composantes réelles.

Solution

L'algorithme

```
Algorithme exercie_1
     Variables
          V : Tableau [1..100] de Réel
          i,n : entier
Début
     Lire(n)
     Pour i←1 à n faire
          Lire( V[i] )
     Fin-Pour

     Pour i←1 à n faire
          Écrire( V[i] )
     Fin-Pour
Fin
```



Le programme PASCAL

```
01 Program exercie_1;
02 Uses wincrt ;
03 var
04     V : array[1..100] of Real;
05     i, n : integer ;
06 Begin
07     {Les entrées}
08     Write('Donner la taille du vecteur : ');
09     Read(n);
10     Writeln('Donner les composantes du vecteur : ');
11     For i:=1 to n do
12          Read( V[i] );
13
14     {Les sorties}
15     Writeln('Affichage du Vecteur : ');
16     For i:=1 to n do
17          Write( V[i]:10:3 );{Afficher sur 10 caractères et}
18                             {2 chiffres après la virgule}
19 End.
```

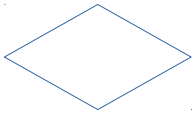
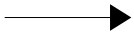
Explication

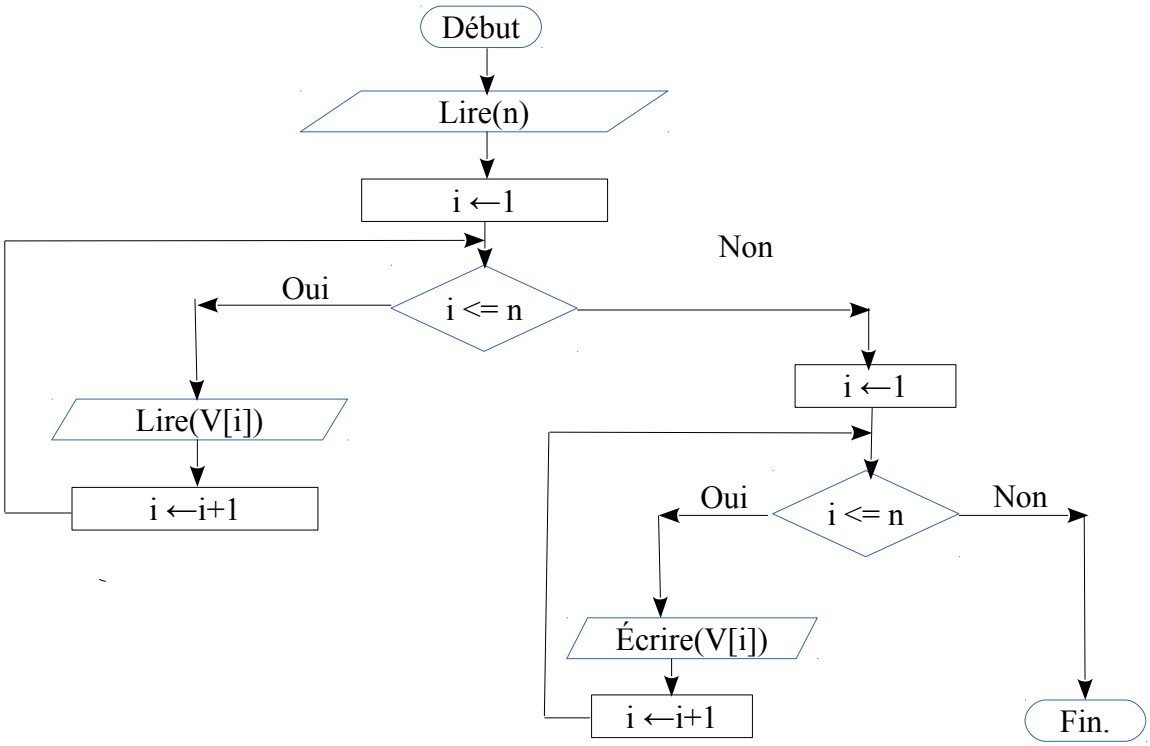
Ce programme montre comment réaliser la lecture et l'écriture d'un vecteur (une séquence de cases mémoire contigüe de même type). Pour les deux opérations (lecture et écriture) on aura besoin toujours d'une boucle **Pour**. Lors de la déclaration, on réserve 100 cases réelles (le taille maximale du vecteur), et on utilise la variable **n** pour déterminer la taille qu'on veut utiliser (par exemple 5 cases). L'accès à une composante d'indice **i** se fait comment suit : **V[i]**. Ainsi, pour réaliser la lecture de la case 2, on écrira **Lire(V[2])**, et **Écrire(T[4])** pour afficher la valeur de la 4ème case.

Il faut noter que ce programme ne réalise pas de traitement, il continent uniquement des entrées et des sorties.

Organigramme

L'organigramme est une façon de montrer un algorithme (ou un programme) sous forme d'actions ***schématisées*** et leurs enchaînement (***flèches***)

Les différentes figures d'organigramme	
Figure	***Sémantique / Sense***
	Représente le début et la Fin de l'organigramme
	Entrées / Sorties : Lecture des données et écriture des résultats.
	Calculs, Traitements
	Tests et décision : on écrit le test à l'intérieur du losange
	Ordre d'exécution des opérations (Enchaînement)
	Connecteur



Exercice 2 : Somme et Moyenne des éléments d'un vecteur

Écrire un algorithme/un programme PASCAL qui permet Calculer la somme et la moyenne des éléments d'un vecteur V réel de taille N.

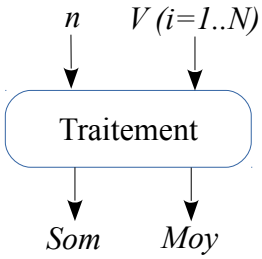
Solution

L'algorithme

```
Algorithme exercie_2
     Variables
          V : Tableau [1..100] de Réel
          i,n : entier
          Som, Moy : Réel
Début
     Lire(n)
     Pour i←1 à n faire
          Lire( V[i] )
     Fin-Pour

     Som ← 0
     Pour i←1 à n faire
          Som ← Som + V[i]
     Fin-Pour
     Moy ← Som / n

     Ecrire(Som, Moy)
Fin
```



Le programme PASCAL

```
01 Program exercie_2;
02 Uses wincrt ;
03 var
04     V : array[1..100] of Real;
05     i, n : integer ;   Som, Moy : real;
06 Begin
07     {Les entrées}
08     Write('Donner la taille du vecteur : ');
09     Read(n);
10     Writeln('Donner les composantes du vecteur : ');
11     For i:=1 to n do
12          Read( V[i] );
13
14     {Les traitements}
15     Som:=0; {Som = V[1] + V[2] + V[3] + … + V[n]}
16     For i:=1 to n do
17          Som:= Som + V[i];
18
19     Moy:=Som/n; {La moyenne égale à la somme sur le nombre d'éléments}
20
21     {Les sorties}
22     Writeln('La somme est : ', Som:8:3);
23     Writeln('La moyenne est : ', Moy:8:3);
24 End.
```

Explication

Soit V un vecteur de n éléments réels, le calcul de la somme de ces éléments se fait par la formule mathématique : V[1]+V[2]+...+V[n]. Si on met l'identificateur Som pour la variable qui représente la somme de ces cases, donc on aura :

$$\text{Som} = V[1]+V[2]+...+V[n] = \sum_{i=1}^{n} V[i]$$

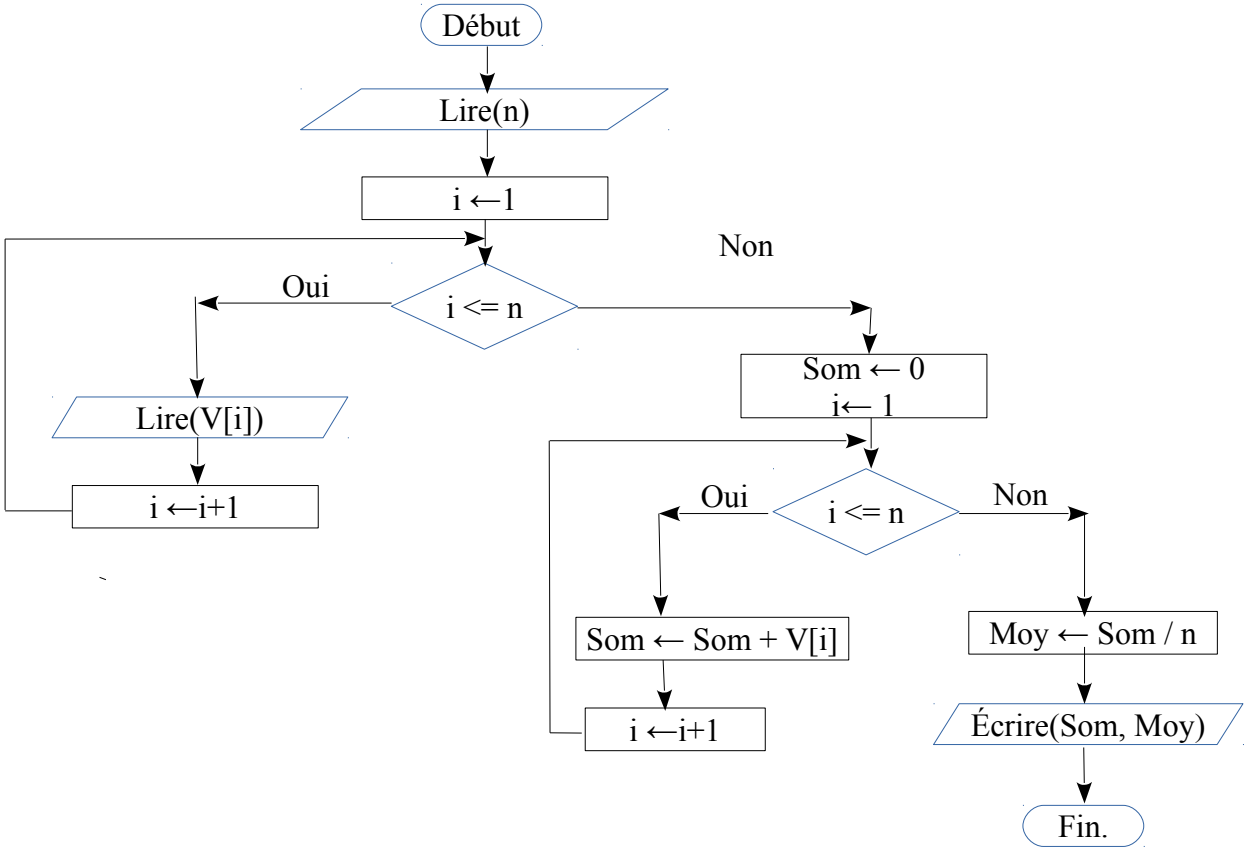
On a vu, auparavant, que le symbole de la somme sera remplacer par un boucle **Pour**, avec le terme répétitif, pour cette somme, égale à V[i], ce qui donne :

Pour i← 1 **à** n **faire**

Som ← Som + V[i]

Remarquer que les bornes de la sommes (i=1 à n) sont les mêmes bornes de la boucle Pour (i←1 à n). Et pour la moyenne, on sait que : Moyenne = Somme / le nombre d'éléments sommés.

Organigramme



Exercice 3 : L'inverse des éléments d'un vecteur

1 Écrire 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

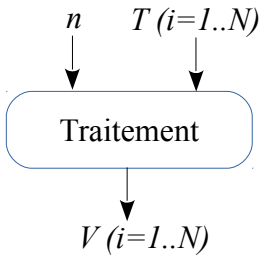
1- Inverser les éléments du vecteur T dans le vecteur V

L'algorithme

```
Algorithme exo3_1
      Variables
            T,V : Tableau [1..100] de Réel
            i,n : entier
Début
      Lire(n)
      Pour i←1 à n faire
            Lire( T[i] )
      Fin-Pour

      Pour i←1 à n faire
            V[i] ← T[n-i+1]
      Fin-Pour


      Pour i←1 à n faire
            Écrire( V[i] )
      Fin-Pour
Fin
```



Le programme PASCAL

```
01 Program exo3_1;
02 Uses wincrt ;
03 var
04      T, V : array[1..100] of Real;
05      i, n : integer ;
06 Begin
07      {Les entrées}
08      Write('Donner la taille du vecteur T : ');
09      Read(n);
10      Writeln('Donner les composantes du vecteur T : ');
11      For i:=1 to n do
12            Read( T[i] );
13
14      {Les traitements}
15      For i:=1 to n do
16            V[i] := T[n-i+1];
17
18      {Les sorties}
19      Writeln('L''affichage du vecteur V : ');
20      For i:=1 to n do
21            Write( V[i] :10:2);
22
23 End.
```

Explication

Mettre les éléments du vecteur T dans un autre vecteur V, mais dans les sens inverse, c'est à dire :

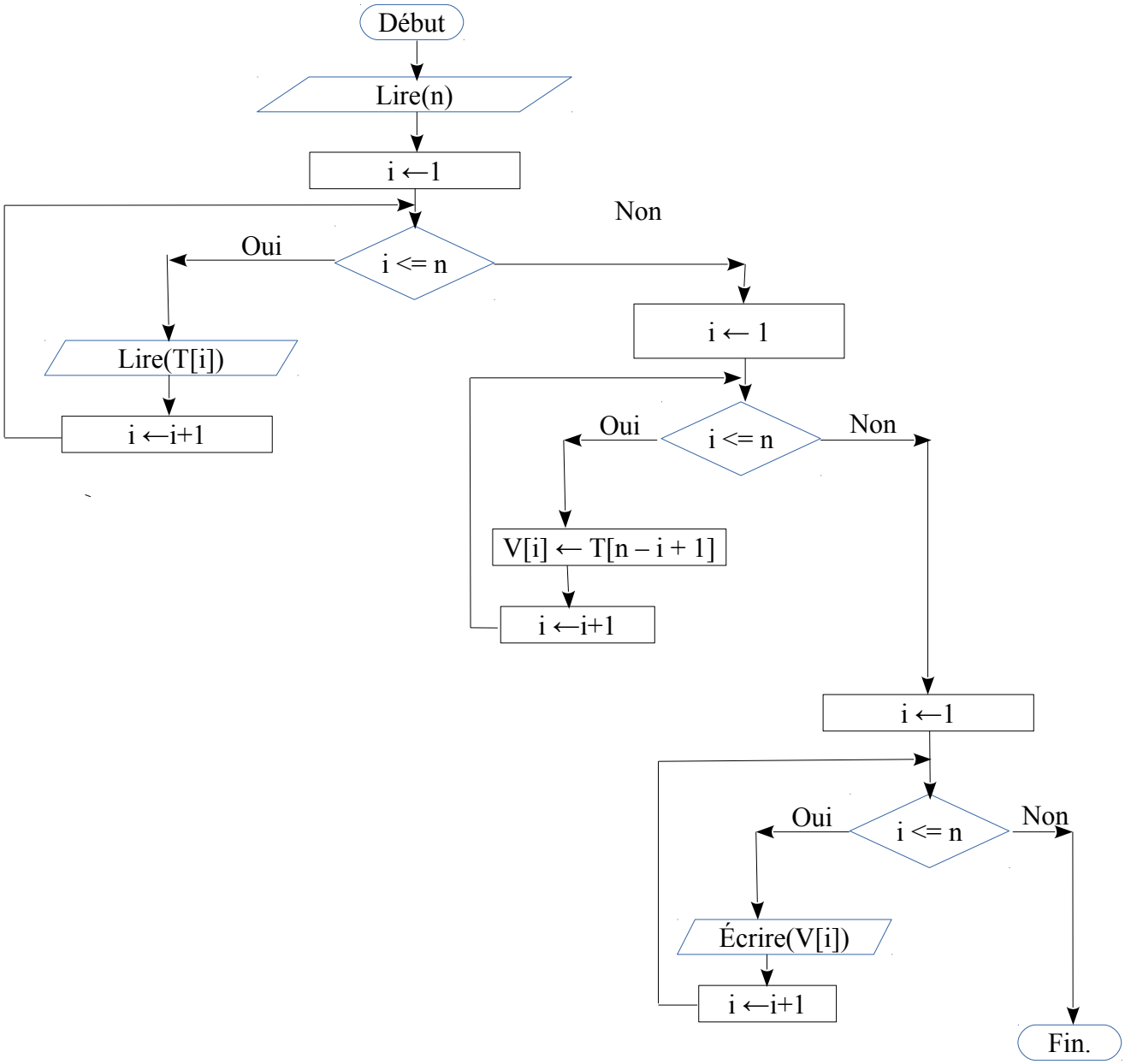
V[1] ← T[n] ; V[2] ← T[n-1] ; V[3] ← T[n-2] ; …….. ; V[n] ← T[1] ;

La correspondance entre les indices de V et ceux de T sont commet suit :

1 ⟷ n – 0 2 ⟷ n – 1 3 ⟷ n – 2 4 ⟷ n – 3 ...

Si on généralise par rapport à l'indice i, on aura : i ⟷ n – (i-1), donc aura l'affectation suivante : Pour toute valeur *i* entre ***1*** et ***n***, *V[i] ← T[n-i+1]*

Organigramme



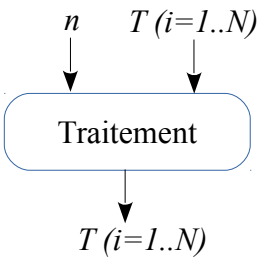
2- Inverser les éléments du vecteur T dans lui même (sans utiliser un autre vecteur)

L'algorithme

```
Algorithme exo3_2
        Variables
                T : Tableau [1..100] de Réel
                i,n : entier
                Z:Réel
Début
        Lire(n)
        Pour i←1 à n faire
                Lire( T[i] )
        Fin-Pour

        Pour i←1 à (n div 2) faire
                Z ← T[i]
                T[i] ← T[n-i+1]
                T[n-i+1] ← Z
        Fin-Pour


        Pour i←1 à n faire
                Écrire( T[i] )
        Fin-Pour
Fin
```



Le programme PASCAL

```
01 Program exo3_2;
02 Uses wincrt ;
03 var
04      T : array[1..100] of Real;
05      i, n : integer ;  Z:real;
06 Begin
07      {Les entrées}
08      Write('Donner la taille du vecteur T : ');
09      Read(n);
10      Writeln('Donner les composantes du vecteur T : ');
11      For i:=1 to n do
12              Read( T[i] );
13
14      {Les traitements}
15      For i:=1 to (n div 2) do {Inverser par permutation de cases}
16       Begin                   {La case i est permutée avec la case n-i+1}
17              Z := T[i];       {Conserver T[i] dans Z}
18              T[i] := T[n-i+1]; {On peut écraser T[i] par T[n-i+1]}
19              T[n-i+1] := Z;    {Maintenant on met Z dans T[n-i+1]}
20       end;
21
22      {Les sorties}
23      Writeln('L''affichage du vecteur T : ');
24      For i:=1 to n do
25              Write( T[i] :10:2);
26
30 End.
```

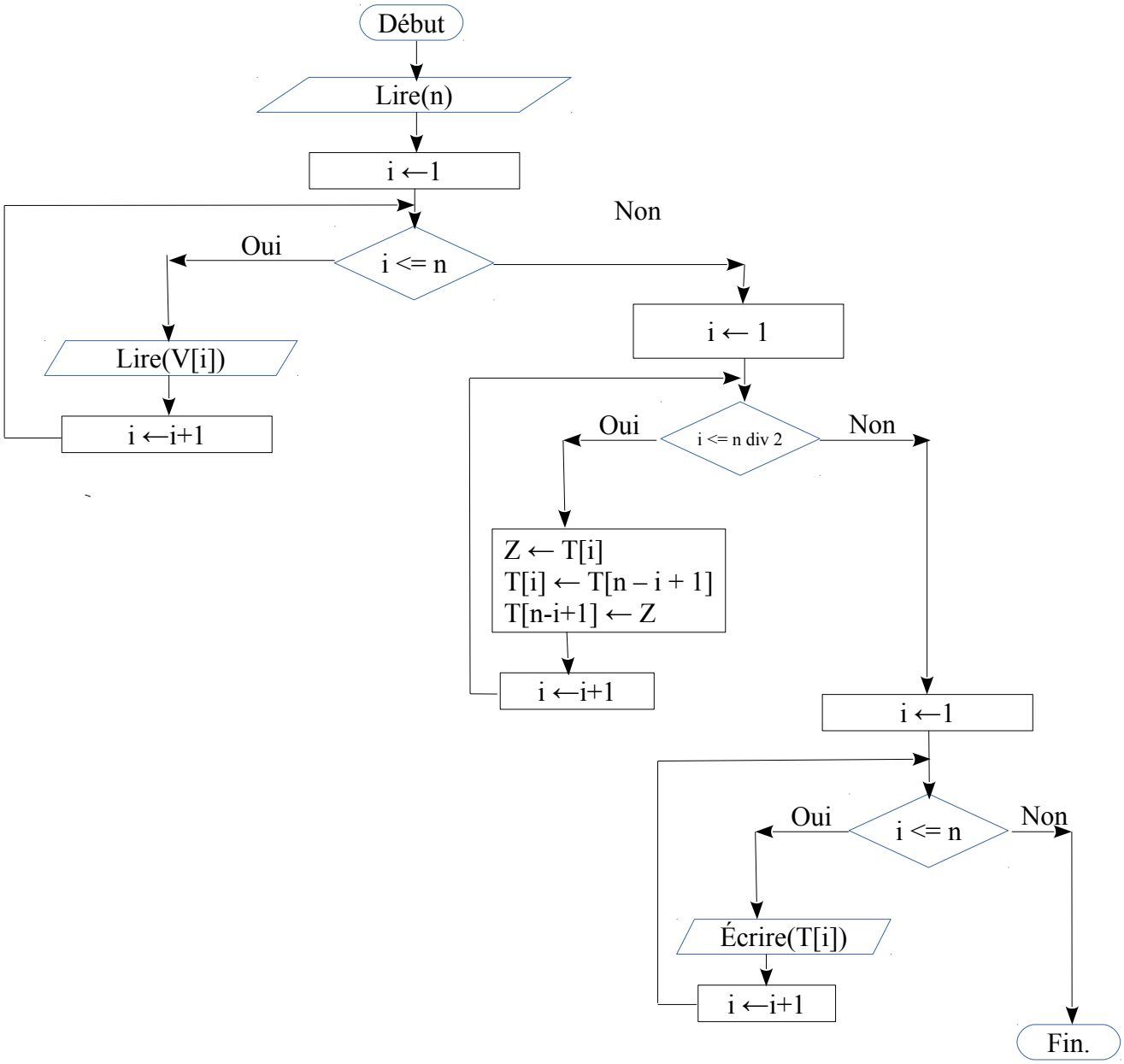
Explication

La solution qui vient immédiatement à l'esprit est une suite d'affectations : T[i] ← T[n-i+1]. Le problème ici est que la valeur de T[i] sera écrasée par T[n-i+1]. Donc ce cas, il faut penser à réaliser une permutation entre les deux cases : *i* et *n-i+1*. Et ce ci avec les trois affectations :

Z ← T[i] ; T[i] ← T[n-i+1] ; T[n-i+1] ← Z ; (comme la permutation entre x et y). Il faut faire attention, il faut réaliser cette permutation entre un case de la première moitié du vecteur avec la

case correspondante de la deuxième moitié du vecteur, donc il faut arrêter les permutations au milieu du vecteur. Ce qui veut dire que le nombre de permutations à réaliser un vecteur de taille *n* est (*n div 2*). (pour n=5, on aura 2 permutations possibles 1 avec 5 et 2 avec 4) (pour n=6, on aura 3 permutations possibles : 1 avec 6, 2 avec 5 et 3 avec 4).

Organigramme



Exercice 4 : Min et le Max d'un vecteur et leurs positions

1 Écrire un algorithme/programme PASCAL qui permet de rechercher le plus petit élément dans un vecteur réel V ainsi que sa position.

2 Écrire un programme PASCAL qui permet de rechercher le plus grand élément dans un vecteur réel V ainsi que sa position.

Solution

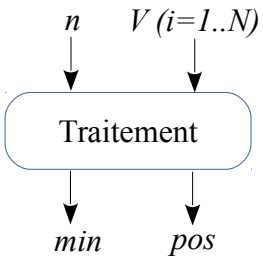
1- Recherche du min et sa position dans vecteur

L'algorithme

```
Algorithme exo4_1_Min
      Variables
            V : Tableau [1..100] de Réel
            i,n, Pos : entier   Min:Réel
Début
      Lire(n)
      Pour i←1 à n faire
            Lire( V[i] )
      Fin-Pour

      Min ← V[1]
      Pos ← 1
      Pour i←1 à n faire
            Si T[i] < Min alors
               Min ←  V[i]
               Pos ← i
            Fin-Si
      Fin-Pour


      Écrire( Min, Pos)
Fin
```



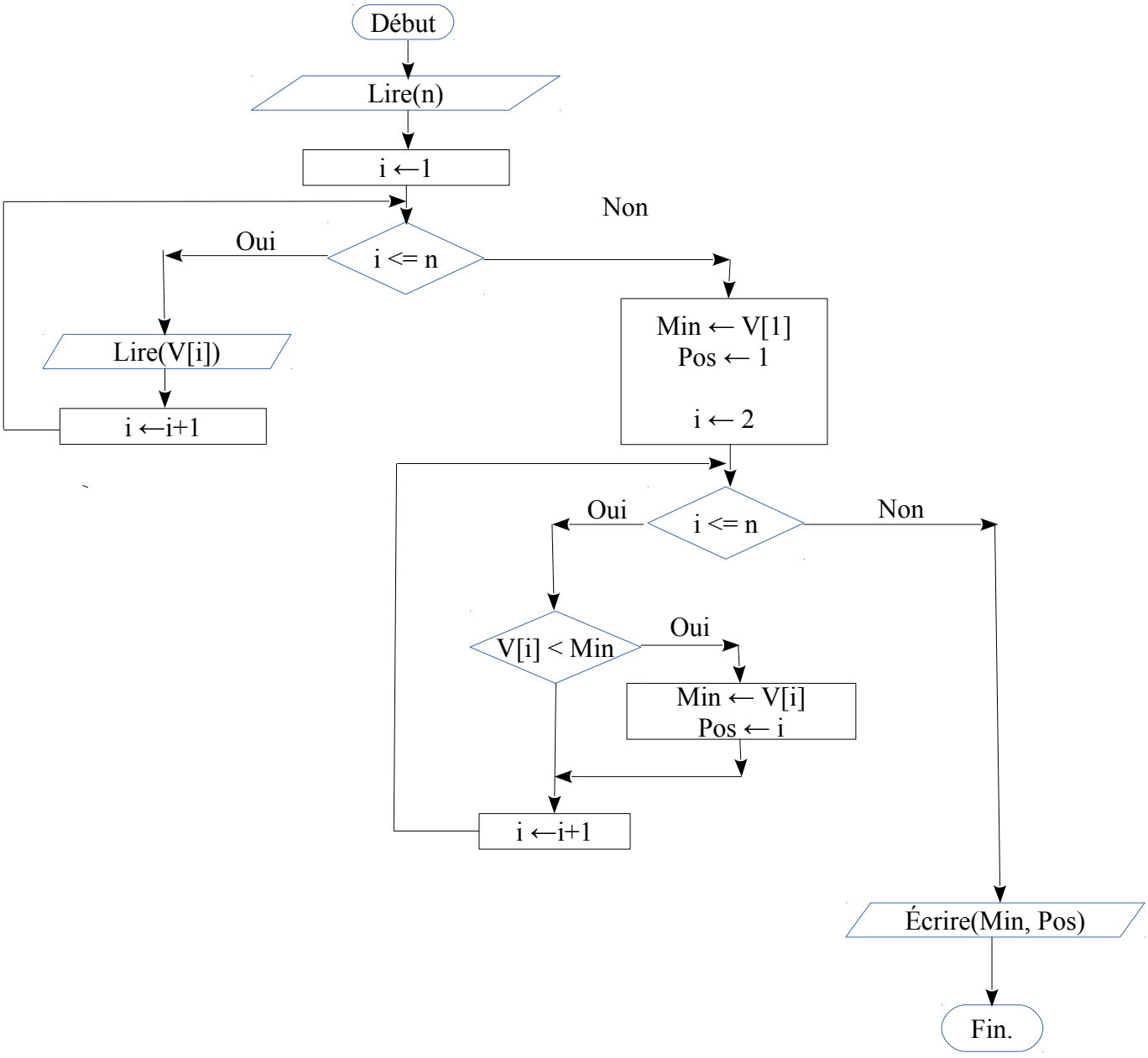
Le programme PASCAL

```
01 Program exo4_1_Min;
02 Uses wincrt ;
03 var
04      T, V : array[1..100] of Real;
05      i, n, Pos : integer ;  Min:Real;
06 Begin
07      {Les entrées}
08      Write('Donner la taille du vecteur V : ');
09      Read(n);
10      Writeln('Donner les composantes du vecteur V : ');
11      For i:=1 to n do
12            Read( V[i] );
13
14      {Les traitements}
15      Min:=V[1]; Pos:=1;  {On suppose que le Min est la valeur de la première case}
16      For i:=2 to n do    {On parcours toutes les autres cases}
17            If V[i] < Min then {Pour chaque case qui est inférieur au Min}
18             begin
19               Min := V[i]; {On actualise le Min (corriger le Min) }
20               Pos := i;    {On actualise sa position}
21             end;
22
23      {Les sorties}
24      Write('Le Min du V est : ',Min:2:2,' et sa position est : ', Pos);
25 End.
```

Explication

La solution de la recherche du min dans un vecteur ainsi que sa position commence par la supposition que le premier élément du vecteur est le minimum (la ligne 15 du programme précédent : *Min:=V[1] ; Pos:=1;*). Par la suite, on parcours toutes les autres cases de l'indice *2* jusqu'à l'indice *n* pour vérifier s'il y parmi ces cases celles qui vérifiassent la condition *V[i]<Min*, pour chaque élément *V[i]* qui vérifie la condition précédente, on met à jour le *Min* par *V[i]* et la valeur de *Pos* par *i*.

Organigramme



Déroulement

Déroulement du programme (ou l'algorithme) pour *n=6* et *V= [5, 12, -1, 2, -8, 10]*.

Les valeur de *n* et *V* données précédemment correspondent aux entrées du programme (c'est à dire les lignes du code n° *08* à *12*). Le déroulement se fait pour la partie des traitements (n° de ligne entre 15 et 21).

Instructions	***Variables du Programme***				
	n	***i***	***V***	***Min***	***Pos***
Entrées (N° de ligne : 08-12)	5	/	[5, 12, -1, 2, -8, 10]	/	/
Min := V[1] ; Pos := 1 ;	"	/		5	1
For i :=2 if V[2]<Min (12 < 5)→ False (on n'entre pas au test **If**)	"	2			
For i :=3 if V[3]<Min (-1 < 5)→ True (on entre au test **If**) Min := V[i] ; (Min ← -1) Pos := i; (Pos ← 3)	"	3	"	-1	3
For i :=4 if V[4]<Min (2 < -1)→ False (on n'entre pas au test **If**)	"	4	"	"	"
For i :=5 if V[5]<Min (-8 < 5)→ True (on entre au test **If**) Min := V[i] ; (Min ← -8) Pos := i; (Pos ← 5)	"	5	"	-8	5
For i :=6 if V[6]<Min (10 < -8)→ False (on n'entre pas au test **If**) Arrêter la boucle **For** (car *i* = *n*)	"	6	"	"	"
Write(Min, Pos)	"	"	"	-8	5

Donc le min est **-8** et sa position est la case N° **5**.

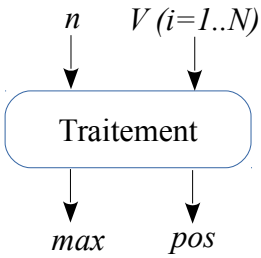
1- Recherche du max et sa position dans vecteur

L'algorithme

```
Algorithme exo4_2_Max
      Variables
            V : Tableau [1..100] de Réel
            i,n, Pos : entier   Max:Réel
Début
      Lire(n)
      Pour i←1 à n faire
                  Lire( V[i] )
      Fin-Pour

      Max ← V[1]
      Pos ← 1
      Pour i←2 à n faire
            Si T[i] > Max alors
               Max ←  V[i]
               Pos ← i
            Fin-Si
      Fin-Pour


      Écrire( Max, Pos)
Fin
```



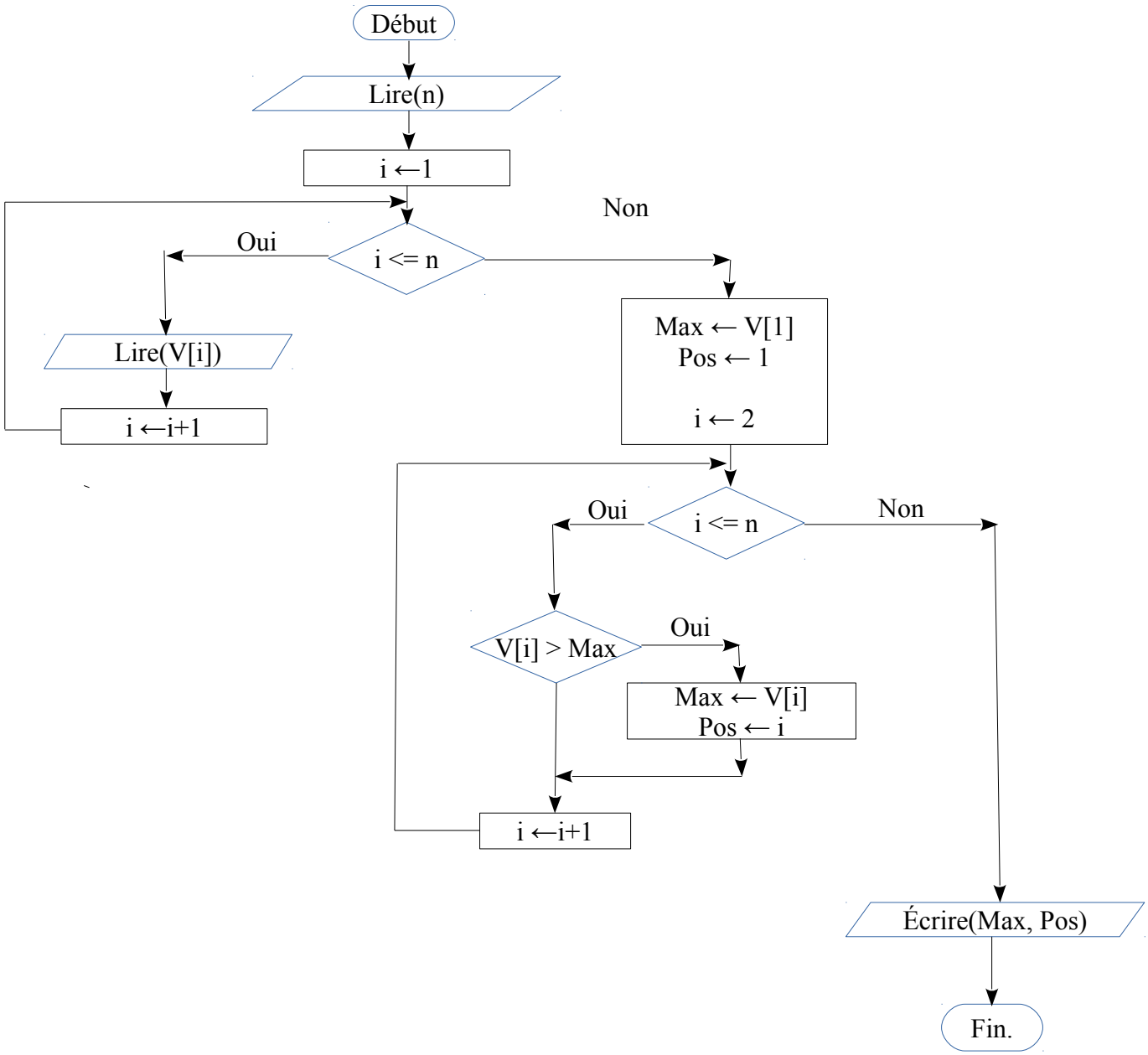
Le programme PASCAL

```
01 Program exo4_2_Max;
02 Uses wincrt ;
03 var
04      T, V : array[1..100] of Real;
05      i, n, Pos : integer ;  Max:Real;
06 Begin
07      {Les entrées}
08      Write('Donner la taille du vecteur V : ');
09      Read(n);
10      Writeln('Donner les composantes du vecteur V : ');
11      For i:=1 to n do
12                  Read( V[i] );
13
14      {Les traitements}
15      Max:=V[1]; Pos:=1; {On suppose que le Max est la valeur de la première case}
16      For i:=2 to n do    {On parcours toutes les autres cases}
17            If V[i] > Max then {Pour chaque case qui est supérieur au Max}
18             begin
19               Max := V[i]; {On actualise le Max (corriger le Max) }
20               Pos := i;    {On actualise sa position}
21             end;
22
23      {Les sorties}
24      Write('Le Max du V est : ',Max:2:2,' et sa position est : ', Pos);
25 End.
```

Explication

Le Même principe que la recherche du Min. Juste la condition qui est modifiée (*V[i] > Max*)

Organigramme



Déroulement

Déroulement du programme (ou l'algorithme) pour *n=6* et *V= [5, 12, -1, 2, -8, 10]*.

Les valeur de *n* et *V* données précédemment correspondent aux entrées du programme (c'est à dire les lignes du code n° *08* à *12*). Le déroulement se fait pour la partie des traitements (n° de ligne entre 15 et 21).

Instructions	***Variables du Programme***				
	n	***i***	***V***	***Max***	***Pos***
Entrées (N° de ligne : 08-12)	5	/	[5, 12, -1, 2, -8, 10]	/	/
Max := V[1] ; Pos := 1 ;	"	/		5	1
For i :=2 if V[2]>Max (12 > 5)→ True (on entre au test **If**) Max := V[i] ; (Max ← 12) Pos := i; (Pos ← 2)	"	2		12	2
For i :=3 if V[3]]>Max (-1 > 12)→ False (on n'entre pas au test **If**)	"	3	"	"	"
For i :=4 if V[4]]>Max (2 > 12)→ False (on n'entre pas au test **If**)	"	4	"	"	"
For i :=5 if V[5]<Min (-8 > 12)→ False (on n'entre pas au test **If**)	"	5	"	"	"
For i :=6 if V[6]>Max (10 > 12)→ False (on n'entre pas au test **If**) Arrêter la boucle **For** (car *i* = *n*)	"	6	"	"	"
Write(Max, Pos)	"	"	"	12	2

Donc le max est **12** et sa position est la case N° **2**.

Exercice 5 : La recherche d'une valeur dans un vecteur

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

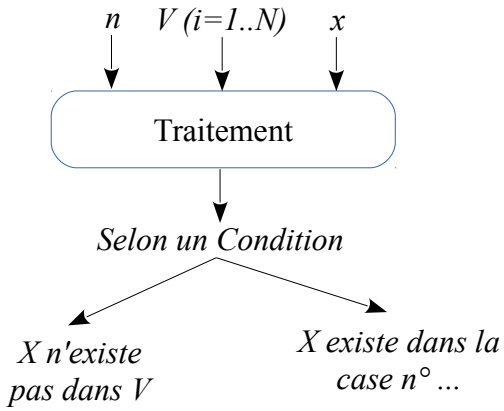
Solution ***Solution 01 : utiliser un booléen Trouve***

L'algorithme

```
Algorithme exo5_a
      Variables
            V : Tableau [1..100] de Réel
            i,n, Pos : entier   x:Réel
            Trouve : booléen
Début
  Lire(n)
  Pour i←1 à n faire
    Lire( V[i] )
  Fin-Pour
  Lire(x)

  Trouve ← False     i ← 1
  Tantque (i<=n) ET (Trouve = False) Faire
    Si T[i] = x alors
      Trouve ←  True
      Pos ← i
    Fin-Si
    i ← i+1
  Fin-Tantque

  Si Trouve = True Alors
    Écrire('X existe dans la position : ', Pos)
  Sinon
    Écrire('X n''existe pas dans V')
  Fin-Si
Fin
```



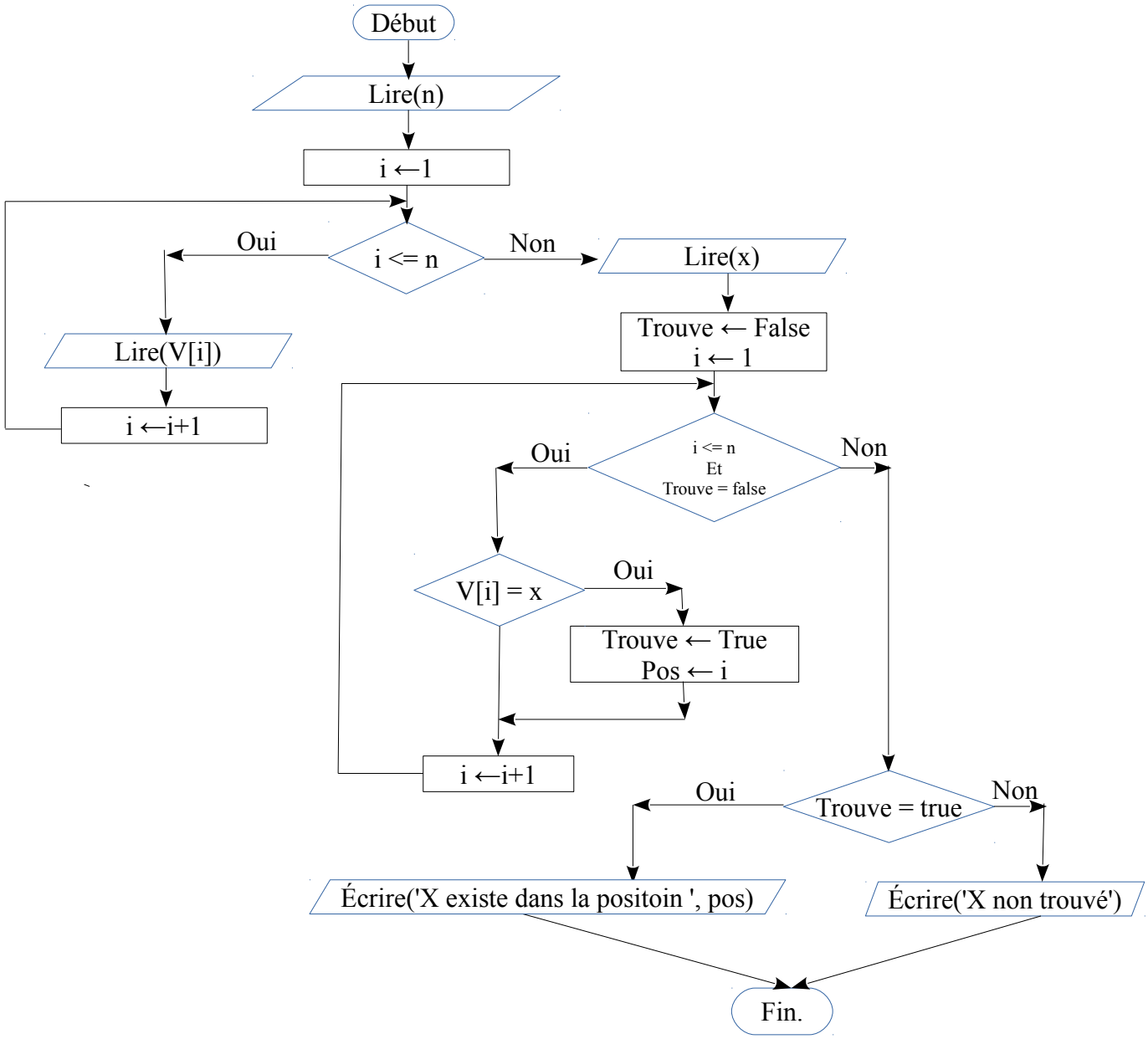
Le programme PASCAL

```
01 Program exo5_a;
02 Uses wincrt ;
03 var
04      T, V : array[1..100] of Real;
05      i, n, Pos : integer ;  x:Real; Trouve:boolean;
06 Begin
07      {Les entrées}
08      Write('Donner la taille du vecteur V : ');
09      Read(n);
10      Writeln('Donner les composantes du vecteur V : ');
11      For i:=1 to n do
12                 Read( V[i] );
13      Write('Donner la valeur de x : '); Read(x);
14
15      {Les traitements}
16      Trouve:=false; i:=1; {Initialiser Trouve à false}
17      While (i<=n) AND (Trouve = false) do
18       begin
19            If V[i]=x then {Si une case égale à x}
20             begin
21                Trouve:=true; Pos:=i;
22             End;
23            i := i+1;
24       end;
25
26      {Les sorties}
27      If Trouve=true Then
28         Write('x existe dans le vecteur et sa positon est : ', Pos)
29      Else
30         Write('x n''existe pas dans le vecteur');
31 End.
```

Explication

Les entrées du programme sont le vecteur V et la valeur de x à rechercher. Il y a une boucle de recherche qui permet de parcourir toutes les case de 1 jusqu'à n et elle s'arrête dans deux cas : Soit V[i]=x (donc on trouve la valeur x et on continue pas la recherche), soit i>n (on a parcouru toutes les case sans trouver aucun valeur). La variable booléenne Trouve représente l'existence de x dans V ou non : Si Trouve = True, donc on a trouvé x sinon on l'a pas trouve (ou on a pas encore trouvé x). La boucle de recherche **`Tantque`** `(i<=n)` **`ET`** `(Trouve = False)` **`Faire`** signifie : Tantque on a pas terminé le tableau (i<=n) et toujours on pas encore trouve x (Trouve = False) donc on entre à la boucle, une teste permet de vérifie la case actuelle si est égale à x (Si V[i]=x Alors), donc ce cas on sauvegarde la position (qui est i : pos ← i) et on indique qu'on a trouvé la valeur x (Trouve←true), ensuite, on se met à la case suivante : i←i+1, et ainsi de suite.

Organigramme



Déroulement

1- Déroulement du programme (ou l'algorithme) pour *n=6* et *V= [5, 12, -1, 2, -8, 10]*. et *x=2*

Instructions	***Variables du Programme***						
	n	***i***	***V***	***x***	***Trouve***	***Pos***	***Réponse (résultat)***
Entrées (N° de ligne : 08-13)	6	/	[5, 12, -1, 2, -8, 10]	2	/	/	
Trouve := false ; i:=1 ;	"	1		"	False	"	

While (i<=n)AND(Trouve=false)
(**1**<=6)And(false=false)
→ True (on entre à While)
If (V[**1**] = x) (5=2) → False
(On n'entre pas à If)
i:= i+1 ; (i=2)
	"	**2**		"	"	"	

While (i<=n)AND(Trouve=false)
(**2**<=6)And(false=false)
→ True (on entre à While)
If (V[**2**] = x) (12=2) → False
(On n'entre pas à If)
i:= i+1 ; (i=3)
	"	**3**	"	"	"	"	

While (i<=n)AND(Trouve=false)
(**3**<=6)And(false=false)
→ True (on entre à While)
If (V[**3**] = x) (-1=2) → False
(On n'entre pas à If)
i:= i+1 ; (i=4)
	"	**4**	"	"	"	"	

While (i<=n)AND(Trouve=false)
(**4**<=6)And(false=false)
→ True (on entre à While)
If (V[**4**] = x) (2=2) → True
(On entre à If)
Trouve := true
Pos:= i (Pos=4)
i:= i+1 ; (i=5)
	"	**5**	"	"	**True**	**4**	

While (i<=n)AND(Trouve=false) (**5**<=6)And(true=false) True And False→ False (on n'entre pas à While)	"		"	"	"	"	
If Trouve = true → True (on entre à If) Write('x existe dans ', pos)	"		"	"	"	"	X existe dans la position 4

Donc le programme affichera le résultat : **X existe dans le vecteur et sa position est : 4**

1- Déroulement du programme (ou l'algorithme) pour *n=6* et *V= [5, 12, -1, 2, -8, 10].* et *x=55*

Instructions	***Variables du Programme***						
	n	***i***	***V***	***x***	***Trouve***	***Pos***	***Réponse (résultat)***
Entrées (N° de ligne : 08-13)	6	/	[5, 12, -1, 2, -8, 10]	5	/	/	
Trouve := false ; i:=1 ;	"	1		"	**False**	"	

While (i<=n)AND(Trouve=false)
(**1**<=6)And(false=false)
→ True (on entre à While)
If (V[**1**] = x) (5=55) → False
(On n'entre pas à If)
i:= i+1 ; (i=2)
	"	**2**		"	"	"	

While (i<=n)AND(Trouve=false)
(**2**<=6)And(false=false)
→ True (on entre à While)
If (V[**2**] = x) (12=55) → False
(On n'entre pas à If)
i:= i+1 ; (i=3)
	"	**3**	"	"	"	"	

While (i<=n)AND(Trouve=false)
(**3**<=6)And(false=false)
→ True (on entre à While)
If (V[**3**] = x) (-1=55) → False
(On n'entre pas à If)
i:= i+1 ; (i=4)
	"	**4**	"	"	"	"	

While (i<=n)AND(Trouve=false)
(**4**<=6)And(false=false)
→ True (on entre à While)
If (V[**4**] = x) (2=55) → False
(On n'entre pas à If)
i:= i+1 ; (i=5)
	"	**5**	"	"	"	""	

While (i<=n)AND(Trouve=false)
(**5**<=6)And(false=false)
→ True (on entre à While)
If (V[**5**] = x) (-8=55) → False
(On n'entre pas à If)
i:= i+1 ; (i=6)
	"	**6**	"	"	"	"	

While (i<=n)AND(Trouve=false)
(**6**<=6)And(false=false)
→ True (on entre à While)
If (V[**5**] = x) (10=55) → False
(On n'entre pas à If)
i:= i+1 ; (i=7)
		7	"	"	"	"	

While (i<=n)AND(Trouve=false) (**7**<=6)And(false=false) False And True → False (on n'entre pas à While)		"	"	"	"	"	
If Trouve = true → Fase (on n'entre pas à If → On entre à Else) Write('x n''existe pas dans ')	"		"	"			X n'existe pas dans le vecteur

Donc le programme affichera le résultat : **X n'existe pas dans le vecteur**.

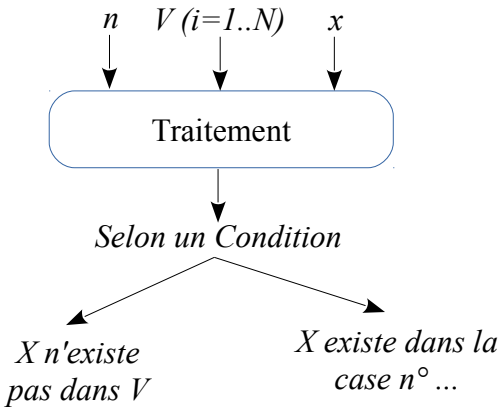
Solution 02 : utiliser l'indice i (selon la valeur de i)

L'algorithme

```
Algorithme exo5_b
      Variables
            V : Tableau [1..100] de Réel
            i,n: entier   x:Réel
Début
  Lire(n)
  Pour i←1 à n faire
    Lire( V[i] )
  Fin-Pour
  Lire(x)

  i ← 1
  Tantque (i <= n) ET (T[i] <> x) Faire
    i ← i+1
  Fin-Tantque

  Si i<=n Alors
    Écrire('X existe dans la position : ', i)
  Sinon
    Écrire('X n''existe pas dans V')
  Fin-Si
Fin
```



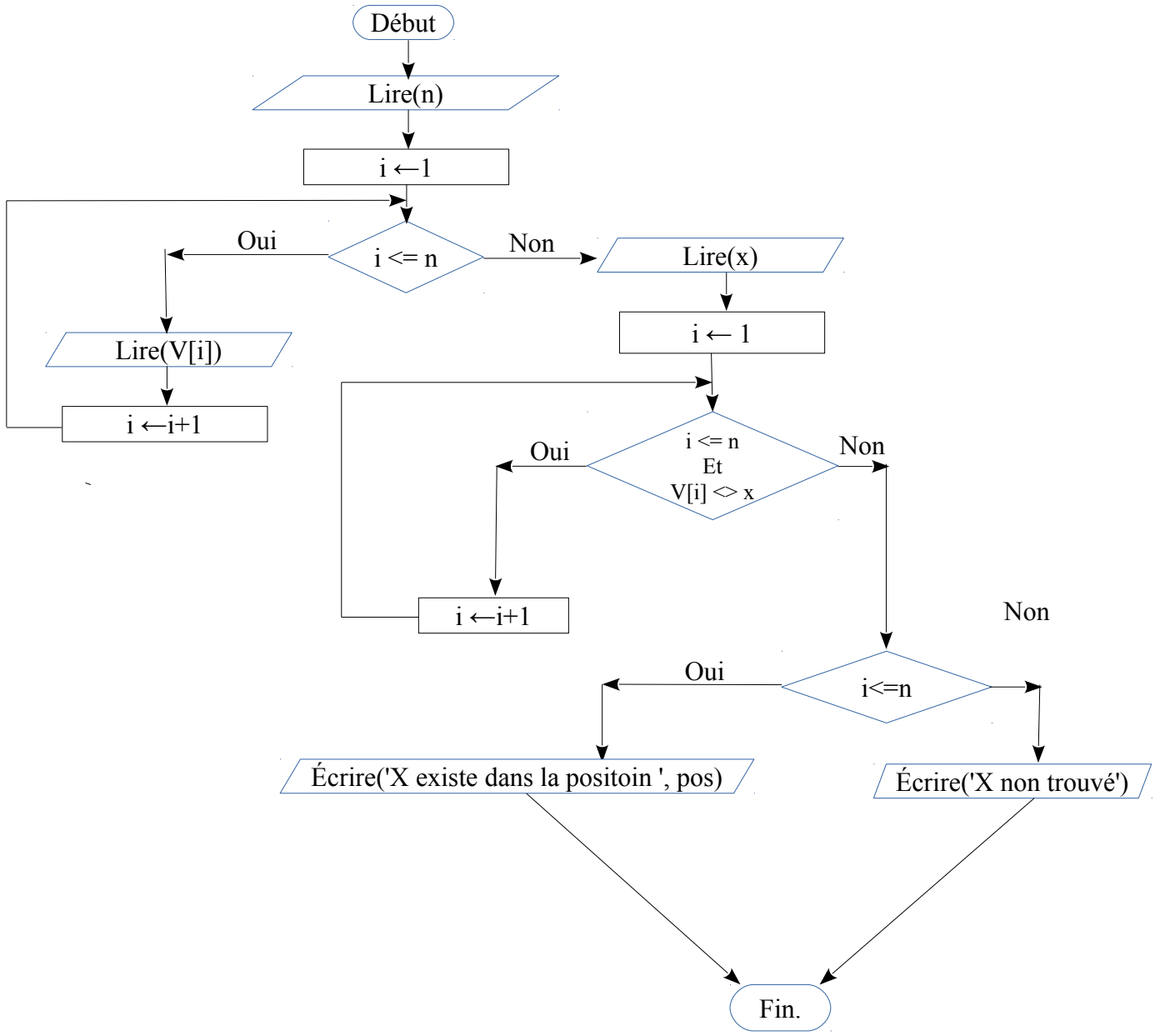
Le programme PASCAL

```
01  Program exo5_b;
02  Uses wincrt ;
03  var
04       T, V : array[1..100] of Real;
05       i, n : integer ;  x:Real;
06  Begin
07       {Les entrées}
08       Write('Donner la taille du vecteur V : ');
09       Read(n);
10       Writeln('Donner les composantes du vecteur V : ');
11       For i:=1 to n do
12                 Read( V[i] );
13       Write('Donner la valeur de x : '); Read(x);
14
15       {Les traitements}
16       i:=1; {Initialiser Trouve à false}
17       While (i<=n) AND (V[i] <> x) do
18          i := i+1;
19
20       {Les sorties}
21       If i <= n Then
22          Write('x existe dans le vecteur et sa positon est : ', i)
23       Else
24          Write('x n''existe pas dans le vecteur');
25  End.
```

Explication

Dans cette solution, la boucle recherche s'arrête lorsque la condition `(i<=n)` **AND** `(V[i]<>x)` est fausse, c-à-d, elle devient `(i>n)` **OR** `(V[i]=x)` (Soit i>n ou bien V[i]=x). Lorsque la boucle est terminée, on aura deux cas par rapport à la valeur de *i* : Soit i>n ou bien i<=n. Dans le cas où i>n, çà veut dire qu'on a parcouru toutes cases du vecteur sans trouver aucun élément qui égale à x, sinon (i<n), ça veut dire que la boucle a été arrêtée à la case i où on trouvé la valeur x, c'est à dire V[i]=x.

Organigramme



Déroulement

1- Déroulement du programme (ou l'algorithme) pour *n=6* et *V= [5, 12, -1, 2, -8, 10]*. et *x=2*

Instructions	***Variables du Programme***				
	n	***i***	***V***	***x***	***Réponse (résultat)***
Entrées (N° de ligne : 08-13)	6	/	[5, 12, -1, 2, -8, 10]	2	
i:=1 ;	"	1		"	
While (**i**<=n)AND(V[**i**]<>x) (**1**<=6)And(5<>2) (True)And(True)→ True (on entre à While) i:= i+1 ; (i=2)	"	**2**	"	"	
While (**i**<=n)AND(V[**i**]<>x) (**2**<=6)And(12<>2) (True)And(True)→ True (on entre à While) i:= i+1 ; (i=3)	"	**3**	"	"	
While (**i**<=n)AND(V[**i**]<>x) (**3**<=6)And(-1<>2) (True)And(True)→ True (on entre à While) i:= i+1 ; (i=4)	"	**4**	"	"	
While (**i**<=n)AND(V[**i**]<>x) (**4**<=6)And(2<>2) (True)And(False)→ False (on n'entre pas à While)	"	"	"	"	
If i<=n (4 ≤ 6) → True (on entre à If) Write('x existe dans ', i)	"		"	"	X existe dans la position 4

Donc le programme affichera le résultat : **X existe dans le vecteur et sa position est : 4**

1- Déroulement du programme (ou l'algorithme) pour *n=6* et *V= [5, 12, -1, 2, -8, 10].* et *x=55*

Instructions	***Variables du Programme***				
	n	***i***	***V***	***x***	***Réponse (résultat)***
Entrées (N° de ligne : 08-13)	6	/	[5, 12, -1, 2, -8, 10]	2	
i:=1 ;	"	1		"	
While (**i**<=n)AND(V[**i**]<>x) (**1**<=6)And(5<>55) (True)And(True)→ True (on entre à While) i:= i+1 ; (i=2)	"	**2**	"	"	
While (**i**<=n)AND(V[**i**]<>x) (**2**<=6)And(12<>55) (True)And(True)→ True (on entre à While) i:= i+1 ; (i=3)	"	**3**	"	"	
While (**i**<=n)AND(V[**i**]<>x) (**3**<=6)And(-1<>55) (True)And(True)→ True (on entre à While) i:= i+1 ; (i=4)	"	**4**	"	"	
While (**i**<=n)AND(V[**i**]<>x) (**4**<=6)And(2<>55) (True)And(True)→ True (on entre à While) i:= i+1 ; (i=5)	"	**5**	"	"	
While (**i**<=n)AND(V[**i**]<>x) (**5**<=6)And(-8<>55) (True)And(True)→ True (on entre à While) i:= i+1 ; (i=6)		**6**			
While (**i**<=n)AND(V[**i**]<>x) (**6**<=6)And(10<>55) (True)And(True)→ True (on entre à While) i:= i+1 ; (i=7)		**7**			
While (**i**<=n)AND(V[**i**]<>x) (**7**<=6)And(0<>2) (False)And(True)→ False (on n'entre pas à While)					
If i<=n (7 ≤ 6) → False (on n'entre pas à If, on entre à Else) Write('x n''existe pas dans le vecteur')	"		"	"	X n'existe pas dans le vecteur

Donc le programme affichera le résultat : **X n'existe pas dans le vecteur**.

Exercice 6 : Lecture et affichage d'une matrice

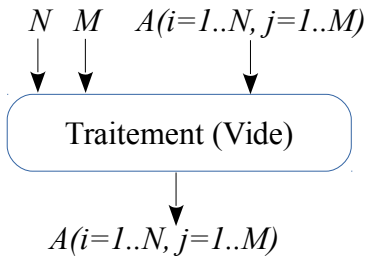
Écrire un algorithme/programme PASCAL qui permet de lire et afficher une matrice de type réel A de N lignes et M colonnes.

Solution ***Solution 01 : utiliser un booléen Trouve***

L'algorithme

```
Algorithme exo6_lecture_ecriture
      Variables
            A:Tableau [1..10,1..10] de Réel
            i,j, N, M : entier
Début
  Lire(N, M)
  Pour i←1 à N faire
    Pour j←1 à M faire
      Lire( A[i,j] )
    Fin-Pour
  Fin-Pour


  Pour i←1 à N faire
    Pour j←1 à M faire
      Écrire( A[i,j] )
    Fin-Pour
  Fin-Pour
Fin
```



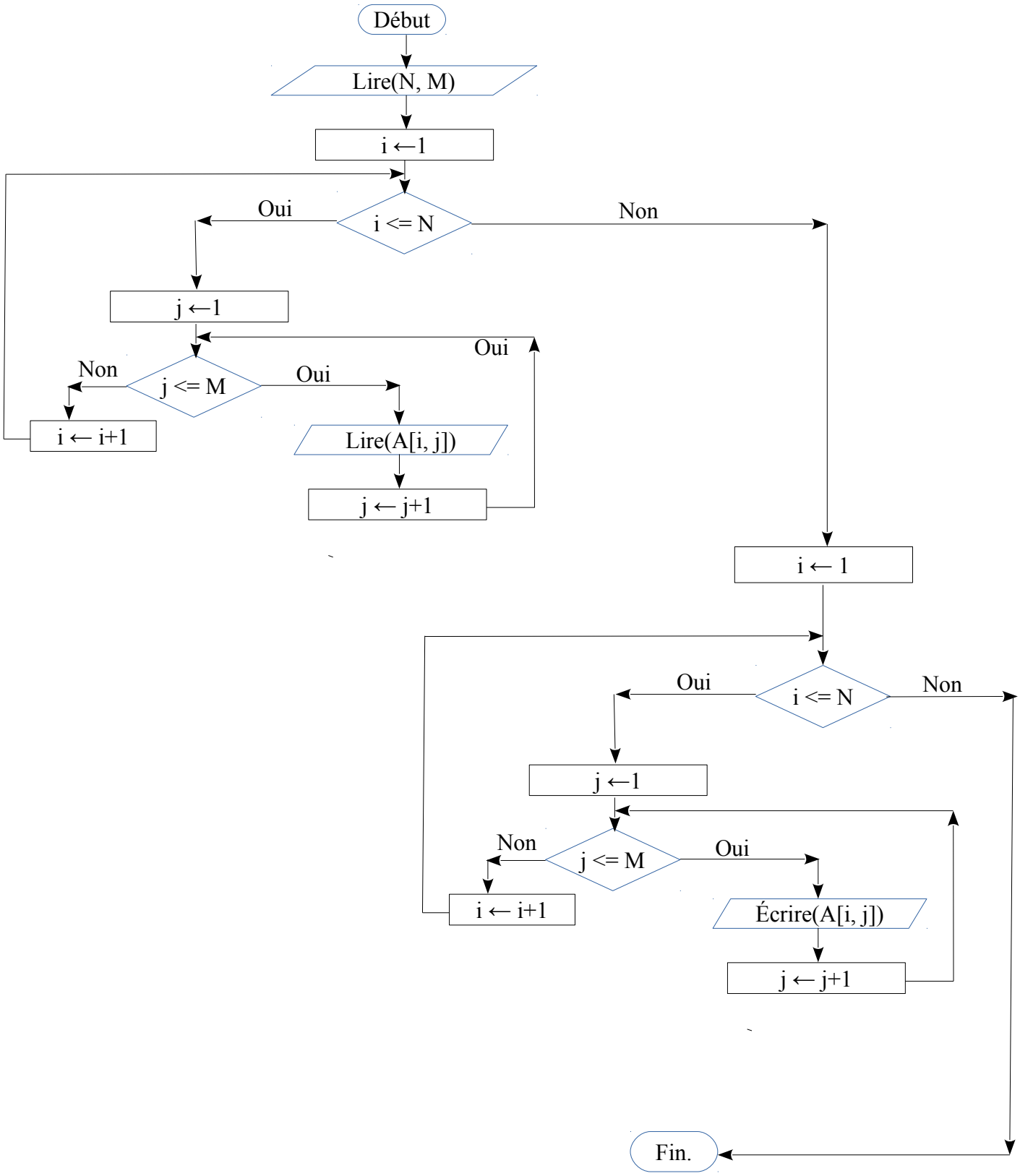
Le programme PASCAL

```
01  Program exo6_lecture_ecriture;
02  Uses wincrt ;
03  var
04       A : array[1..10, 1..10] of Real;
05       i, j, N, M : integer;
06  Begin
07       {Les entrées}
08       Write('Donner les dimensions de la Matrice A : ');
09       Read(N, M);
10       Writeln('Donner les composantes de la matrice A : ');
11       For i:=1 to N do
12            For j:=1 to M do
13                 Read( A[i, j] );
14
15       {Les traitements}
16
17
18       {Les sorties}
19       Writeln('L''affichage de la matrice A : ');
20       For i:=1 to N do
21        begin
22            For j:=1 to M do
23                 Write( V[i]:8:2 );
24
25            Writeln; {Sauter la ligne}
26        end;
27  End.
```

Explication

Ce programme illustre comment déclarer, lire et écrire une matrice d'ordre N*M. Pour lire ou écrire une matrice, on aura besoin de deux boucles imbriqués, la boucle externe pour les lignes (**For** i) et la boucle interne pour les colonnes (**For** j).

Organigramme



Exercice 7 : Somme, Moyenne et Produit des éléments d'une matrice

Soit une matrice A réelle d'ordre NxM.

1 Écrire un algorithme/programme PASCAL qui calcule la somme et la moyenne des éléments de la matrice A.

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

Solution

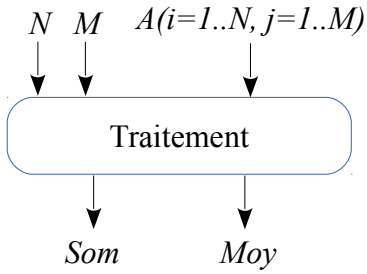
1- Somme et la moyenne des éléments de la matrice A

L'algorithme

```
Algorithme exo7_1
      Variables
            A:Tableau [1..10,1..10] de Réel
            i,j, N, M : entier
            Som, Moy : réel
Début
  Lire(N, M)
  Pour i←1 à N faire
    Pour j←1 à M faire
      Lire( A[i,j] )
    Fin-Pour
  Fin-Pour

  Som ← 0
  Pour i←1 à N faire
    Pour j←1 à M faire
        Som ← Som + A[i, j]
    Fin-Pour
  Fin-Pour
  Moy ← Som / (N*M)

  Écrire (Som, Moy);
Fin
```



Le programme PASCAL

```
01  Program exo7_1;
02  Uses wincrt ;
03  var
04       A : array[1..10, 1..10] of Real;
05       i, j, N, M : integer;
06       Som, Moy:real;
07  Begin
08       {Les entrées}
09       Write('Donner les dimensions de la Matrice A : ');
10       Read(N, M);
11       Writeln('Donner les composantes de la matrice A : ');
12       For i:=1 to N do
13            For j:=1 to M do
14                 Read( A[i, j] );
15
16       {Les traitements}
17       Som:=0; {Initialiser toujours la somme à ZÉRO}
18       For i:=1 to N do
19            For j:=1 to M do
20                 Som := Som + A[i, j];
21
22       Moy := Som / (N*M); {La moyenne égale à la somme sur nombre d'éléments}
23
24       {Les sorties}
25       Write('La somme est : ',Som:2:2,'  La Moyenne est : ',Moy:2:2);
26  End.
27
```

Explication

La somme des éléments d"une matrices est donnée par la formule : A[1, 1]+A[1, 2]+ … + A[1,M]+ A[2, 1]+A[2, 2]+ … + A[2,M]+ … …. … + A[N, 1]+A[N, 2]+ … + A[N,M]. On peut écrire cette formule comme suit :

Som = A[1, 1]+A[1, 2]+ … + A[1,M] //La somme de la première ligne

+ A[2, 1]+A[2, 2]+ … + A[2,M] //La somme de la deuxième ligne

+ A[3, 1]+A[3, 2]+ … + A[3,M] //La somme de la troisième ligne

……

……

+ A[N, 1]+A[N, 2]+ … + A[N,M] //La somme de la dernière ligne

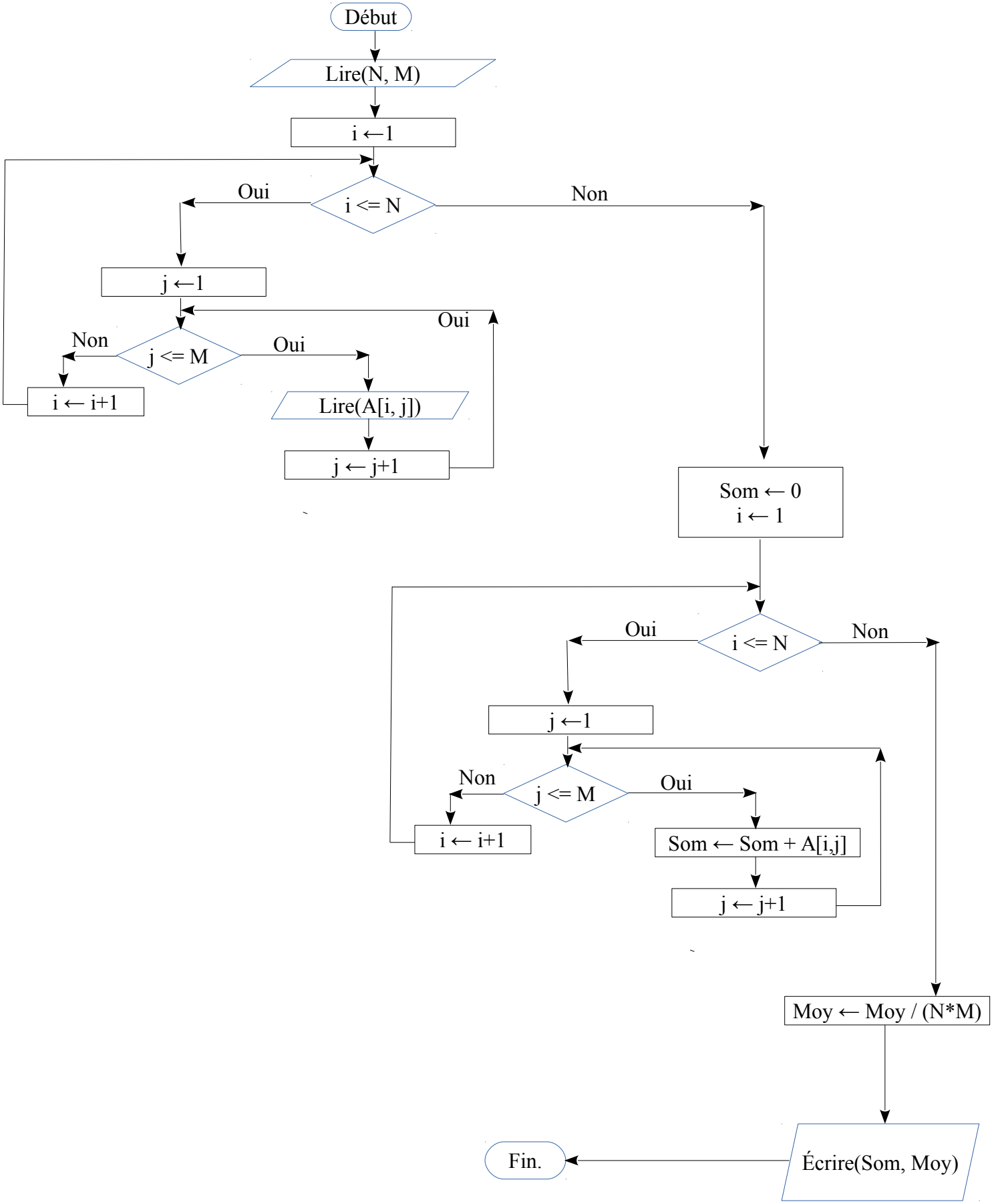
Ça devient :

$$\text{Som} = \sum_{j=1}^{M} A[1,j] + \sum_{j=1}^{M} A[2,j] + \ldots + \sum_{j=1}^{M} A[i,j] + \ldots + \sum_{j=1}^{M} A[N,j]$$

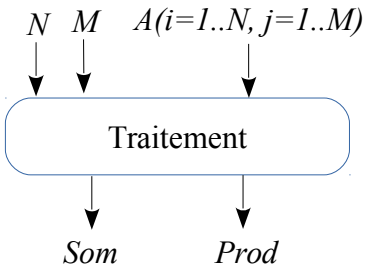
$$\text{Som} = \sum_{i=1}^{N} \sum_{j=1}^{M} A[i,j]$$

Chaque symbole de somme sera replacé par une boucle POUR, donc, on aura deux boucles imbriquées, qui permettent de répéter l'instruction : *Som* ← *Som* + *A[i, j]*

Organigramme



2- Somme de chaque ligne et Produit de chaque colonnes



L'algorithme

```
Algorithme exo7_2
     Variables
          A:Tableau [1..10,1..10] de Réel
          i,j, N, M : entier
          Som, Moy : Tableau[1..10] de Réel
Début
  Lire(N, M)
  Pour i←1 à N faire
    Pour j←1 à M faire
     Lire( A[i,j] )
    Fin-Pour
  Fin-Pour

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

  Pour j←1 à M faire
    Prod[j] ← 1
    Pour i←1 à N faire
       Prod[j] ← Prod[j] * A[i, j]
    Fin-Pour
  Fin-Pour

  Pour i←1 à N faire
    Écrire( Som[i] )
  Fin-Pour

  Pour j←1 à M faire
    Écrire( Prod[j] )
  Fin-Pour
Fin
```

Le programme PASCAL

```
01  Program exo7_2;
02  Uses wincrt ;
03  var
04      A : array[1..10, 1..10] of Real;
05      i, j, N, M : integer;
06      Som, Prod : array[1..10] of Real;
07  Begin
08      {Les entrées}
09      Write('Donner les dimensions de la Matrice A : ');
10      Read(N, M);
11      Writeln('Donner les composantes de la matrice A : ');
12      For i:=1 to N do
13           For j:=1 to M do
14                Read( A[i, j] );
15
16      {Les traitements}
17      For i:=1 to N do
18       begin
19           Som[i] := 0; {Initialiser la somme de la ligne i à 0}
20           For j:=1 to M do
21                Som[i] := Som[i] + A[i, j];
22       end;
23
24      For j:=1 to M do
25       begin
26           Prod[j] := 1; {Initialiser le produit de la colonne j à 1}
27           For i:=1 to N do
28                Prod[j] := Prod[j] * A[i, j];
29       end;
30
31      {Les sorties}
32      Writeln;
33      Writeln('La somme des lignes de la matrice A : ');
34      For i:=1 to N do
35           Write(Som[i]:10:2);
36
37      Writeln;
38      Writeln('Le produit des colonnes de la matrice A : ');
39      For j:=1 to M do
40           Write(Prod[i]:10:2);
41  End.
42
```

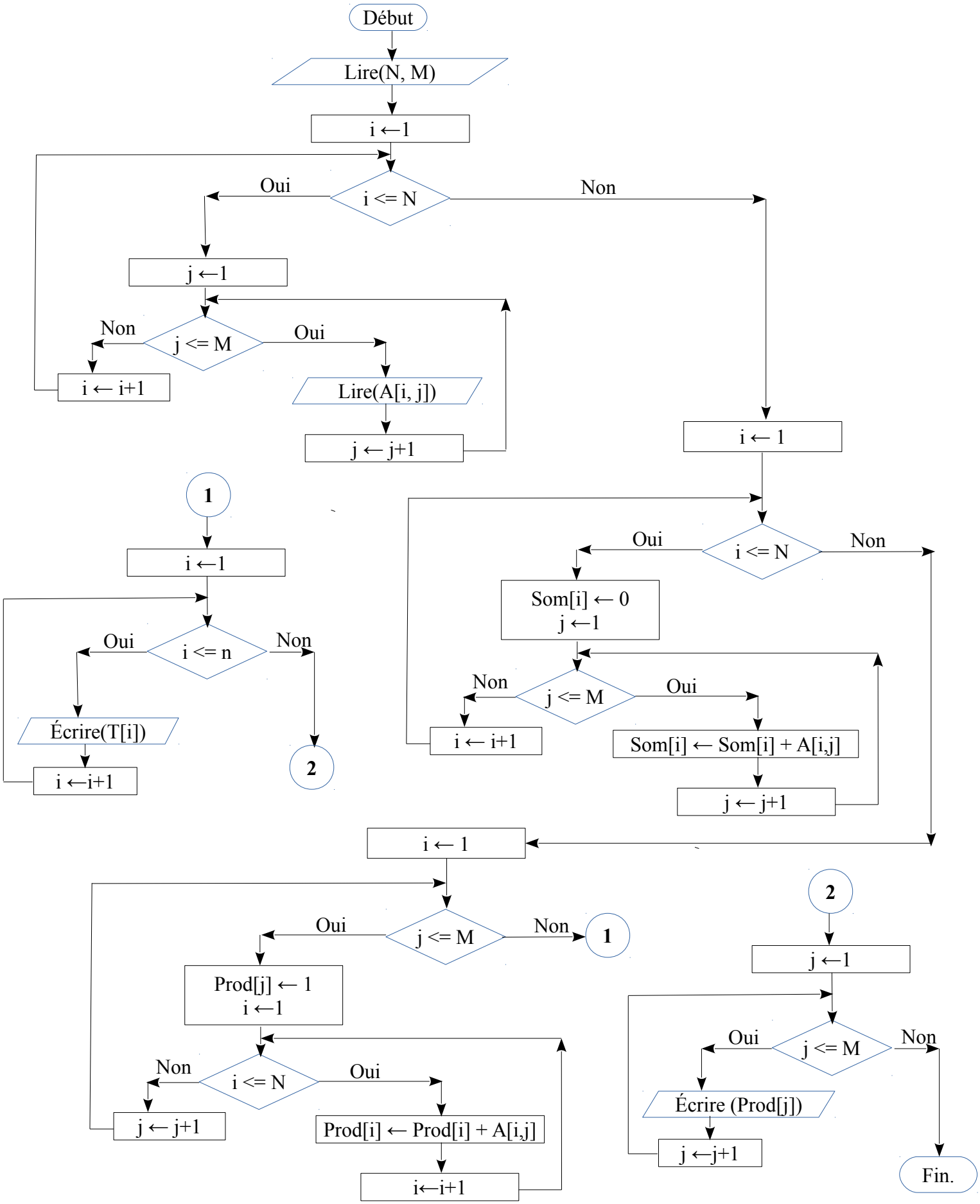
Explication

Pour une matrice A de N lignes et M Colonnes, on aura, pour les lignes N sommes et, pour les colonnes, M Produits. Donc, on déclare les sommes des lignes comme un vecteur *Som* dont la taille est N, et les produits des colonnes comme un vecteur *Prod* dont la taille est M.

Donc on aura, pour chaque ligne i : $Som[i] = \sum_{j=1}^{M} A[i,j]$

Et pour chaque colonne j : Prod[j] = $\prod_{i=1}^{N} A[i,j]$

Organigramme



Exercice 8 : Min et le Max dans une matrice et leurs positions

Soit A une matrice réelle d'ordre NxM.

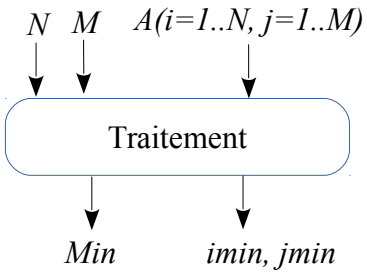
1 Écrire un algorithme/programme PASCAL qui permet de rechercher le plus petit élément dans la matrice A ainsi que sa position.

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

Solution

1- Recherche du min et sa position dans une matrice

L'algorithme




```
Algorithme exo8_1_Min
     Variables
          A : Tableau [1..10, 1..10] de Réel
          i, j, N, M, imin, jmin : entier
          Min:Réel
Début
     Lire(N, M)
     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


     Écrire( Min, imin, jmin)
Fin
```

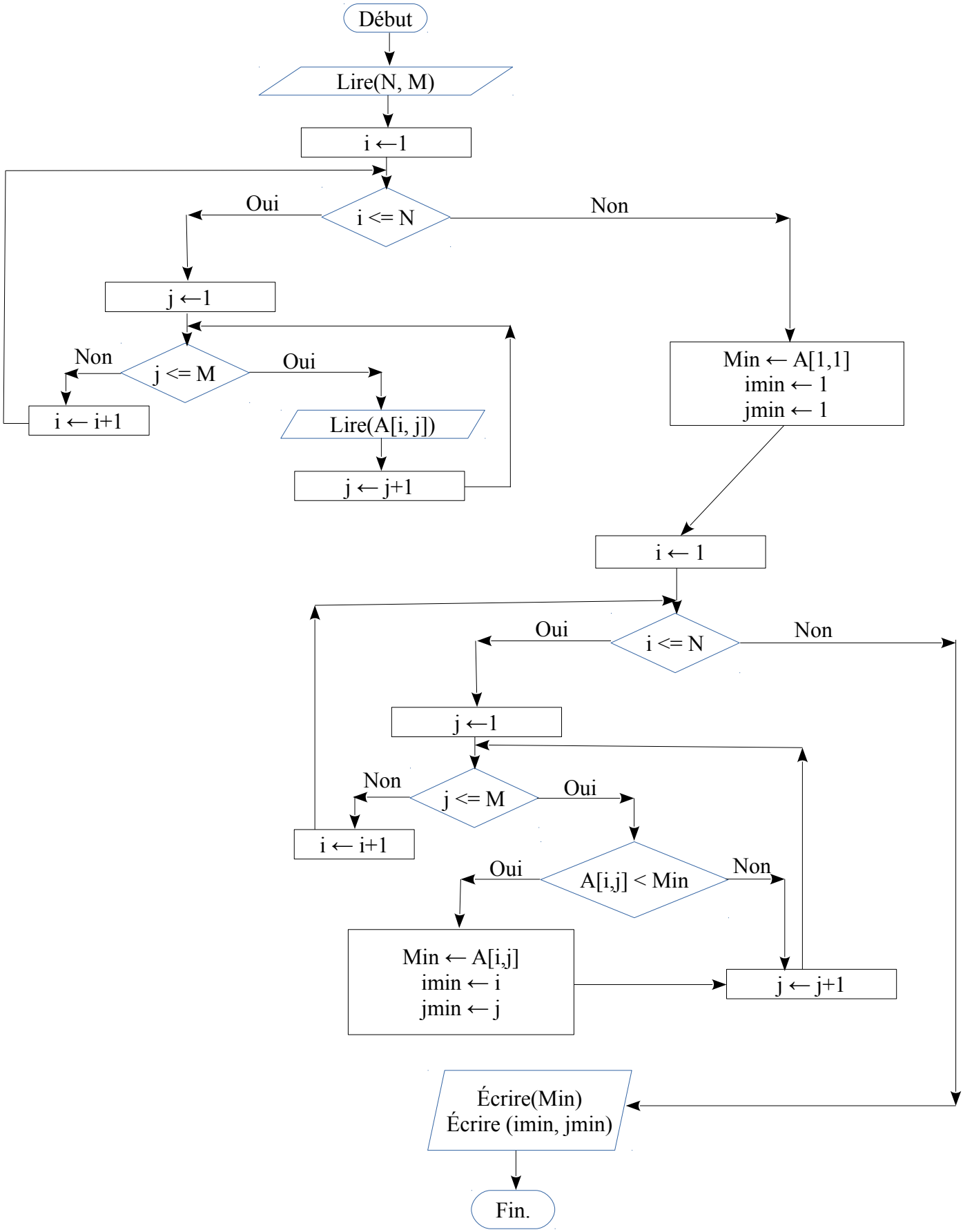
Le programme PASCAL

```
01  Program exo8_1_Min;
02  Uses wincrt ;
03  var
04      A : array[1..10, 1..10] of Real;
05      i, j, N, M, imin, jmin : integer;
06      Min : real;
07  Begin
08      {Les entrées}
09      Write('Donner les dimensions de la Matrice A : ');
10      Read(N, M);
11      Writeln('Donner les composantes de la matrice A : ');
12      For i:=1 to N do
13           For j:=1 to M do
14                Read( A[i, j] );
15
16      {Les traitements}
17      Min := A[1, 1];
18      imin:= 1;
19      jmin := 1;
20
21      For i:=1 to N do
22        For j:=1 to M do
23         if A[i, j] < Min Then
24           begin
25             Min := A[i,j];
26             imin := 1;
27             jmin := 1;
28           end;
29
30      {Les sorties}
31      Write('Min=', Min:2:2, 'Et sa position est : ', imin,' ',jmin);
32  End.
```

Explication

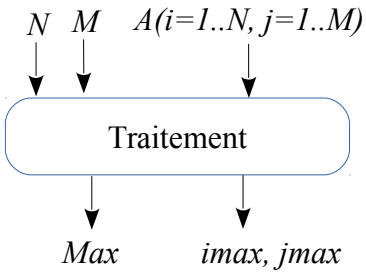
La solution consiste aux parcours des éléments de la matrice pour les comparer un à un à la valeur supposée minimale de la matrice (la ligne N° 23), dés qu'un élément est inférieur au minimum, ce dernier sera mis à jour ainsi que sa position imin (la ligne) et jmin (la colonne) (les lignes N° 25, 26 et 27). La valeur initiale du Min et la première case de la matrice, c'est à dire A[1,1], par conséquent, sa position est : imin=1 et jmin=1.

Organigramme



1- Recherche du max et sa position dans une matrice

L'algorithme




```
Algorithme exo8_2_Max
     Variables
          A : Tableau [1..10, 1..10] de Réel
          i, j, N, M, imax, jmax : entier
          Max:Réel
Début
     Lire(N, M)
     Pour i←1 à N faire
       Pour j←1 à M faire
          Lire( A[i, j] )
       Fin-Pour
     Fin-Pour

     Max ← A[1, 1]
     imax ← 1
     jmax ← 1

     Pour i←1 à N faire
       Pour j←1 à M faire
          Si A[i, j] > Max alors
             Max ←  A[i, j]
             imax ← i
             jmax ← j
          Fin-Si
       Fin-Pour
     Fin-Pour


     Écrire( Max, imax, jmax)
Fin
```

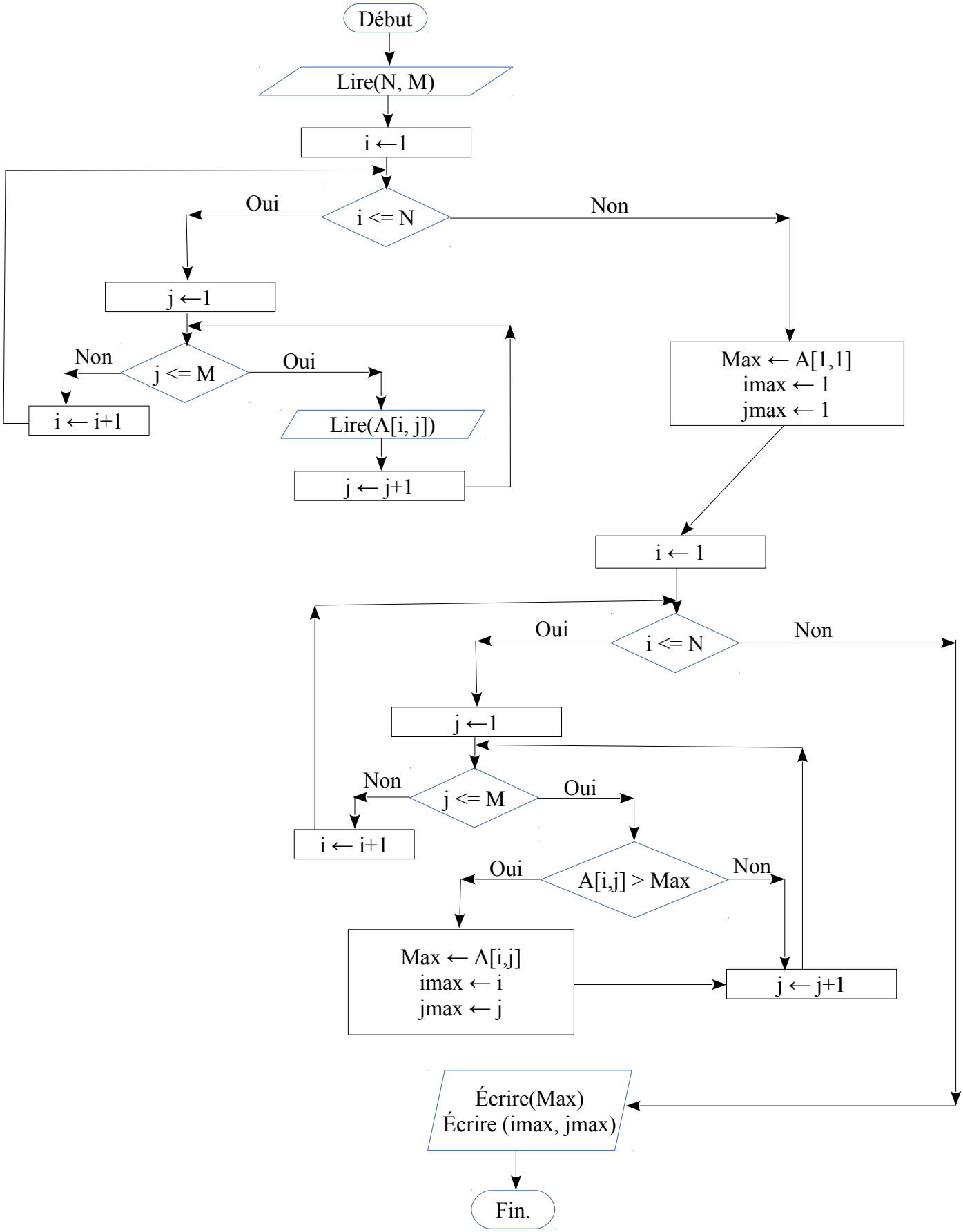
Le programme PASCAL

```
01  Program exo8_2_Max;
02  Uses wincrt ;
03  var
04      A : array[1..10, 1..10] of Real;
05      i, j, N, M, imax, jmax : integer;
06      Max : real;
07  Begin
08      {Les entrées}
09      Write('Donner les dimensions de la Matrice A : ');
10      Read(N, M);
11      Writeln('Donner les composantes de la matrice A : ');
12      For i:=1 to N do
13           For j:=1 to M do
14                Read( A[i, j] );
15
16      {Les traitements}
17      Max := A[1, 1];{On suppose que la première case est la max}
18      imax:= 1; {Donc sa position est la ligne 1}
19      jmax := 1; {et la colonne 1}
20
21      For i:=1 to N do {Pour toute ligne i}
22        For j:=1 to M do {Pour toute colonne j}
23         if A[i, j] > Max Then {Si la case A[i, j] est > au Max}
24           begin
25             Max := A[i,j]; {On actualise le Max}
26             imax := 1; {sa ligne imax}
27             jmax := 1; {sa colonne jmax}
28           end;
29
30      {Les sorties}
31      Write('Max=', Max:2:2, 'Et sa position est : ', imax,' ',jmax);
32  End.
```

Explication

Le même principe que le min et sa position

Organigramme



Exercice 9 : Transposée d'une matrice. Somme de deux matrices

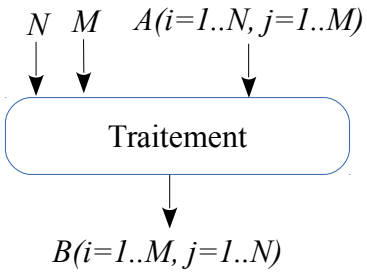
1 Écrire un algorithme/programme PASCAL qui permet de calculer la transposée d'une matrice réelle A d'ordre NxM.

2 Écrire un algorithme/programme PASCAL qui permet de réaliser la somme de matrices réelles A et B d'ordre NxM.

Solution

1- Transposée d'une matrice

L'algorithme



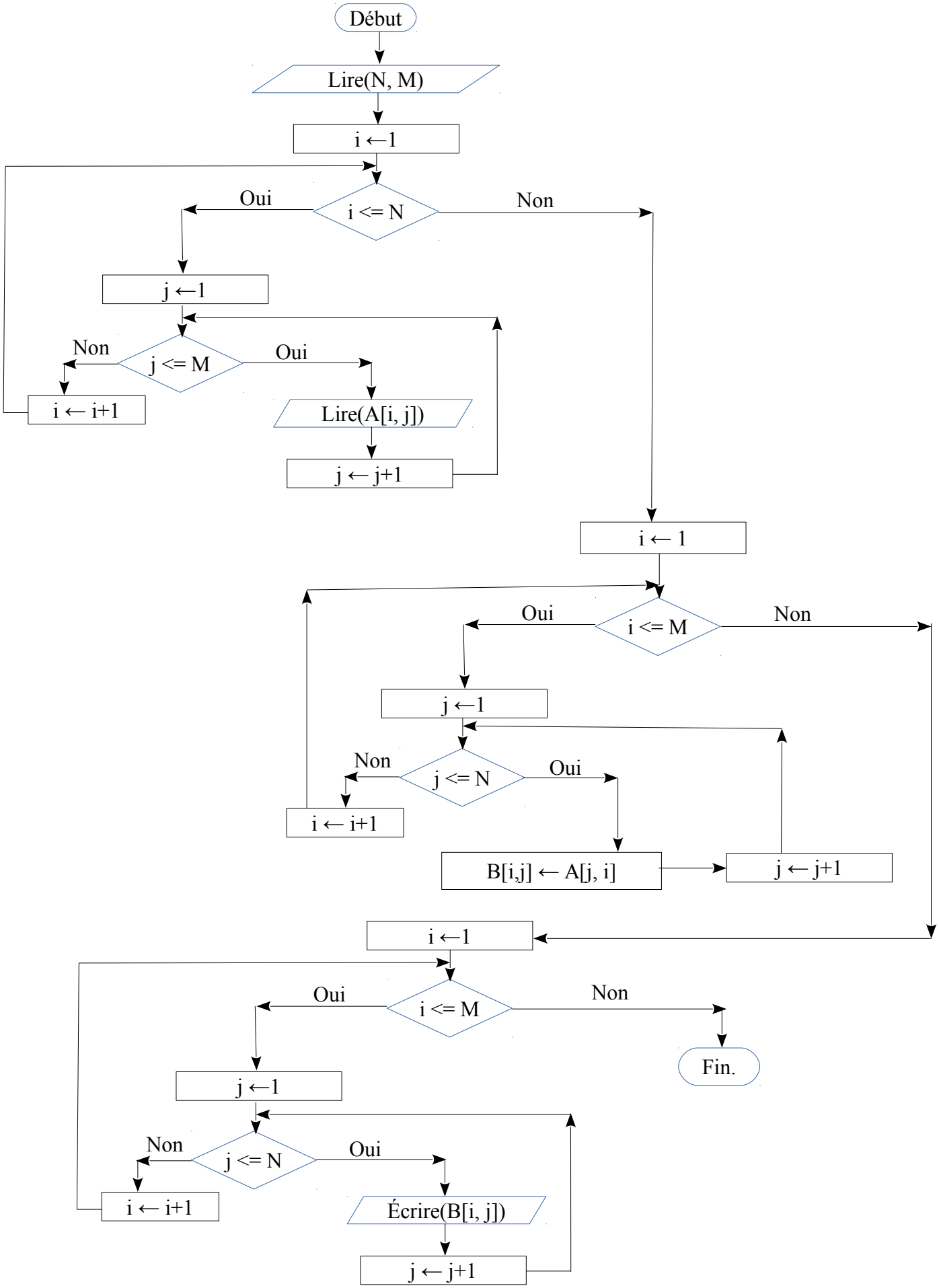
Explication

Le transposé d'une matrice A d'ordre NxM est un matrice B d'ordre MxN. Chaque ligne de A devient une colonne de B (ou chaque colonne de A devient une ligne pour B).

Chaque case B[i, j] correspond à la case A[j, i] tel que : i=1, …, M et j=1, …, N.

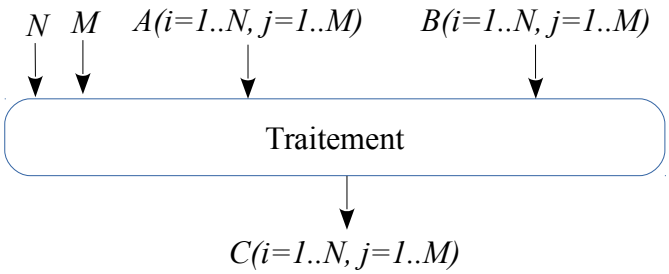
Essayer de faire l'algorithme et le programme PASCAL.

Organigramme



2- Somme de deux matrices

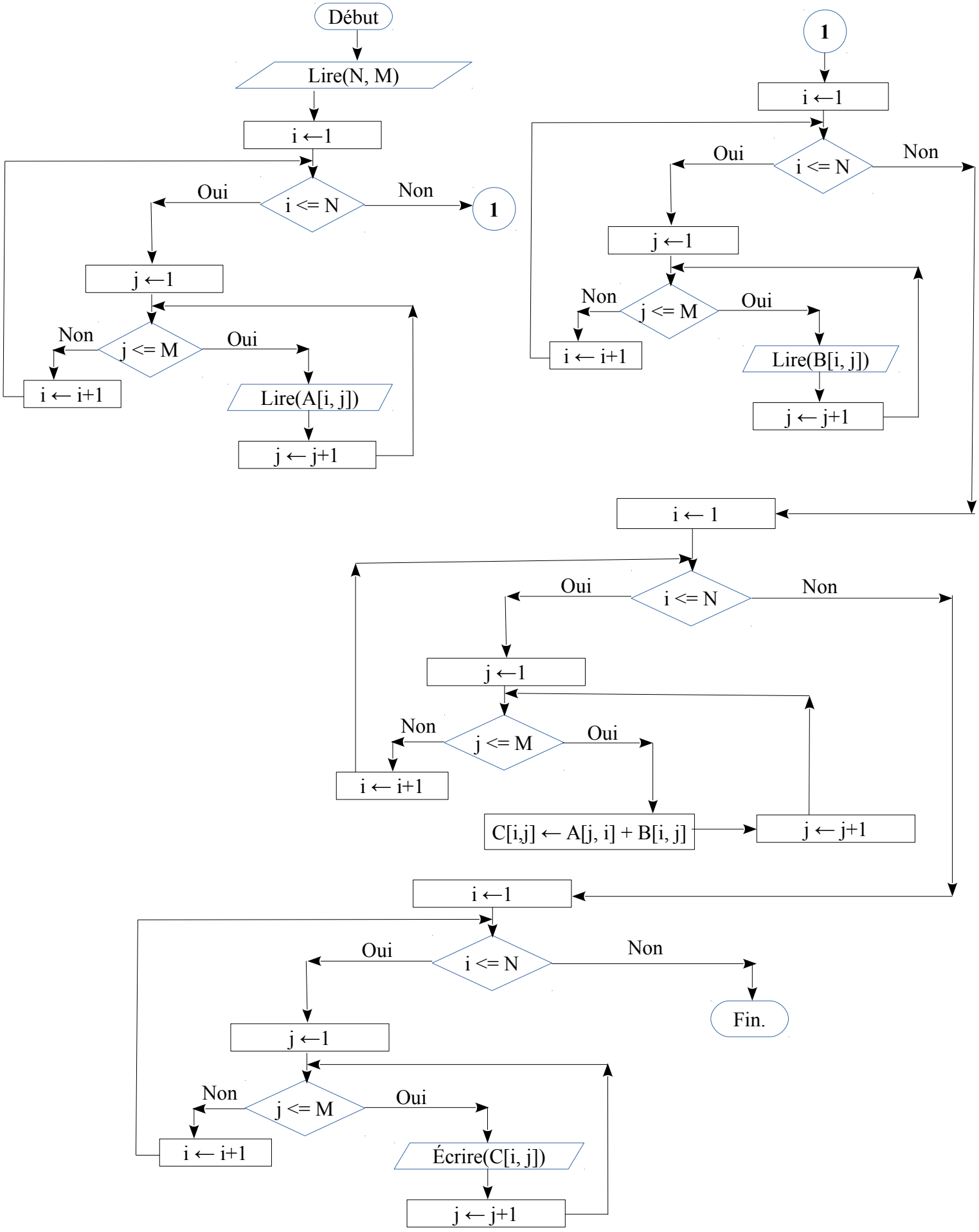
L'algorithme



Explication

Pour réaliser la somme de deux matrices A et B, il faut que ces dernières soient de même dimensions (nombre de lignes et nombre de colonne). La matrice C résultat de la somme est aussi du même dimension que A et B. Chaque case C[i, j] est égale à A[i,j] + B[i, j] (pour i=1, …, N et j = 1… M).

Organigramme

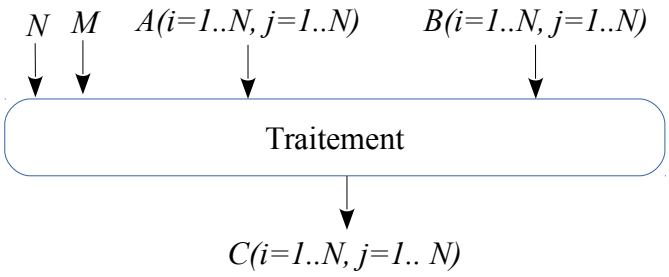


Exercice 10 : Produit de deux matrices

Soit A et B deux matrices carrées d'ordre N. Écrire un algorithme/programme PASCAL qui permet de calculer le produit de A et B.

Solution

L'algorithme



Explication

Pour réaliser le produit de deux matrices A et B, il faut que le nombre de colonnes de A soit égale au nombre de ligne de B. La matrice C résultat aura le même nombre de lignes de A et le même nombre de colonnes de B. Ainsi, on écrit : A(NxM) x B(MxL) = C(NxL) (A est une matrice de N lignes et M colonnes, B est une matrice de M lignes et L colonnes, donc C elle aura N lignes et L colonnes).

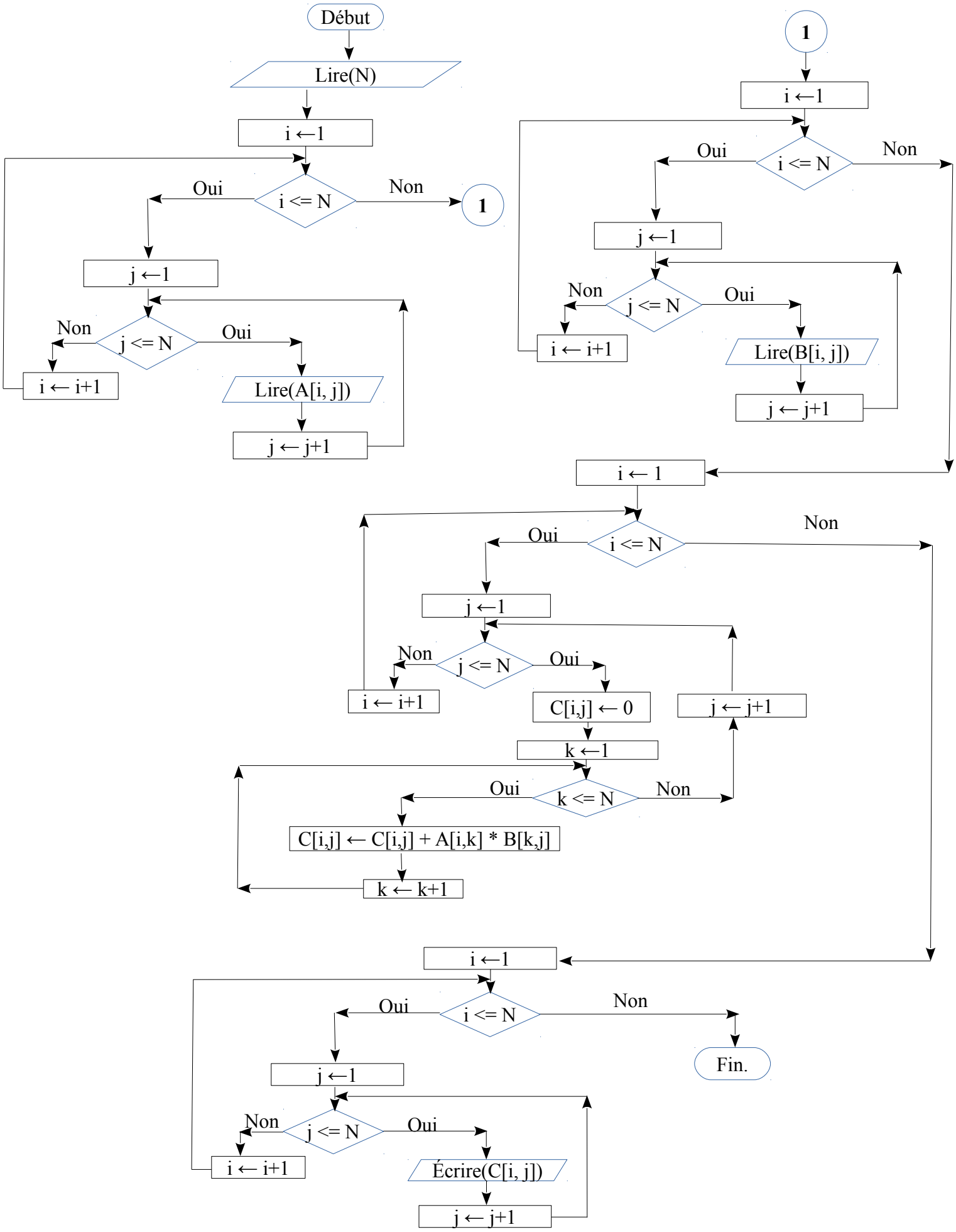
Chaque case de C sera calculée en utilisant la formule suivante :

C[i, j] = A[i, 1]*B[1, j] + A[i, 2]*B[2, j] + A[i, 3]*B[3, j] + … + A[i, M]*B[M, j]

Pour chaque i=1...N et j=1...L

Donc, C[i, j] = $\sum_{k=1}^{M} A[i,k] \times B[k,j]$

Dans le cas de matrices carrée d'ordre N (comme le cas de cet exercice), la formule devient comme suit : C[i, j] = $\sum_{k=1}^{N} A[i,k] \times B[k,j]$


Début
Lire(N)
i ←1
i <= N
Oui
Non
1
j ←1
j <= N
Non
Oui
i ← i+1
Lire(A[i, j])
j ← j+1
1
i ←1
i <= N
Oui
Non
j ←1
j <= N
Non
Oui
i ← i+1
Lire(B[i, j])
j ← j+1
i ← 1
i <= N
Oui
Non
j ←1
j <= N
Non
Oui
i ← i+1
C[i,j] ← 0
j ← j+1
k ←1
k <= N
Oui
Non
C[i,j] ← C[i,j] + A[i,k] * B[k,j]
k ← k+1
i ←1
i <= N
Oui
Non
Fin.
j ←1
j <= N
Non
Oui
i ← i+1
Écrire(C[i, j])
j ← j+1