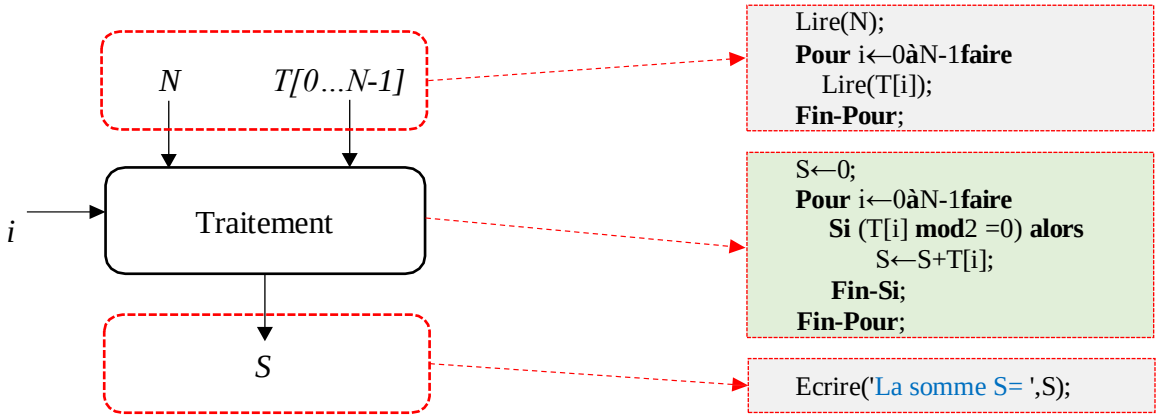
TP- Initiation à la Programmation

Corrigé de la Série de TP N°2–Tableaux à une dimension-Vecteurs

Solution de ExerciceN°01: *Algorithme → Programme C*

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



Remarque:

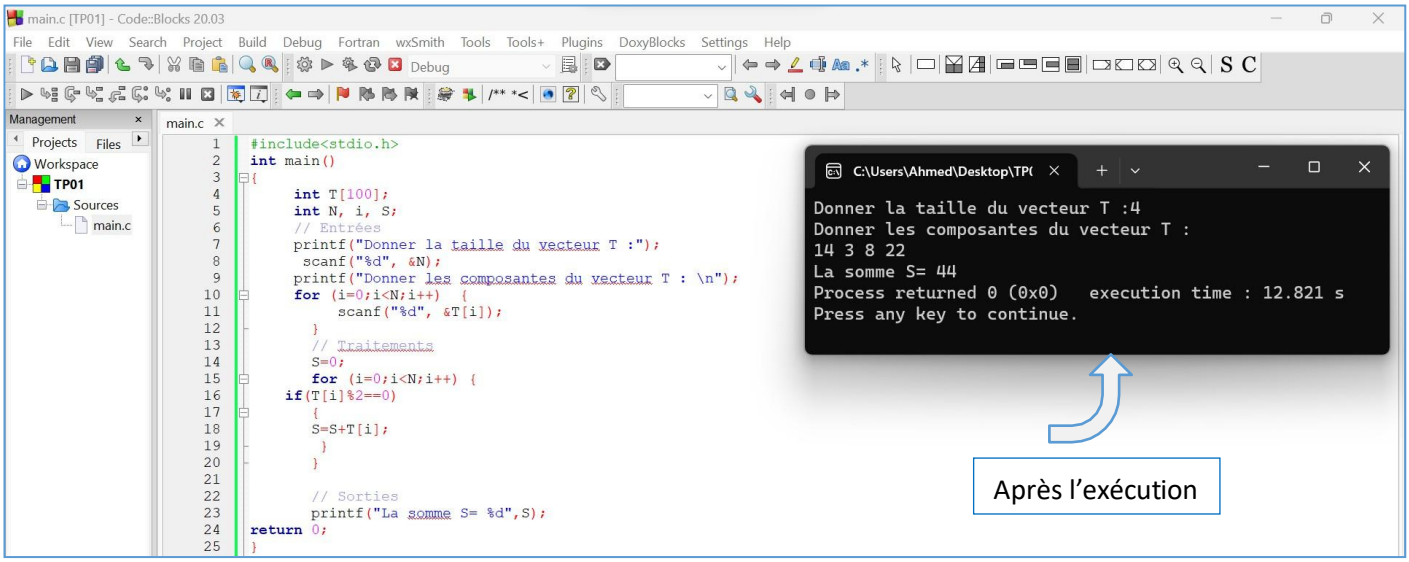
La variable ***i*** est une variable de traitement ou intermédiaire, utilisée pour parcourir le vecteur T.

1- Traduire l'algorithme en programme C

Algorithme	Programme C

Algorithme Vecteur;
Variables
T:Tableau[0..99] **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←0 **à** N-1 **faire**
Lire(T[i]);
FinPour;
// Traitements
S←0;
Pour i←0 **à** N-1 **faire**
Si(T[i] **mod**2 =0) **alors**
S←S+T[i];
FinSi;
FinPour;
// Sorties
Ecrire("La somme S=",S);
Fin.
	#include <stdio.h>
	int main( )
	{
	int T[100];
	int N,i,S;
	//Entrées
	printf("Donner la taille du vecteur T:");
	scanf("%d", &N);
	printf("Donner les composantes du vecteur T:\n");
	for (i=0;i<N;i++){
	scanf("%d",&T[i]);
	}
	//Traitements
	S=0;
	for (i=0;i<N;i++){
	if(T[i]%2==0)
	{
	S=S+T[i];
	}
	}
	//Sorties
	printf("La somme S=%d ",S);
	return 0;
	}

2- Compiler et exécuter le programme pour: N=4 et T= [14 3 8 22].



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).

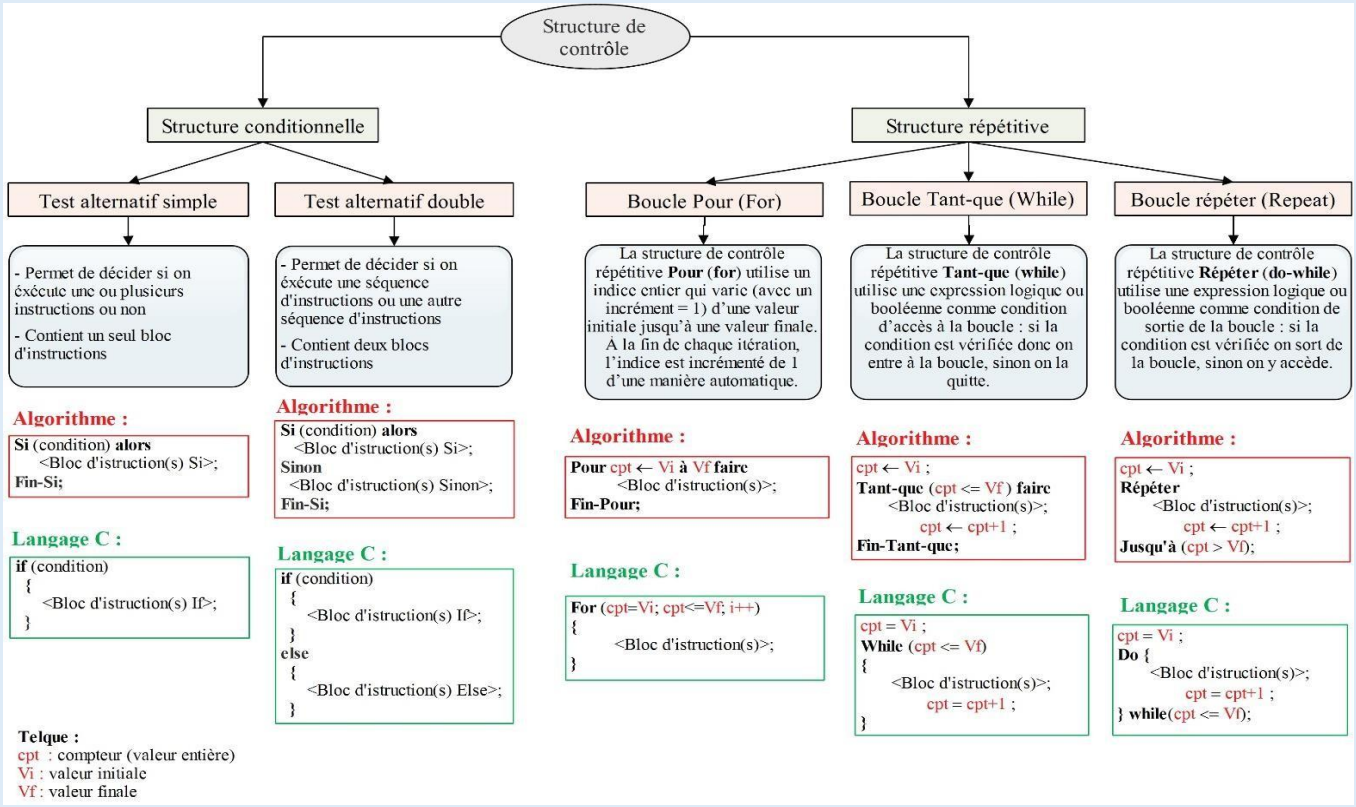
Instructions	Variables				Affichage
	N	i	T	S	
Ecrire("Donner la taille du vecteur T:");	/	/	/	/	Donner la taille du vecteur T:
Lire(N)	4	/	/	/	//
Ecrire("Donner les composantes du Vecteur T:");	4	/	/	/	Donner les composantes du Vecteur T:
Pour i←0 **à** N-1 **faire** Lire(T[i]) **Fin-Pour**	4	0 1 2 3	0 1 2 3 [14 3 8 22]	/	//
S←0	4	/	[14 3 8 22]	0	//
Pour i=0 **Si** T[0]mod2=0 **Alors** **14 mod 2 = 0 Vrai** S←S+T[i]=S+T[0] S ←0+14=14	4	0	[14 3 8 22]	14	//
Pour i=1 **Si**T[1]mod2=0**Alors** **3 mod 2 = 0 Faux**	4	1	[14 3 8 22]	//	//
Pour i=2 **Si**T[2]mod2=0 **Alors** **8 mod 2 = 0 Vrai** S←S+T[i]=S+T[2] S←14+8=22	4	2	[14 3 8 22]	22	//
Pour i=3 **Si**T[3] mod2=0 **Alors** **22 mod 2 = 0 Vrai** S←S+T[i]=S+T[3] S←22+22=44	4	3	[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)

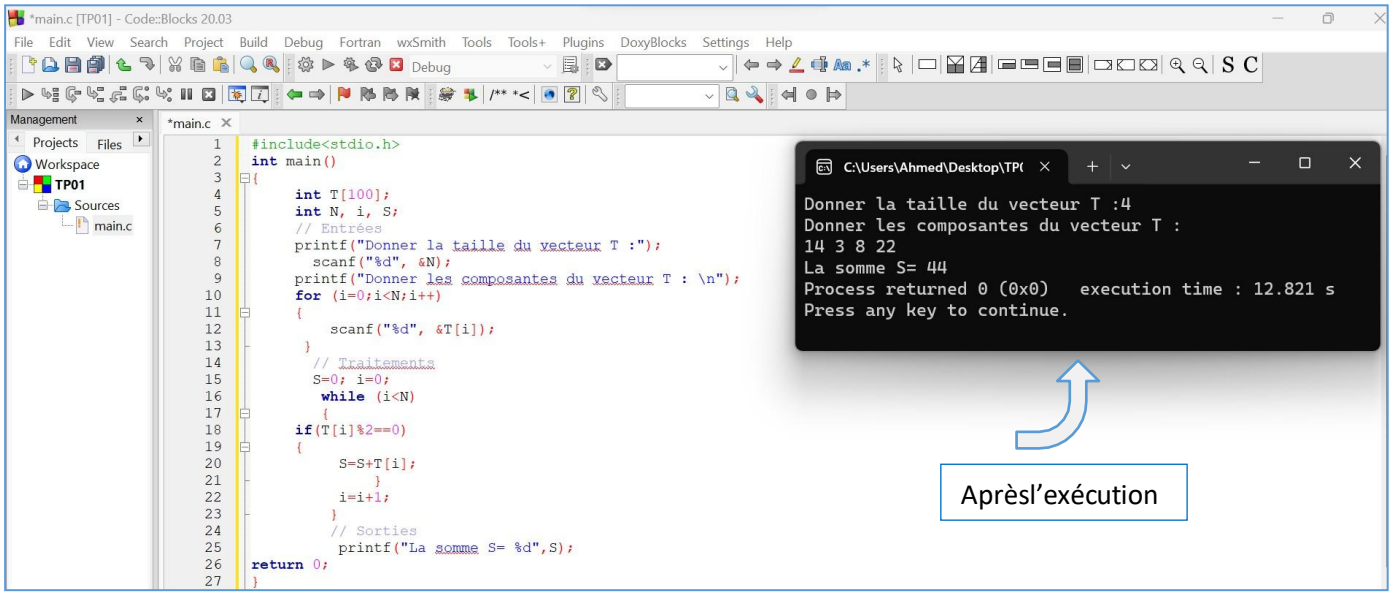
Rappel sur les structures de contrôle:



Algorithme/Programme C

Algorithme	Programme C

Algorithme Vecteur;
Variables
T:Tableau [0..99] **d'entier**;
N, i, S : **entier**;
Début
//Entrées
Ecrire("Donner la taille du vecteur T:");
Lire(N);
Ecrire("Donner les composantes du vecteur T:");
Pouri←0 **à** N-1 **faire**
Lire(T[i]);
FinPour;
// Traitements
S←0;
i←0;
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.
	#include <stdio.h>
	int main( )
	{
	int T[100];
	int N,i,S;
	//Entrées
	printf("Donner la taille du vecteur T:");
	scanf("%d", &N);
	printf("Donner les composantes du vecteur T : \n");
	for(i=0;i<N;i++)
	{
	scanf("%d",&T[i]);
	}
	//Traitements
	S=0; i=0;
	while(i<N)
	{
	if(T[i]%2==0)
	{
	S=S+T[i];
	}
	i=i+1;
	}
	//Sorties
	printf("La somme S=%d ",S);
	return 0;
	}

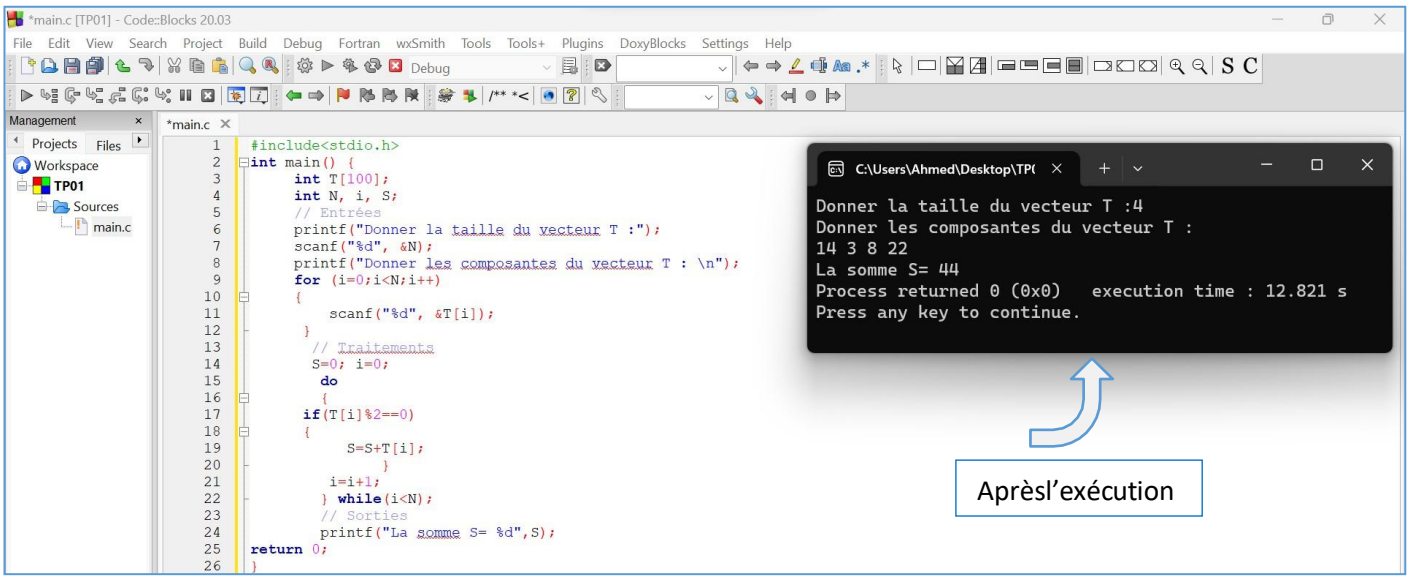


6- Boucle Répéter (do-while)

Algorithme/Programme C

Algorithme	Programme C

Algorithme Vecteur;
Variables
T:Tableau [0..99] **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←0 **à** N-1 **faire**
Lire(T[i]);
FinPour;
//Traitements
S ← 0; i ← 0;
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.
	#include <stdio.h>
	int main( ) {
	int T[100];
	int N,i,S;
	//Entrées
	printf("Donner la taille du vecteur T:");
	scanf("%d", &N);
	printf("Donner les composantes du vecteur T : \n");
	for(i=0;i<N;i++)
	{
	scanf("%d",&T[i]);
	}
	//Traitements
	S=0; i=0;
	do
	{
	if(T[i]%2==0)
	{
	S=S+T[i];
	}
	i=i+1;
	}**while(i<N);**
	// Sorties
	printf("La somme S=%d ",S);
	return 0;
	}



Remarques:

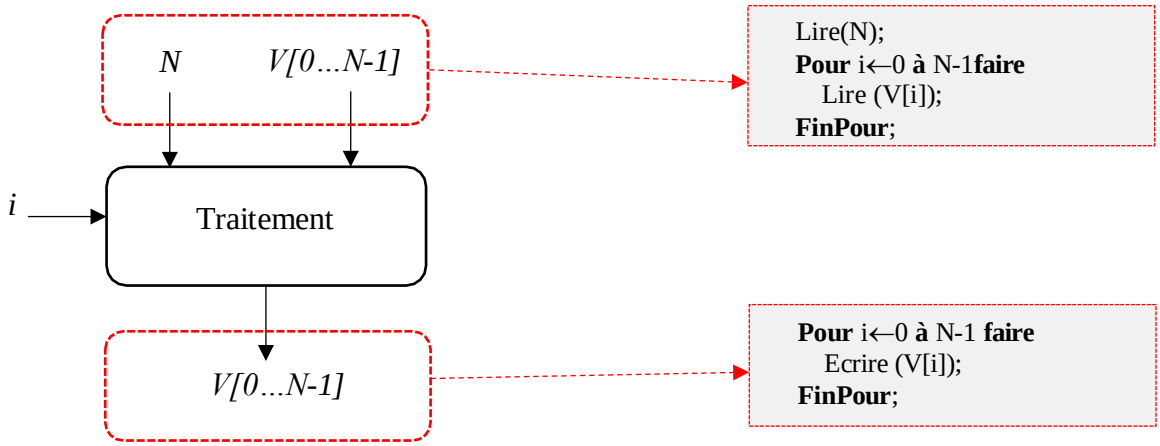
- Un vecteur (Tableau à une dimension) est une suite de cases mémoires adjacentes. Ces cases mémoires définissent des variables de même types.
- La déclaration suivante: T: **Tableau**[0..99]d'entier;
  Signifie que nous réservons100 cases de type entier.100 est la taille maximale du vecteur T. Chaque case est accessible par un indice qui peut prendre les valeurs 0 à 99.
- 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 **0** à la valeur **N-1**. Même chose pour l'affichage.

ExerciceN°02: *Lecture et Affichage d'un vecteur*

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

Solution :

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



Remarque:

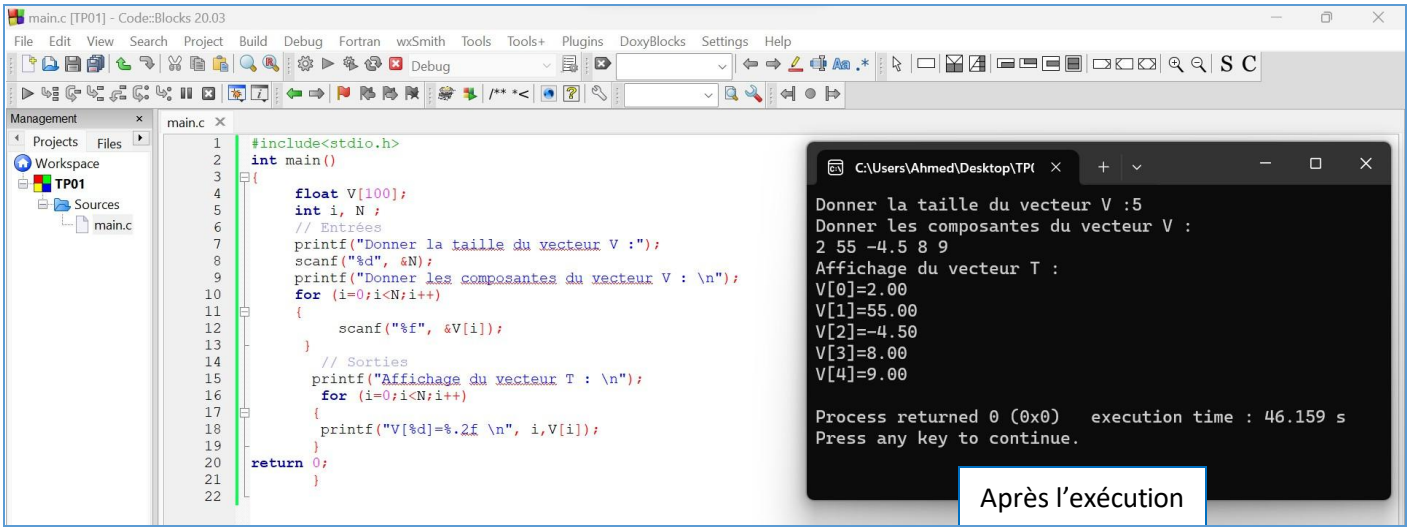
Il faut noter que ce programme ne réalise pas de traitement, il ne contient que des entrées et des sorties.

Algorithme	Programme C

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

Une autre méthode:

```
for (i=0;i<N;i++)
printf("%d", V[i]);
```



Explication ☺

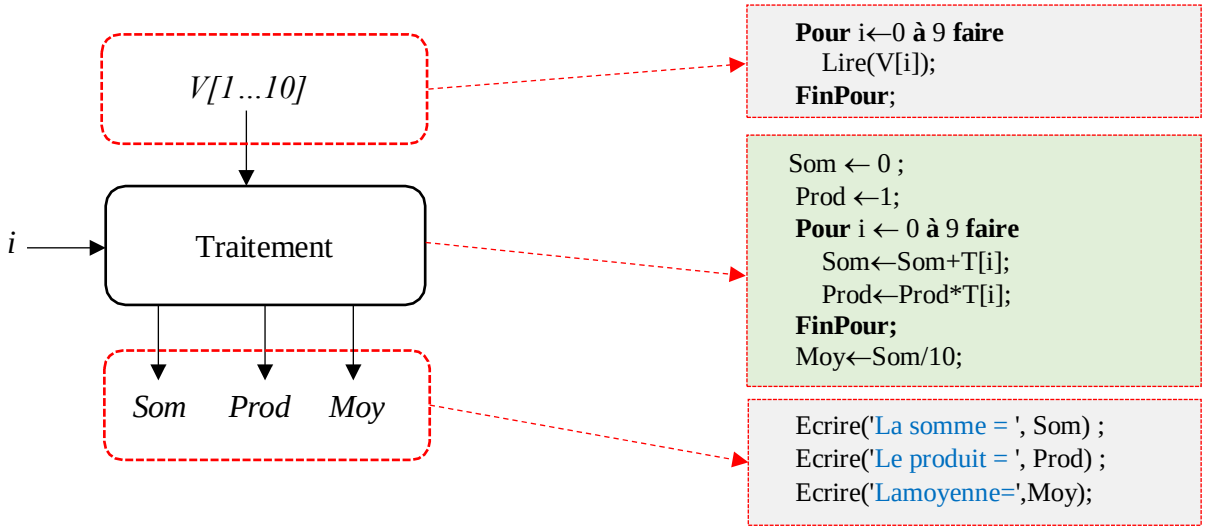
Ce programme montre comment lire et écrire un vecteur. Pour les deux opérations (lecture et écriture), nous aurons toujours besoin d'une boucle **Pour**. Lors de la déclaration, nous réservons 100 cases réelles (la taille maximale du vecteur), et nous utilisons la variable **N** pour déterminer la taille que nous voulons utiliser (par exemple 5 cases). L'accès à une composante d'indice **i** se fait comme suit : **V[i]**. Ainsi, pour lire la case 2, nous écrivons **Lire(V[1])**, et **Ecrire(V[3])** pour afficher la valeur de la 4ème case.

ExerciceN°03 : *La somme, le produit et la moyenne des éléments d'un tableau*

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

Solution :

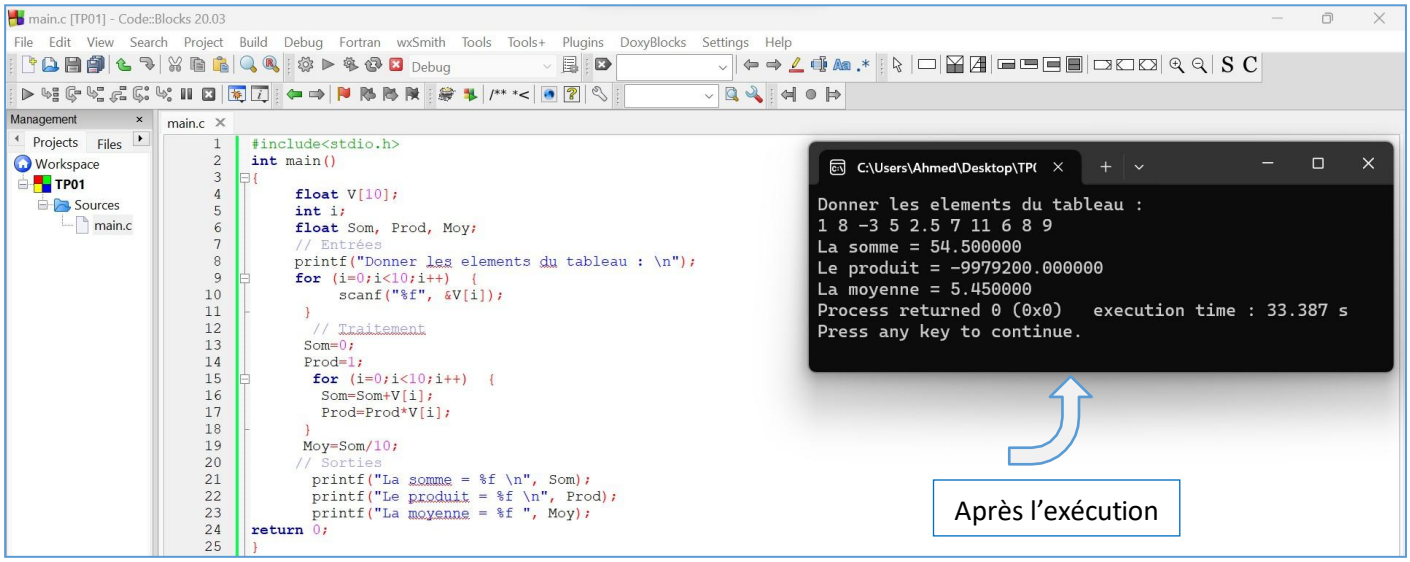
Les variables d'entrée, variable de sorite et la partie traitement sont présentées dans le schéma ci-dessous:



Algorithme/Programme C

Algorithme	Programme C

Algorithme Som_Moy_Prod_Tab;
Variables
V: Tableau [0..9] **de réel**;
i : **entier** ;
Som,Prod, Moy: **réel**;
Début
//Entrées
Ecrire("Donnez les éléments du tableau:");
Pour i←0 **à** 9 **faire**
Lire(V[i]);
FinPour;
 	
//Traitement
Som ← 0 ;
Prod ← 1 ;
Pour i ← 0 **à** 9 **faire**
Som←Som+V[i];
Prod←Prod*V[i];
FinPour;
Moy←Som/10;
 	
//Sorties
Ecrire("La somme = ", Som) ;
Ecrire("Le produit = ", Prod) ;
Ecrire("Lamoyenne=",Moy);
Fin.
	#include <stdio.h>
	int main( )
	{
	float V[10];
	int i;
	float Som,Prod, Moy;
	
	//Entrées
	printf("Donner les éléments du tableau:\n");
	for(i=0;i<10;i++){
	scanf("%f",&V[i]);
	}
	
	//Traitement
	Som=0;
	Prod=1;
	for(i=0;i<10;i++){
	Som=Som+V[i];
	Prod=Prod*V[i];
	}
	Moy=Som/10;
	
	//Sorties
	printf("La somme=%f\n",Som);
	printf("Le produit=%f\n",Prod);
	printf("La moyenne=%f",Moy);
	return 0;
	}



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.

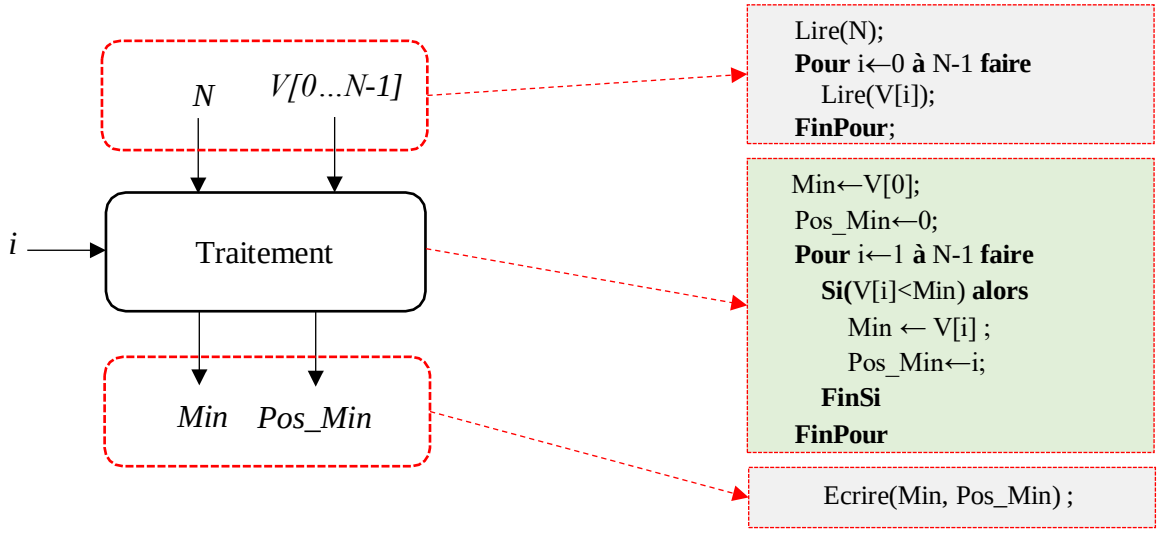
Pour calculer la moyenne, il suffit de diviser la somme par le nombre de cases du tableau.

ExerciceN°04:Le Min et le Max dans un vecteur

1. Écrire un algorithme / programme en C qui permet de rechercher le plus petit élément dans un vecteur réel **V**, ainsi que sa position.
2. Modifier le programme en C afin de rechercher le plus grand élément dans un vecteur réel **V** et d'afficher sa position.

Solution:(Question1)

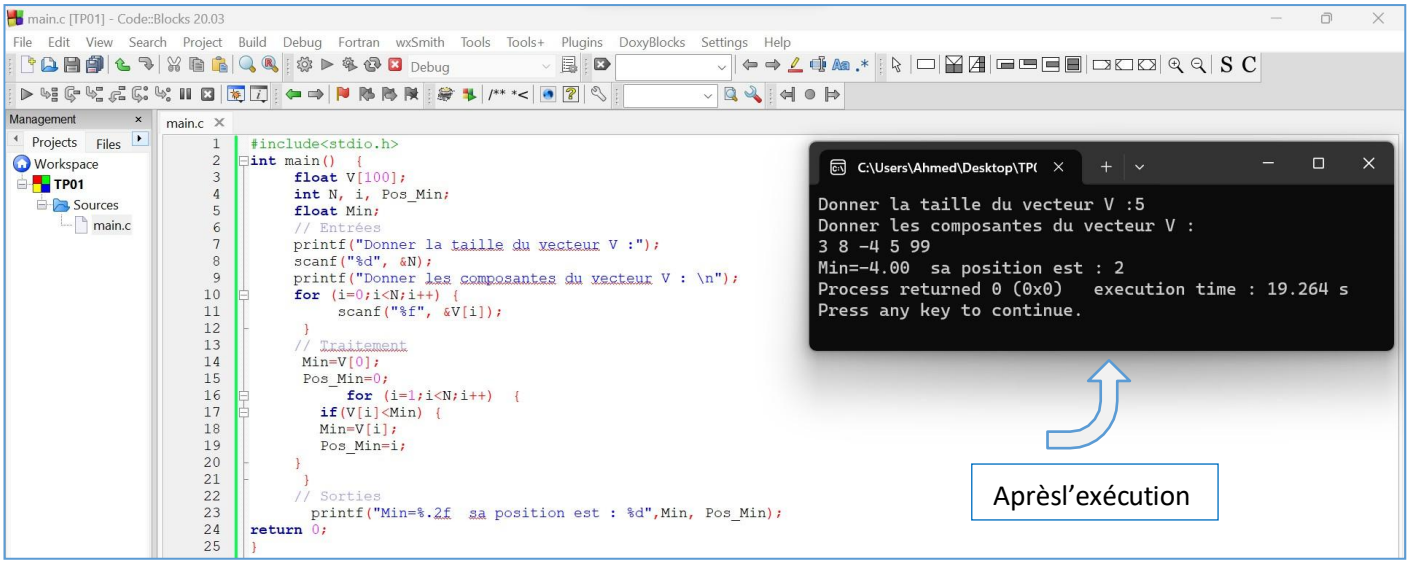
Les variables d'entrée, variables de sorite et la partie traitement sont représentées dans le schéma ci-dessous:



Algorithme /Programme C

Algorithme	Programme C

Algorithme Min_PosMin;
Variables
V:Tableau [0..99] **de réel;**
i,N,Pos_Min: **entier**;
Min : **réel** ;
Début
//Entrées
Ecrire("Donner la taille du vecteur V:");
Lire(N) ;
Ecrire("Donner les composantes du vecteur V:");
Pour i←0 **à** N-1 **faire**
Lire(V[i]);
FinPour
//Traitement
Min←V[0]; Pos_Min← 0 ;
Pour i←1 **à** N-1 **faire**
Si(V[i]<Min) **alors**
Min ← V[i] ;
Pos_Min←i;
FinSi
FinPour
//Sorties
Ecrire(Min, Pos_Min);
Fin.
	#include <stdio.h>
	int main( ){
	float V[100];
	int N,i,Pos_Min;
	float Min;
	//Entrées
	printf("Donner la taille du vecteur V:");
	scanf("%d", &N);
	printf("Donner les composantes du vecteur V:\n");
	for (i=0;i<N;i++) {
	scanf("%f",&V[i]);
	}
	//Traitement
	Min=V[0]; Pos_Min=0;
	for (i=1;i<N;i++){
	if(V[i]<Min) {
	Min=V[i];
	Pos_Min=i;
	}
	}
	//Sorties
	printf("Min=%.2f sa position est:%d",Min, Pos_Min);
	return 0;
	}

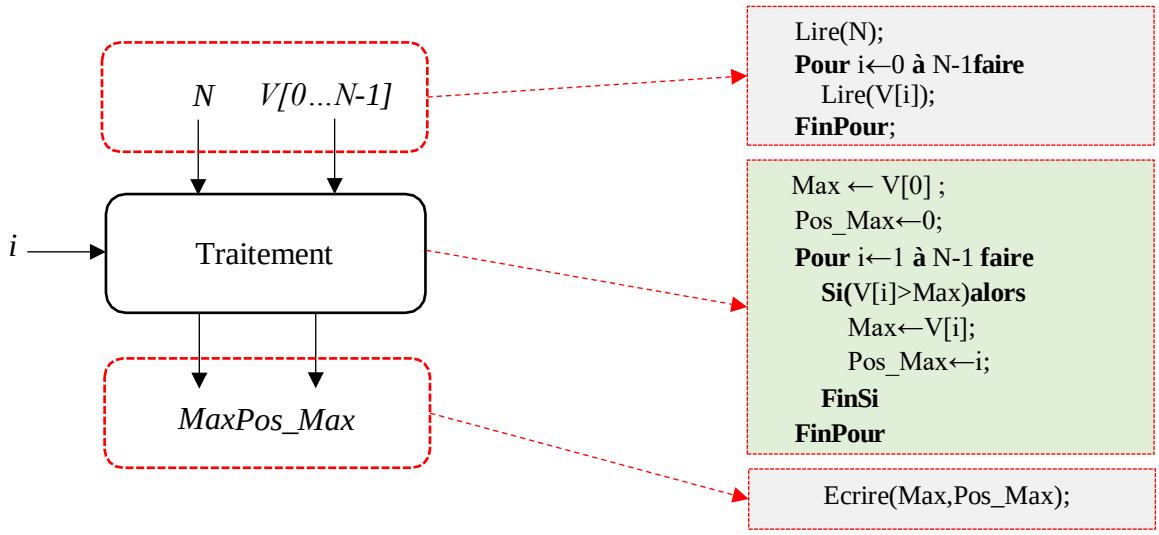


Explication ☺

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 (*Min:=V[0] ; Pos_Min:=0;*). 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_Min* par *i*.

Solution:(Question2)

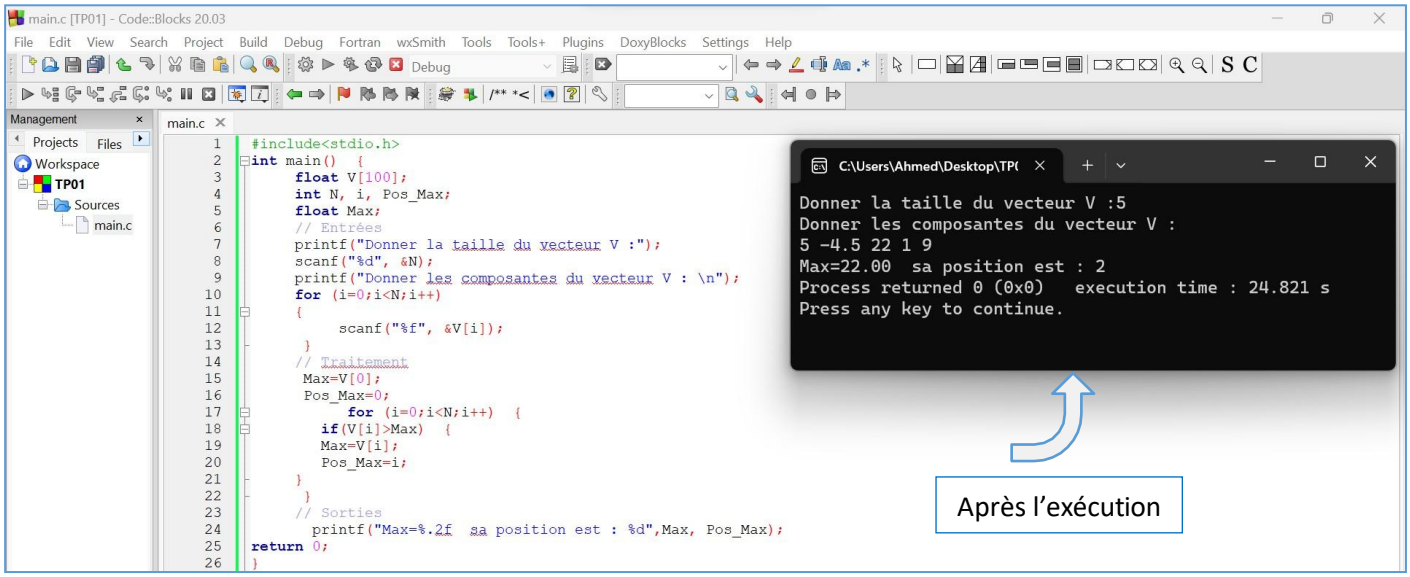
Les variables d'entrée, variables de sorite et la partie traitement sont représentées dans le schéma ci-dessous:



Algorithme/Programme C

Algorithme	Programme C

Algorithme Max_PosMax;
Variables
V:Tableau[0..99] **de réel**;
i, N, Pos_Max : **entier** :
Max:**réel**;
Début
//Entrées
Ecrire("Donner la taille du vecteur V:");
Lire(N) ;
Ecrire("Donner les composantes du vecteur V:");
Pour i←0 **à** N-1 **faire**
Lire(V[i]);
FinPour
// Traitement
Max ← V[0] ;
Pos_Max←0;
Pour i←1 **à** N-1 **faire**
Si(V[i]>Max)**alors**
Max ← V[i] ;
Pos_Max←i;
FinSi
FinPour
//Sorties
Ecrire("Max = ",Max, "Pos= ", Pos_Max);
Fin.
	#include<stdio.h>
	int main( ){
	float V[100];
	int N,i,Pos_Max;
	float Max;
	//Entrées
	printf("Donner la taille du vecteur V:");
	scanf("%d", &N);
	printf("Donner les composantes du vecteur V:\n");
	for (i=0;i<N;i++)
	{
	scanf("%f",&V[i]);
	}
	//Traitement
	Max=V[0];
	Pos_Max=0;
	for(i=0;i<N;i++){
	if(V[i]>Max){
	Max=V[i];
	Pos_Max=i;
	}
	}
	//Sorties
	printf("Max=%.2f sa position est:%d ",Max, Pos_Max);
	return 0;
	}



NB :

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