TP - Programmation

Corrigé de la Série de TP N°2 – Tableaux à deux dimensions - Matrices

Exercice N°01 : Algorithme → Programme C

Soit l'algorithme suivant :

```
Algorithme Matrice ;
Variables
    A : Tableau [0..99, 0..99] de réel;
    i, j, N : entier;
    S, M : réel;
Début
    // Entrées
    Ecrire("Donner la taille de la matrice carrée A : ");
    Lire(N);
    Ecrire("Donner les composantes de la matrice A:");
    Pour i ←0 à N-1 faire
        Pour j ←0 à N-1 faire
            Lire(A[i,j]);
        FinPour;
    FinPour;
    // Traitement
    S ← 0;
    Pour i ← 0 à N-1 faire
        S ← S+A[i,i] ;
    FinPour;
    M ← S/N ;
    // Sorties
    Ecrire("S=", S, "M=", M);
Fin.
```

Questions :

1- Traduire l'algorithme en Programme C.

2- Compiler et exécuter le programme pour :
N = 3 et

$$A = \begin{bmatrix} 2 & 4 & -4 \\ 4 & 8 & 1 \\ 3.5 & 9 & 4 \end{bmatrix}$$

3- Dérouler le programme pour les valeurs de N et A ci-dessus ?

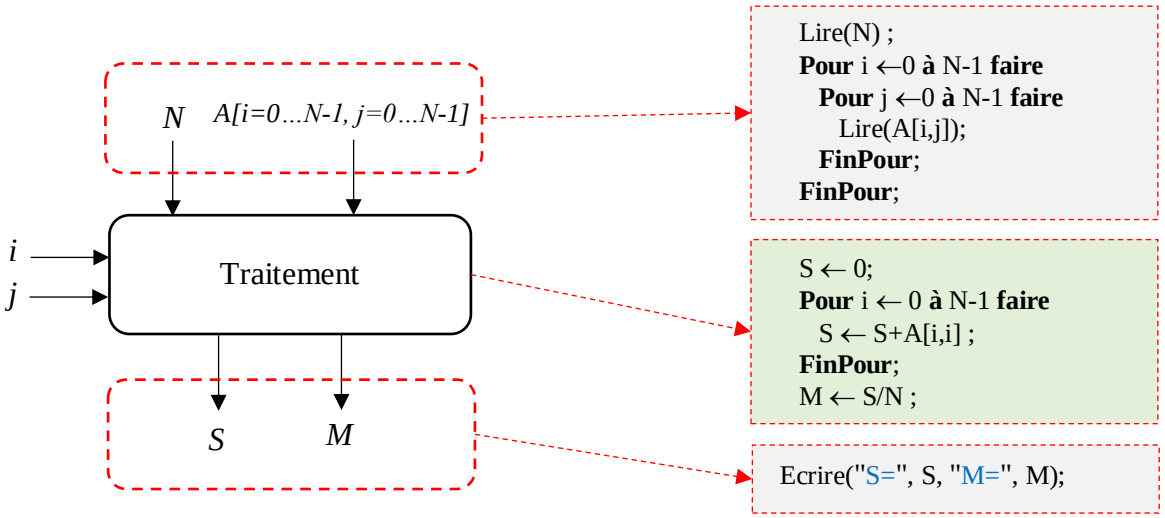
4- Déduire ce que fait le programme ?

5- Ré-écrire le programme en remplaçant la boucle *Pour* par la boucle *Tantque* **dans la partie des entrées**.

6- Ré-écrire le programme en remplaçant la boucle *Pour* par la boucle *Répéter* **dans la partie des entrées**.

Solution :

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



Remarque :

Les variables ***i*** et ***j*** sont des variables de traitement ou intermédiaires, utilisées pour parcourir la matrice A.

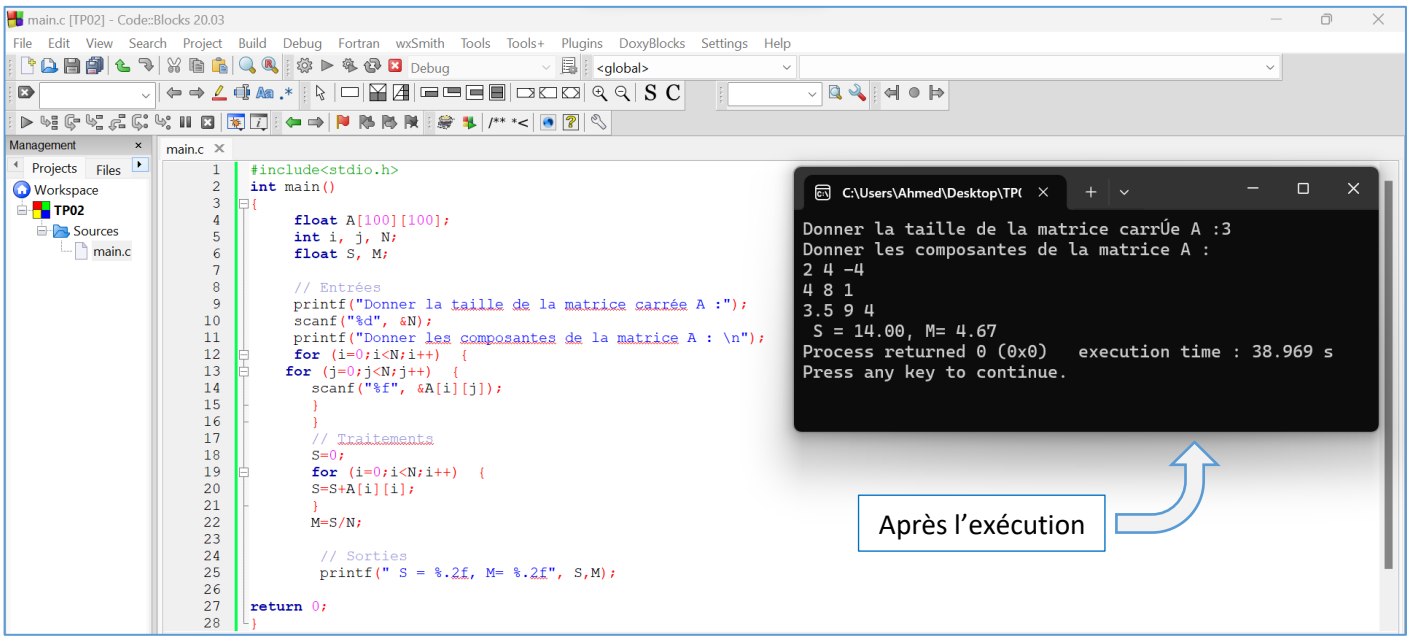
1 - Algorithme/programme C :

Algorithme	Programme C

Algorithme Matrice ;
Variables
A : Tableau [0..99, 0..99] **de réel**;
i, j, N : **entier**;
S, M : **réel**;
Début
// Entrées
Ecrire("Donner la taille de la matrice carrée A : ");
Lire(N);
Ecrire("Donner les composantes de la matrice A:");
Pour i ←0 **à** N-1 **faire**
Pour j ←0 **à** N-1 **faire**
Lire(A[i,j]);
FinPour;
FinPour;
// Traitements
S ← 0;
Pour i ← 0 **à** N-1 **faire**
S ← S+A[i,i] ;
FinPour;
M ← S/N ;
// Sorties
Ecrire("S= ", S, "M=", M);
Fin.
	#include<stdio.h>
	int main()
	{
	float A[100][100];
	int i, j, N;
	float S, M;
	// Entrées
	printf("Donner la taille de la matrice carrée A :");
	scanf("%d", &N);
	printf("Donner les composantes de la matrice A : \n");
	for (i=0;i<N;i++) {
	for (j=0;j<N;j++) {
	scanf("%f", &A[i][j]);
	}
	}
	// Traitements
	S=0;
	for (i=0;i<N;i++) {
	S=S+A[i][i];
	}
	M=S/N;
	// Sorties
	printf(" S = %.2f, M= %.2f", S,M);
	return 0;
	}

2 - Compiler et exécuter le programme pour : N = 3 et

$$A = \begin{bmatrix} 2 & 4 & -4 \\ 4 & 8 & 1 \\ 3.5 & 9 & 4 \end{bmatrix}$$



3 - Dérouler le programme pour les valeurs de N et A ci-dessus ?

Instructions	Variables						Affichage
	N	i	j	A	S	M	
printf("Donner la dimension de la matrice A :");	/	/	/	/	/	/	Donner la dimension de la matrice A :
scanf("%d", &N);	3	/	/	/	/	/	
printf("Donner les composantes de la matrice A : \n");	3	/	/	/	/	/	Donner les composantes de la matrice A :

for(i=0;i<N;i++){
for (j=0;j<N;j++){
scanf("%f", &A[i][j]);
}}
	3	0
		1
		2
		3
			1
			2
			3
				j=0 j=1 j=2
				i=0 2 4 -4
				i=1 4 8 1
				i=2 3.5 9 4
					/	/	//

S=0;	3	/	/	//	0	/	//
For i=0 S=S+A[i,i]; S=0+A[0,0]; S=0+2=2;	3	0		//	2	/	//
For i=1 S=S+A[i,i]; S=2+A[1,1]; S=2+8=10;	3	1	/	//	10	/	//
For i=2 S=S+A[i,i]; S=10+A[2,2]; S=10+4=14;	3	2	/	//	14	/	//
M=S/N; M=14/3=4.67;	3			//	14	4.67	//
printf(" S = %.2f, M= %.2f", S,M);	3			//	14	4.67	**S=14.00 , M=4.67**

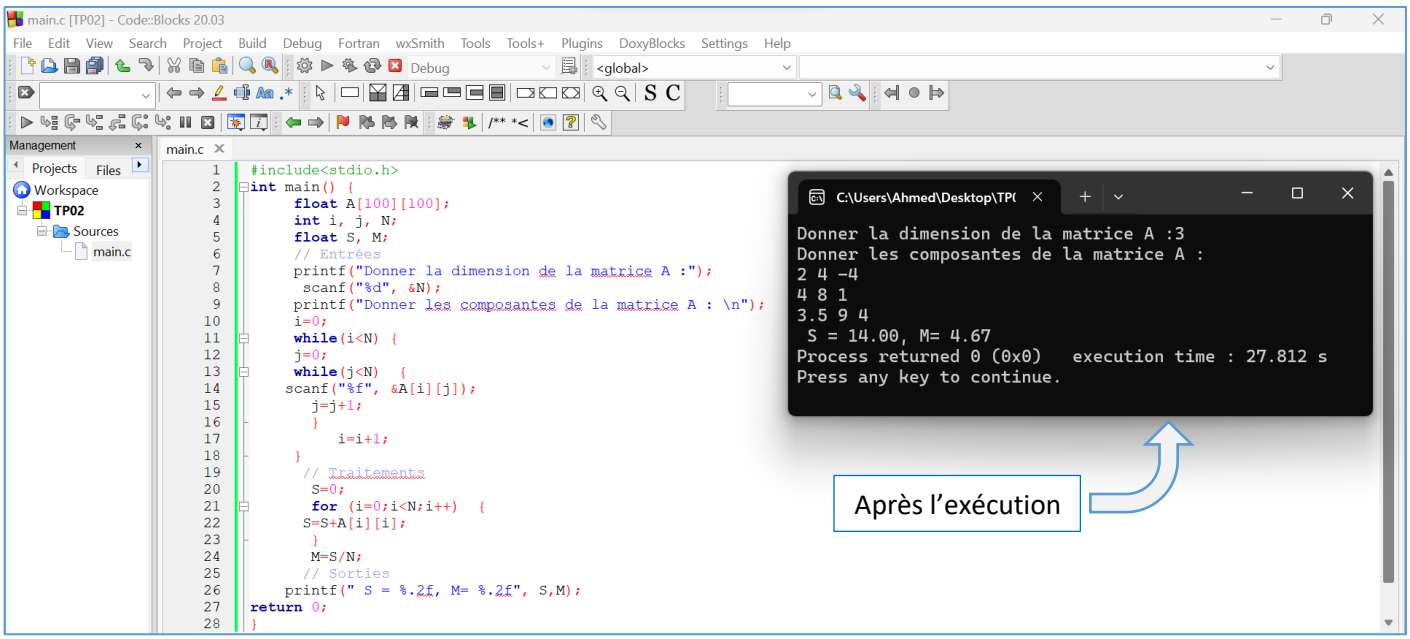
4 - Déduire ce que fait le programme ?

Le programme calcul la somme des éléments de la diagonale de la matrice A et leur moyenne.

5 - Ré-écrire le programme en remplaçant la boucle *Pour* par la boucle *Tantque* dans la partie des entrées.

Programme C (avec la boucle For)	Programme C (avec la boucle While)

#include<stdio.h>
int main() {
float A[100][100];
int i, j, N;
float S, M;
// Entrées
printf("Donner la dimension de la matrice A :");
scanf("%d", &N);
printf("Donner les composantes de la matrice A : \n");
for (i=0;i<N;i++) {
for (j=0;j<N;j++) {
scanf("%f", &A[i][j]);
}
}
// Traitements
S=0;
for (i=0;i<N;i++)
{
S=S+A[i][i];
}
M=S/N;
// Sorties
printf(" S = %.2f, M= %.2f", S,M);
return 0;
}
	#include<stdio.h>
	int main() {
	float A[100][100];
	int i, j, N;
	float S, M;
	// Entrées
	printf("Donner la dimension de la matrice A :");
	scanf("%d", &N);
	printf("Donner les composantes de la matrice A : \n");
	i=0;
	while(i<N) {
	j=0;
	while(j<N) {
	scanf("%f", &A[i][j]);
	j=j+1;
	}
	i=i+1;
	}
	// Traitements
	S=0;
	for (i=0;i<N;i++) {
	S=S+A[i][i];
	}
	M=S/N;
	// Sorties
	printf(" S = %.2f, M= %.2f", S,M);
	return 0;
	}

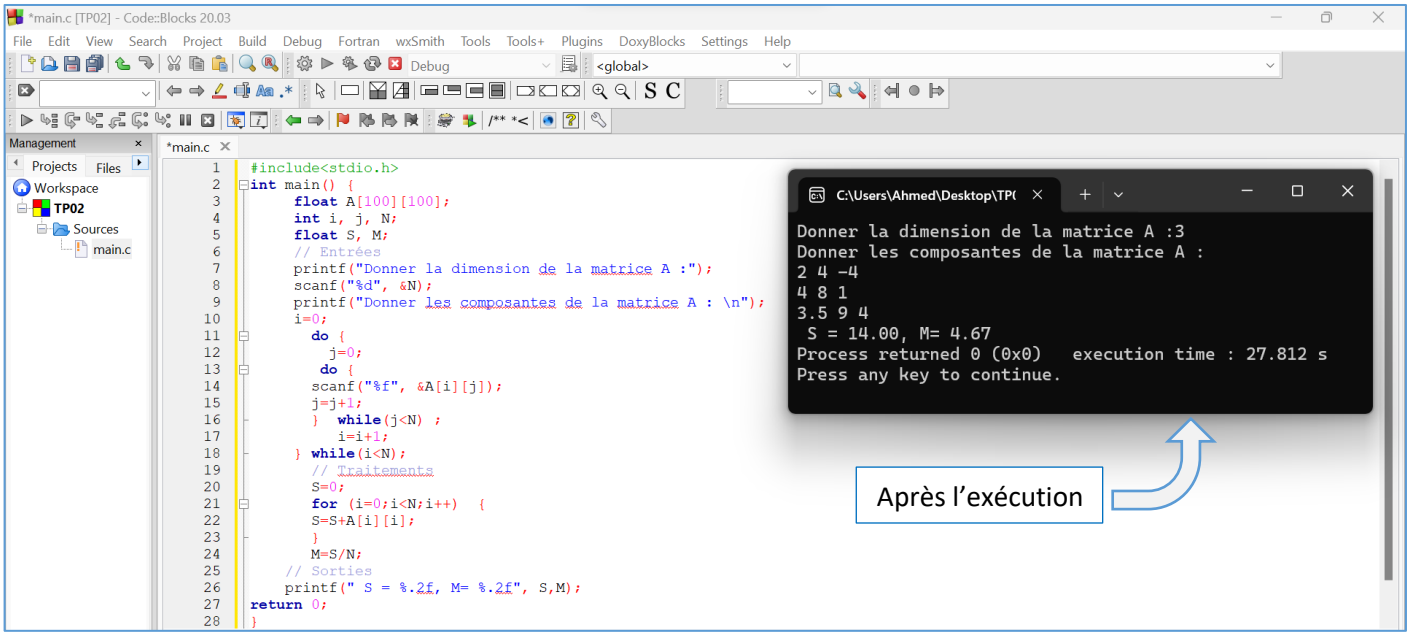


6 - Ré-écrire le programme en remplaçant la boucle *Pour* par la boucle *Répéter* dans la partie des entrées.

Programme C (avec la boucle For)	Programme C (avec la boucle Repeat)

```c
#include<stdio.h>
int main()
{
    float A[100][100];
    int i, j, N;
    float S, M;

    // Entrées
    printf("Donner la dimension de la matrice A :");
     scanf("%d", &N);
    printf("Donner les composantes de la matrice A : \n");
    for (i=0;i<N;i++)  {
        for (j=0;j<N;j++)  {
            scanf("%f", &A[i][j]);
     }
     }

      // Traitements
     S=0;
     for (i=0;i<N;i++)
           {
           S=S+A[i][i];
           }
       M=S/N;

  // Sorties
  printf(" S = %.2f, M= %.2f", S,M);

return 0;
}
```

```c
#include<stdio.h>
int main() {
    float A[100][100];
    int i, j, N;
    float S, M;

    // Entrées
    printf("Donner la dimension de la matrice A :");
    scanf("%d", &N);
    printf("Donner les composantes de la matrice A : \n");
    i=0;
          do {
           j=0;
          do {
          scanf("%f", &A[i][j]);
          j=j+1;
     } while(j<N) ;
        i=i+1;
    } while(i<N);

     // Traitements
     S=0;
     for (i=0;i<N;i++)  {
          S=S+A[i][i];
          }
          M=S/N;

   // Sorties
   printf(" S = %.2f, M= %.2f", S,M);
return 0;
}
```

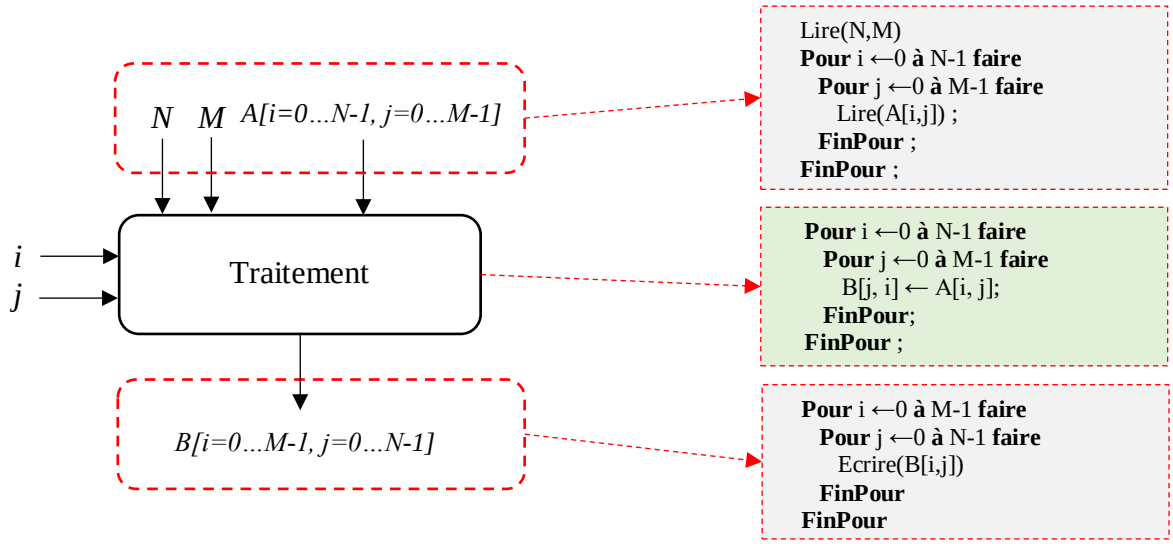


Exercice N°02 : Transposée d'une matrice

Ecrire un algorithme/programme C qui permet de calculer la matrice B transposée d'une matrice réelle A d'ordre N x M.

Solution :

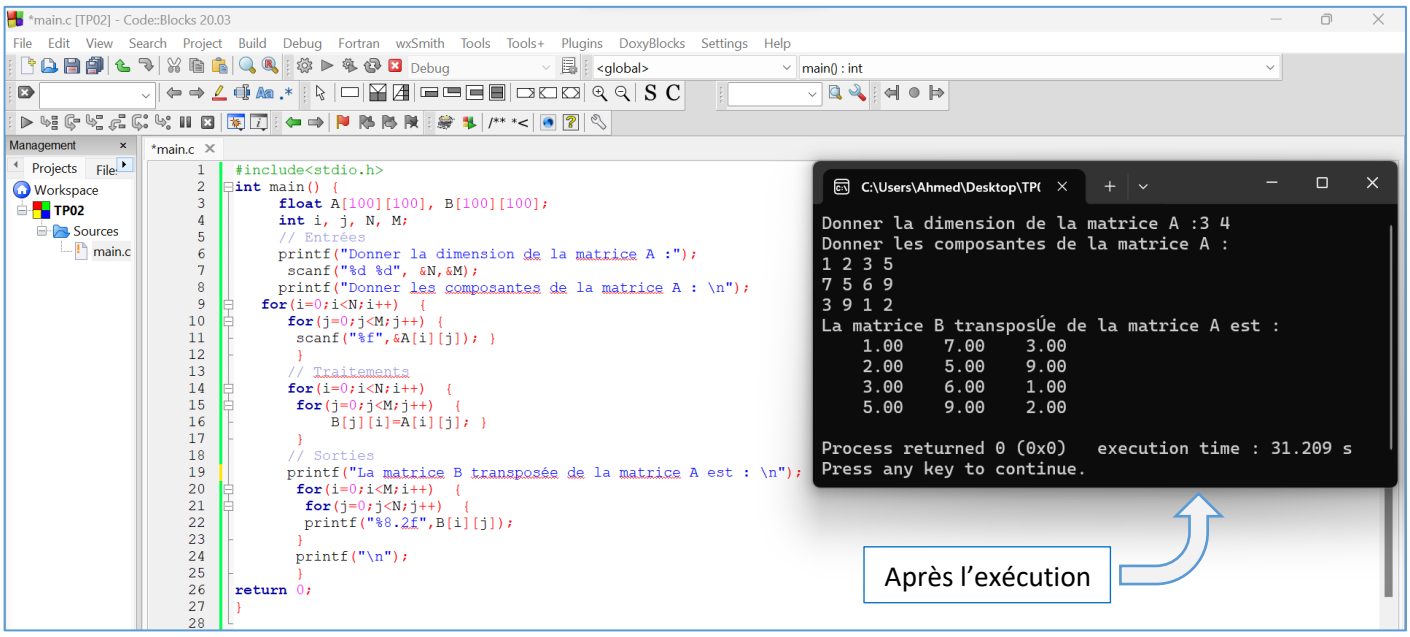
Le transposé d'une matrice A d'ordre N x M est un matrice B d'ordre M x N.
Chaque ligne de A devient une colonne de B (ou chaque colonne de A devient une ligne pour B). Chaque case B[i, j] correspond à la case A[j, i] tel que : i=0, …, M-1 et j=0, …, N-1.
Les variables d'entrée, variable de sorite et la partie traitement sont présentées dans le schéma ci-dessous :



Algorithme/programme C :

Algorithme	Programme C

Algorithme Transposee ;
Variables
A, B : Tableau [0..99, 0..99] **de réel** ;
i, j, N, M : **Entier** ;
Début
// Entrées
Ecrire("Donner le nombre des lignes et des colonnes de A:") ;
Lire(N,M) ;
Ecrire("Donner les composantes de la matrice A : ") ;
Pour i←0 **à** N-1 **faire**
Pour j←0 **à** M-1 **faire**
Lire(A[i,j]) ;
FinPour ;
FinPour ;
// Traitements
Pour i←0 **à** N-1 **faire**
Pour j←0 **à** M-1 **faire**
B[j, i]← A[i, j] ;
FinPour ;
FinPour ;
// Sorties
Ecrire("La matrice B Transposée de A est : ") ;
Pour i←1 **à** M **faire**
Pour j←1 **à** N **faire**
Ecrire(B[i,j]) ;
FinPour ;
FinPour ;
Fin.
	#include<stdio.h>
	int main() {
	float A[100][100], B[100][100];
	int i, j, N, M;
	// Entrées
	printf("Donner la dimension de la matrice A :");
	scanf("%d %d", &N,&M);
	printf("Donner les composantes de la matrice A : \n");
	for(i=0;i<N;i++) {
	for(j=0;j<M;j++) {
	scanf("%f",&A[i][j]);
	}
	}
	// Traitements
	for(i=0;i<N;i++) {
	for(j=0;j<M;j++) {
	B[j][i]=A[i][j];
	}
	}
	// Sorties
	printf("La matrice B transposée de la matrice A est : \n");
	for(i=0;i<M;i++) {
	for(j=0;j<N;j++) {
	printf("%8.2f",B[i][j]);
	}
	printf("\n");
	}
	return 0;
	}
	Une autre méthode :
	for(i=0;i<M;i++)
	for(j=0;j<N;j++)
	B[i][j]=A[j][i];



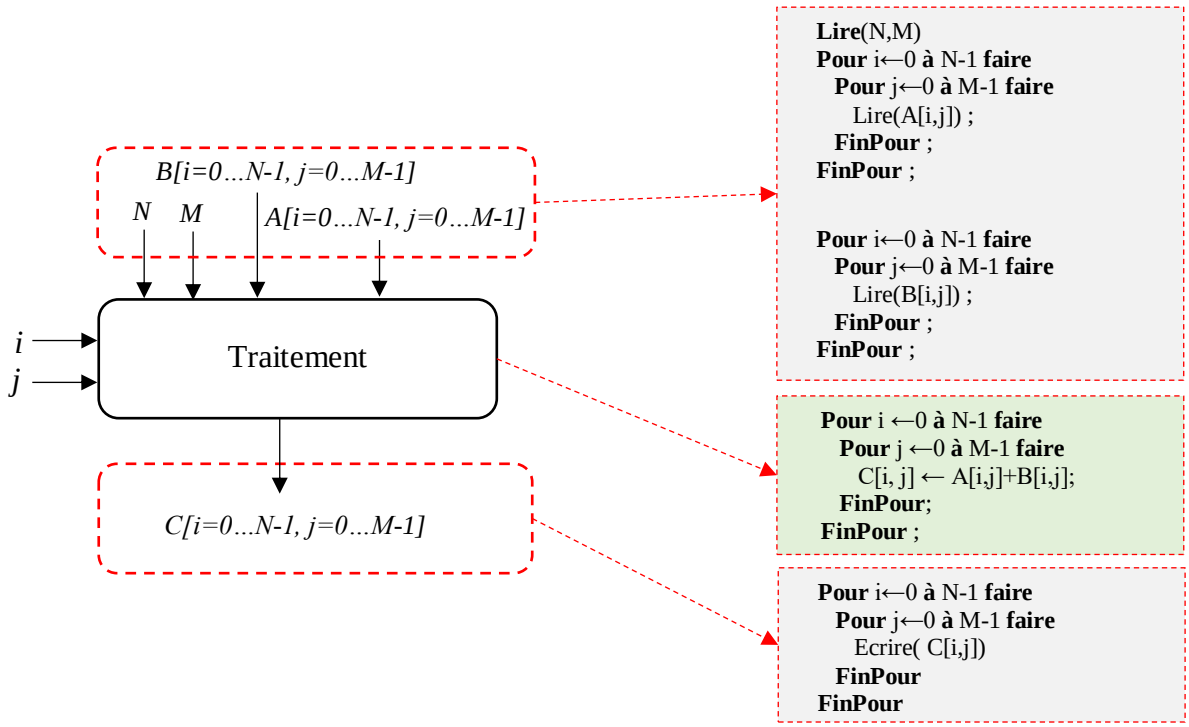
Exercice N°03 : Somme de deux matrices

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

Solution :

Pour pouvoir réaliser la somme de deux matrices réelles, A et B, une condition nécessaire doit être vérifiée : A et B doivent être de même taille. Ainsi, si A est d'ordre NxM, alors B est aussi d'ordre NxM. Par conséquent, la matrice C, la somme des deux matrices, est aussi d'ordre NxM.

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



Algorithme/programme C :

Algorithme	Programme C

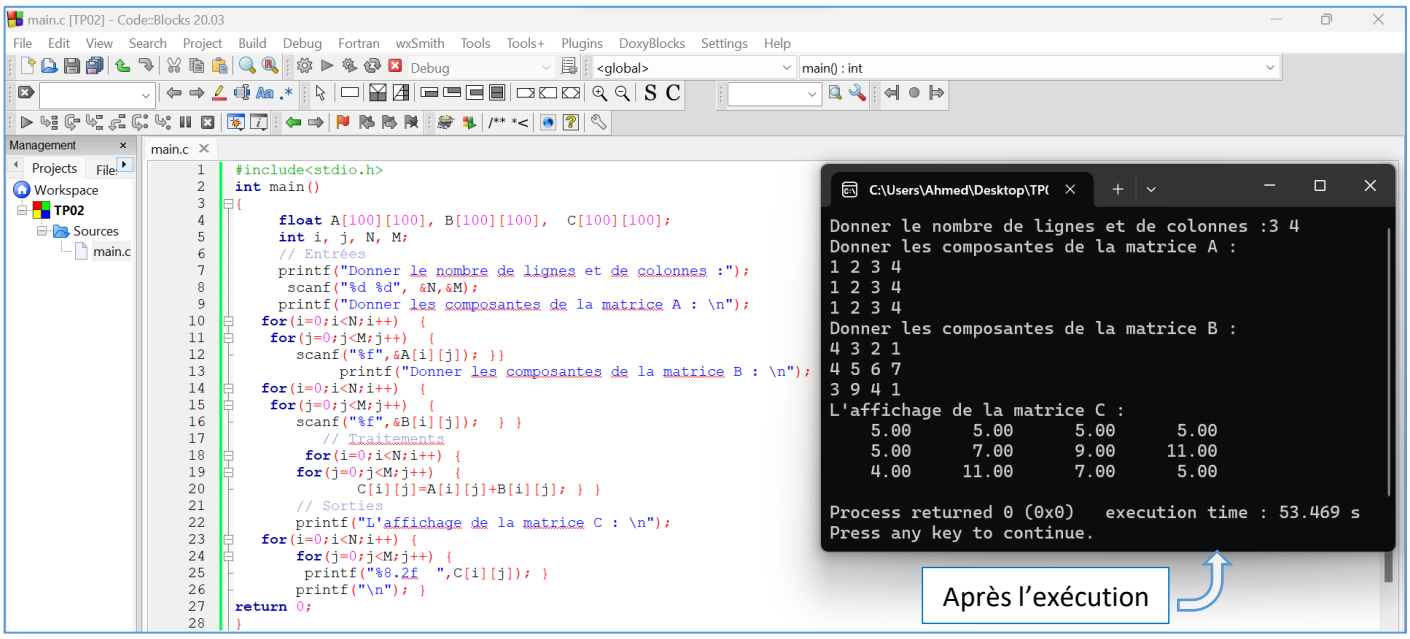
Algorithme Somme_deux_matrices ;
Variables
A, B, C : Tableau [0..99, 0..99] **de réel** ;
i, j, N, M : **Entier** ;
Début
// Entrées
Ecrire("Donner le nombre des lignes et des colonnes : ") ;
Lire(N,M) ;
Ecrire("Donner les composantes de la matrice A : ") ;
Pour i←0 **à** N-1 **faire**
Pour j←0 **à** M-1 **faire**
Lire(A[i,j]) ;
FinPour ;
FinPour ;

Ecrire("Donner les composantes de la matrice B : ") ;
Pour i←0 **à** N-1 **faire**
Pour j←0 **à** M-1 **faire**
Lire(B[i,j]) ;
FinPour ;
FinPour ;

// Traitements
Pour i←0 **à** N-1 **faire**
Pour j←0 **à** M-1 **faire**
C[i, j]← A[i,j]+B[i,j] ;
FinPour ;
FinPour ;

// Sorties
Ecrire(" L'affichage de la matrice C : ") ;
Pour i←0 **à** N-1 **faire**
Pour j←0 **à** M-1 **faire**
Ecrire(C[i,j]) ;
FinPour ;
FinPour ;
Fin.

```c
#include<stdio.h>
int main()
{
   float A[100][100], B[100][100],  C[100][100];
   int i, j, N, M;
   // Entrées
   printf("Donner le nombre de lignes et de colonnes :");
    scanf("%d %d", &N,&M);
   printf("Donner les composantes de la matrice A : \n");
  for(i=0;i<N;i++)  {
   for(j=0;j<M;j++)  {
          scanf("%f",&A[i][j]);
    }
    }
        printf("Donner les composantes de la matrice B : \n");
  for(i=0;i<N;i++)  {
   for(j=0;j<M;j++)  {
          scanf("%f",&B[i][j]);
    }
    }

      // Traitements
       for(i=0;i<N;i++) {
         for(j=0;j<M;j++)  {
           C[i][j]=A[i][j]+B[i][j];
    }
    }

    // Sorties
    printf("L'affichage de la matrice C : \n");
  for(i=0;i<N;i++) {
    for(j=0;j<M;j++) {
       printf("%8.2f  ",C[i][j]);
    }
    printf("\n");
    }
return 0;
}
```



Exercice N°04 : Matrice symétrique

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

Ecrire un programme C qui permet de vérifier si la matrice A est symétrique.

Rappel : Une matrice A est symétrique si A[*i*, *j*] = A[*j*, *i*] pour tout *i* et *j*.

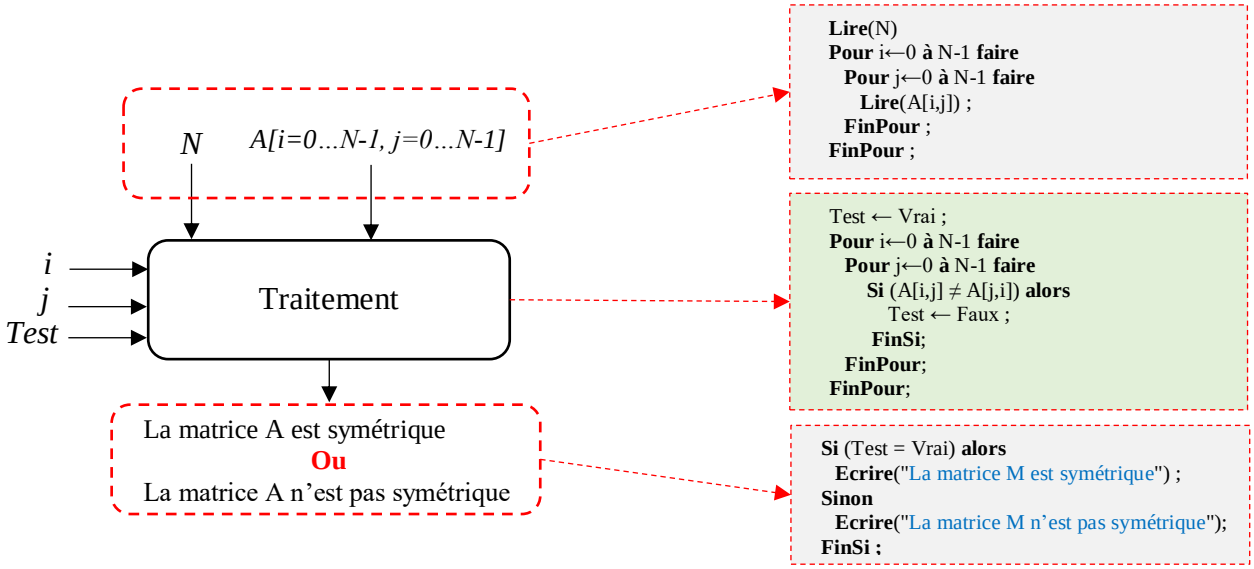
Solution :

Rappel : A est symétrique si A[*i*, *j*] = A[*j*, *i*] pour tout *i* et *j*.

Les étapes à suivre :

- D'abord supposer que A est symétrique (Test = True)
- Ensuite, comparer chaque case A[*j*, *i*] avec la case A[*j*, *i*].
- Si elles sont différentes alors affecter la valeur **False** à la variable Test.
- A la fin, il suffit de voir la valeur de Test pour savoir si la matrice A est symétrique ou non.

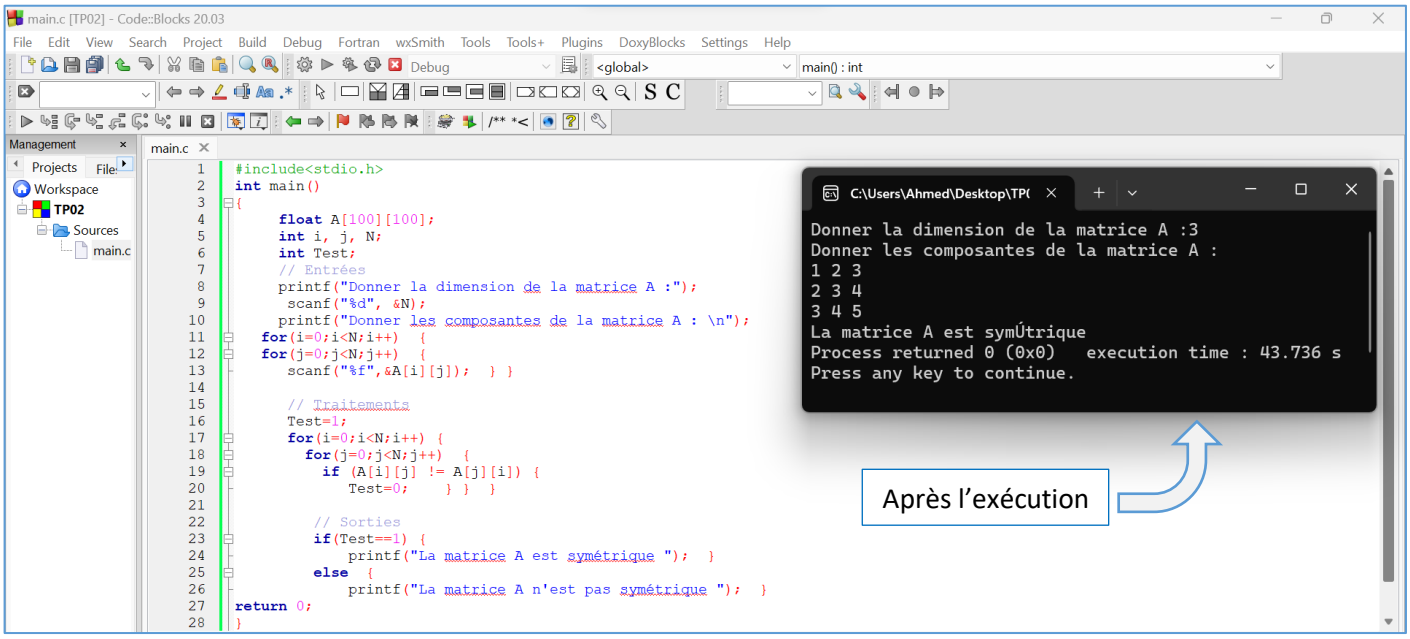
Les variables d'entrée, variable de sorite et la partie traitement sont présentées dans le schéma ci-dessous :



Algorithme/programme C :

Algorithme	Programme C

Algorithme Matrice_Symetrique;
Variables
A : **Tableau** [0..9,0..9] **de réel**;
N, i, j : **entier** ;
Test : **booléen** ;
 	
Début
// Entrées
Ecrire("Donner la dimension de la matrice carrée A : ");
Lire(N) ;
Ecrire("Donner les composantes de la matrice A : ");
Pour 0←1 **à** N-1 **faire**
Pour j←0 **à** N-1 **faire**
Lire(A[i, j]) ;
FinPour;
FinPour;
 	
// Traitements
Test ← Vrai ;
Pour i←0 **à** N-1 **faire**
Pour j←0 **à** N-1 **faire**
Si (A[i,j] ≠ A[j,i]) **alors**
Test ← Faux ;
FinSi;
FinPour;
FinPour;
 	
// Sorties
Si (Test = Vrai) **alors**
Ecrire("La matrice A est symétrique")
Sinon
Ecrire("La matrice A n'est pas symétrique") ;
FinSi ;
Fin.
	#include<stdio.h>
	int main()
	{
	float A[100][100];
	int i, j, N;
	int Test;
	
	// Entrées
	printf("Donner la dimension de la matrice A :");
	scanf("%d", &N);
	printf("Donner les composantes de la matrice A : \n");
	for(i=0;i<N;i++) {
	for(j=0;j<N;j++) {
	scanf("%f",&A[i][j]); }
	}
	
	// Traitements
	Test=1;
	for(i=0;i<N;i++) {
	for(j=0;j<N;j++) {
	if (A[i][j] != A[j][i]) {
	Test=0;
	}
	}
	}
	
	// Sorties
	if(Test==1) {
	printf("La matrice A est symétrique "); }
	else {
	printf("La matrice A n'est pas symétrique "); }
	return 0;
	}



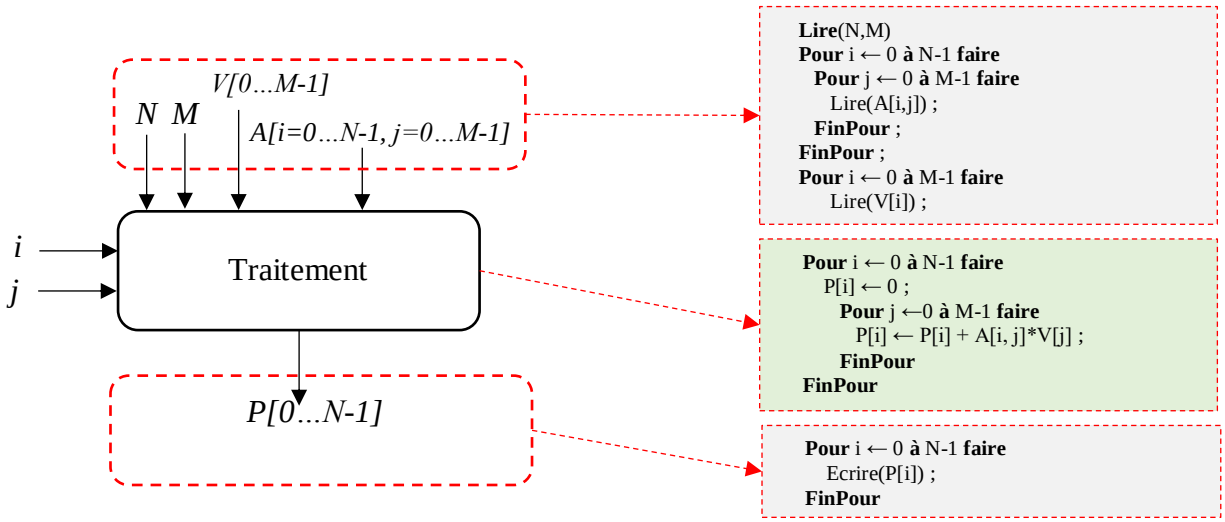
Exercice N°05 : Produit d'une matrice par un vecteur

Soit A une matrice de type réel et d'ordre N x M.

Ecrire un algorithme/programme C qui permet de calculer le produit de la matrice A par un vecteur V de type réel et de taille M.

Solution :

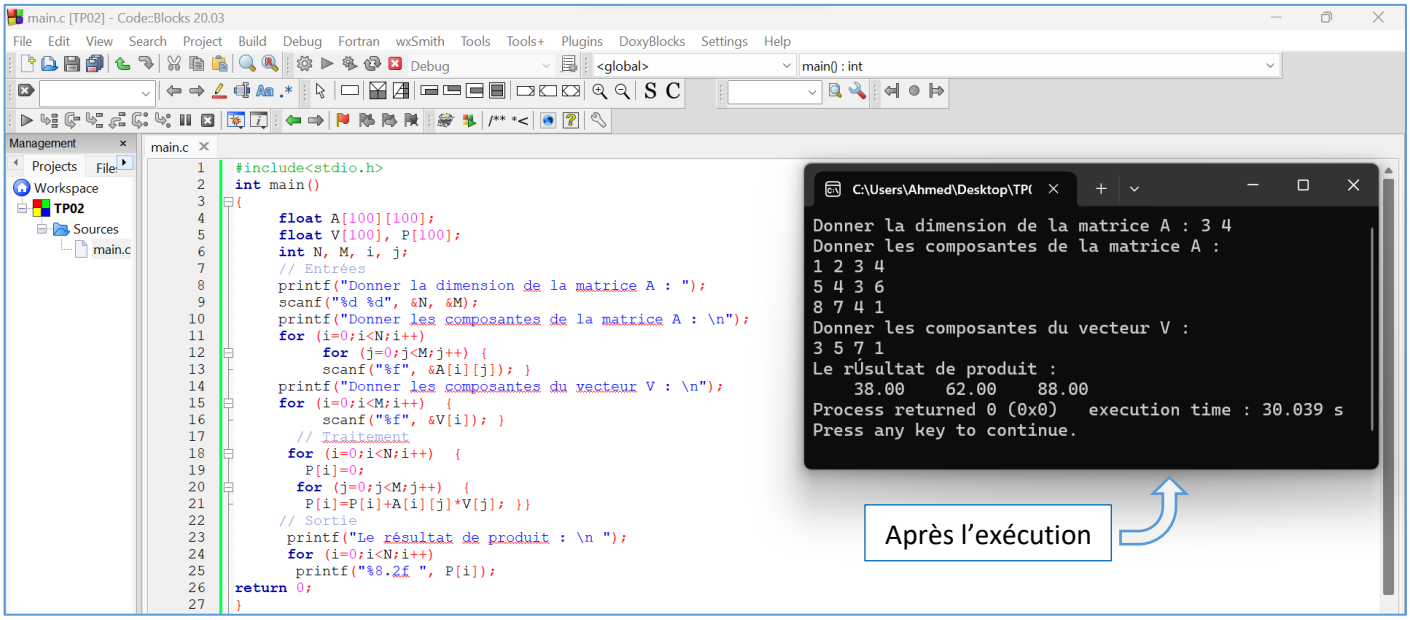
Les variables d'entrée, variable de sorite et la partie traitement sont présentées dans le schéma ci-dessous :



Algorithme/programme C :

Algorithme	Programme C

Algorithme Produit_Matrice_Vecteur ;
Variables
A : Tableau [0..99, 0..99] **de réel** ;
V, P : Tableau [0..99] **de réel** ;
N, M, i, j : **entier** ;
 	
Début
// Entrées
Ecrire("Donner la dimension de la matrice A : ");
Lire (N, M);
Ecrire("Donner les composantes de la matrice A : ");
pour i ←0 **à** N-1 **faire**
Pour j ←0 **à** M-1 **faire**
Lire(A[i, j]) ;
FinPour ;
FinPour ;
Ecrire("Donner les composantes du vecteur V : ");
pour i ← 0 **à** M-1 **faire**
Lire(V[i]) ;
FinPour ;
 	
// Traitement
pour i ← 0 **à** N-1 **faire**
P[i] ← 0 ;
Pour j=0 **à** M-1 **faire**
P[i] ← P[i] + A[i, j]*V[j] ;
FinPour;
FinPour ;
 	
// Sortie
Ecrire("Le résultat de produit : ") ;
pour i ← 0 **à** N-1 **faire**
Ecrire(P[i]) ;
FinPour
Fin.
	#include<stdio.h>
	int main()
	{
	float A[100][100];
	float V[100], P[100];
	int N, M, i, j;
	
	// Entrées
	printf("Donner la dimension de la matrice A : ");
	scanf("%d %d", &N, &M);
	printf("Donner les composantes de la matrice A : \n");
	for (i=0;i<N;i++)
	for (j=0;j<M;j++)
	{
	scanf("%f", &A[i][j]);
	}
	printf("Donner les composantes du vecteur V : \n");
	for (i=0;i<M;i++) {
	scanf("%f", &V[i]);
	}
	
	// Traitement
	for (i=0;i<N;i++) {
	P[i]=0;
	for (j=0;j<M;j++) {
	P[i]=P[i]+A[i][j]*V[j];
	}
	}
	
	// Sortie
	printf("Le résultat de produit : \n ");
	for (i=0;i<N;i++)
	printf("%8.2f ", P[i]);
	return 0;
	}



Il est crucial de souligner que dans le domaine de la programmation, diverses approches peuvent conduire aux mêmes résultats. Il est vivement recommandé de privilégier la simplicité lors du choix de la méthode, afin d'optimiser l'efficacité et la compréhension du code.