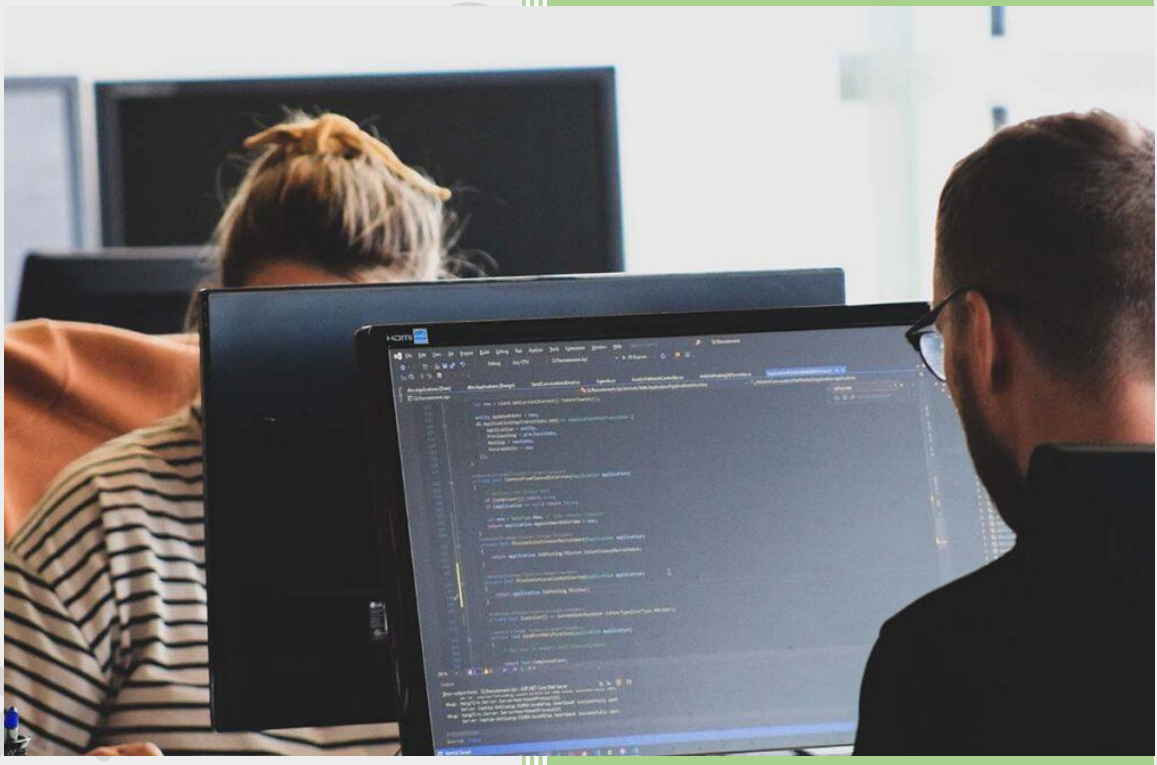
2023/2024

Programmation



OUARET Ahmed

Université A/Mira de Bejaia

2023/2024

Les grandes lignes du cours Programmation

Chapitre 1 : Variables indicées (Tableaux)

1. Introduction
2. Les tableaux à une dimension (Vecteurs)
3. Les tableaux à deux dimensions (Matrices)

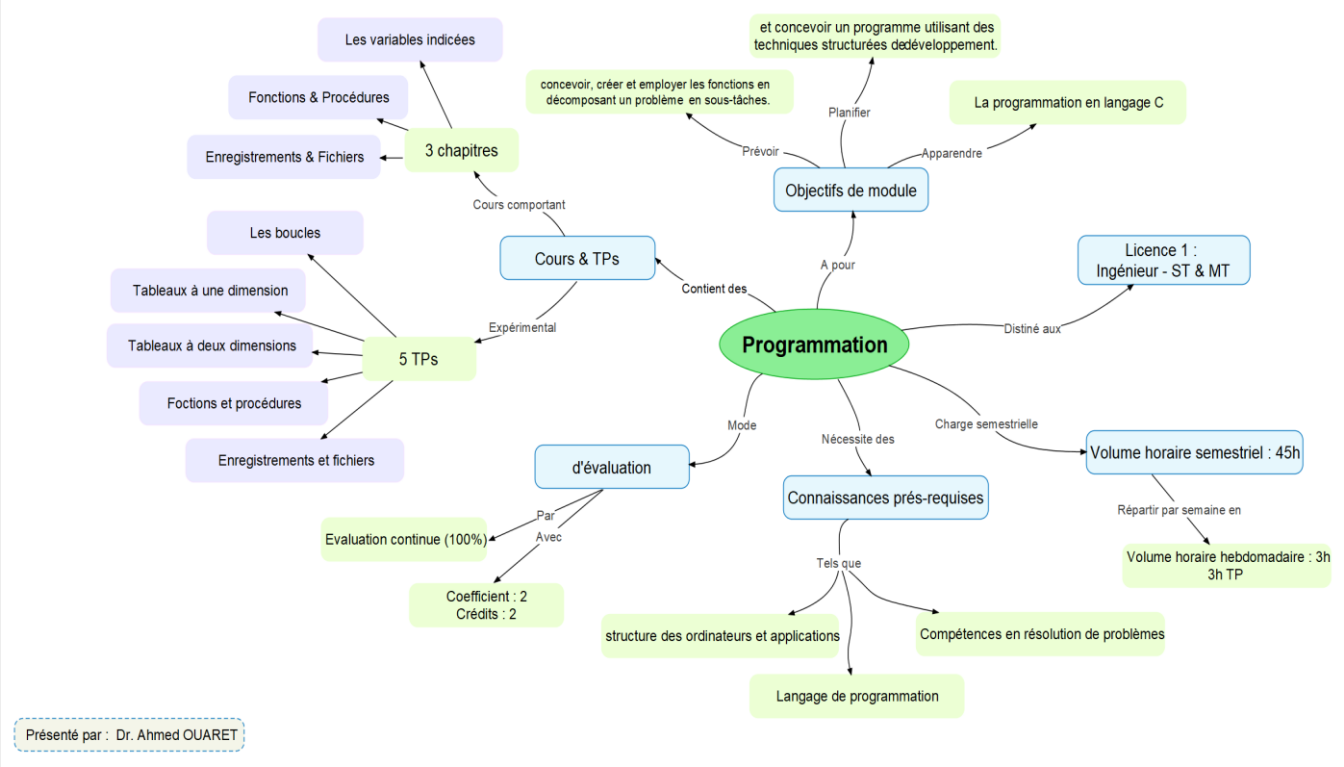
Chapitre 2 : Fonctions & procédures

1. Introduction
2. Déclaration de sous-programme
3. Passage de paramètres (transmission de paramètres)

Chapitre 3 : Enregistrements & fichiers

1. Introduction
2. Les enregistrements
3. Les fichiers

Carte mentale de cours Programmation



Chapitre 01 : Les tableaux

I. Introduction

Imaginons que dans un programme, nous avons besoin d'un grand nombre de variables, il devient difficile de donner un nom à chaque variable.

Exemple :

Ecrire un algorithme permettant de saisir cinq notes et de les afficher après avoir multiplié toutes les notes par trois.

Solution :

```
Algorithme Note ;
Variables
   N1, N2, N3, N4, N5 : réel ;
Début
   Ecrire ('Enter la 1ère Note') ;
   Lire(N1) ;
   Ecrire ('Enter la 2ème Note') ;
   Lire(N2) ;
   Ecrire ('Enter la 3ème Note') ;
   Lire(N3) ;
   Ecrire ('Enter la 4ème Note') ;
   Lire(N4) ;
   Ecrire ('Enter la 5ème Note') ;
   Lire(N5) ;
   Ecrire('La note 1 multipliée par 3 est : ',N1*3) ;
   Ecrire('La note 2 multipliée par 3 est : ',N2*3) ;
   Ecrire('La note 3 multipliée par 3 est : ',N3*3) ;
   Ecrire('La note 4 multipliée par 3 est : ',N4*3) ;
   Ecrire('La note 5 multipliée par 3 est : ',N5*3) ;
Fin.
```

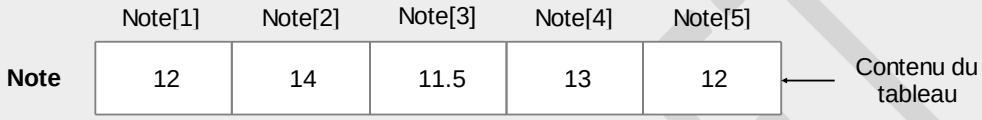
La même instruction se répète cinq fois. Imaginons que si l'on voudrait réaliser cet algorithme avec 100 notes, cela deviendrait fastidieux.

Comme les variables ont des noms différents, on ne peut pas utiliser de boucle, ce qui allonge considérablement le code et le rend très répétitif.

Pour résoudre ce problème, il existe un type de données qui permet de définir plusieurs variables de même type.

II. Définition

Un tableau est une suite d'éléments de même type, Il utilise plusieurs cases mémoire à l'aide d'un seul nom. Comme toutes les cases portent le même nom, elles se différencient par un numéro ou un indice. Nous pouvons représenter schématiquement un tableau nommé **Note** composé de cinq cases, dans la mémoire comme suit :



Nous disposons alors de cinq variables. Pour les nommer, on indique le nom du tableau suivi de son indice entre crochets ou entre parenthèses : La première s'appelle Note[0], la deuxième Note[1], etc. jusqu'à la dernière Note[4].

Note[3] représente le 4ème élément du tableau **Note** et vaut 13.

III. Tableau à une dimension

III.1. Déclaration

La déclaration d'un tableau permet d'associer à un nom une zone mémoire composée d'un certain nombre de cases mémoires de même type.

Syntaxe :

Algorithme

Variable

Variable_identificateur : tableau [Indice_min .. Indice_max] de **type** ;

Langage C

Var

Type Variable_identificateur[nbre d'éléments] ;

- Le premier élément d'un tableau porte l'indice zéro ou l'indice 1 selon les langages (0 en langage C).
- La valeur d'un indice doit être un nombre entier.
- La valeur d'un indice doit être inférieure ou égale au nombre d'éléments du tableau. Par exemple, avec le tableau tab[1..20], il est impossible d'écrire tab[21]. Cette expression fait référence à un élément qui n'existe pas.

Exemple :
L'instruction suivante déclare un tableau de 30 éléments de type réel :

Algorithme	Pascal
Variable Note : tableau [0..29] de réels ;	**Var** float Note [30] ;

Note : c'est le nom du tableau (identificateur).
0 : c'est l'indice du premier élément du tableau.
29 : c'est l'indice du dernier élément du tableau (nombre d'éléments du tableau).

Exercice :
Déclarer deux tableaux nommés A et B composé chacun d'eux de 10 éléments de type réels.

Solution :

Algorithme	Pascal
Variable A, B : tableau [0..9] de réels ;	**Var** float A[10], B[10] ;

Remarques ☺:

- Lorsqu'on déclare un tableau, on déclare aussi de façon implicite toutes les variables indicées qui le constituent. Nous utiliserons la valeur 0 comme premier indice (indice minimum) et l'indice maximum sera égal au nombre d'éléments -1.

Exemple 1 :
L'instruction suivante affecte à la variable X la valeur du premier élément du tableau Note :

Algorithme	Pascal
X ← Note [0] ;	X = Note [0] ;

Exemple 2 :

L'instruction suivante affiche le contenu du quatrième élément (entier) du tableau Note :

Algorithme	Pascal
Ecrire(Note[3]) ;	printf("%d", Note[3]) ;

Exemple 3 :

L'instruction suivante affecte une valeur introduite par l'utilisateur à l'élément trois du tableau Note :

Algorithme	Pascal
Lire(Note[2]) ;	scanf("%d",&Note[2]) ;

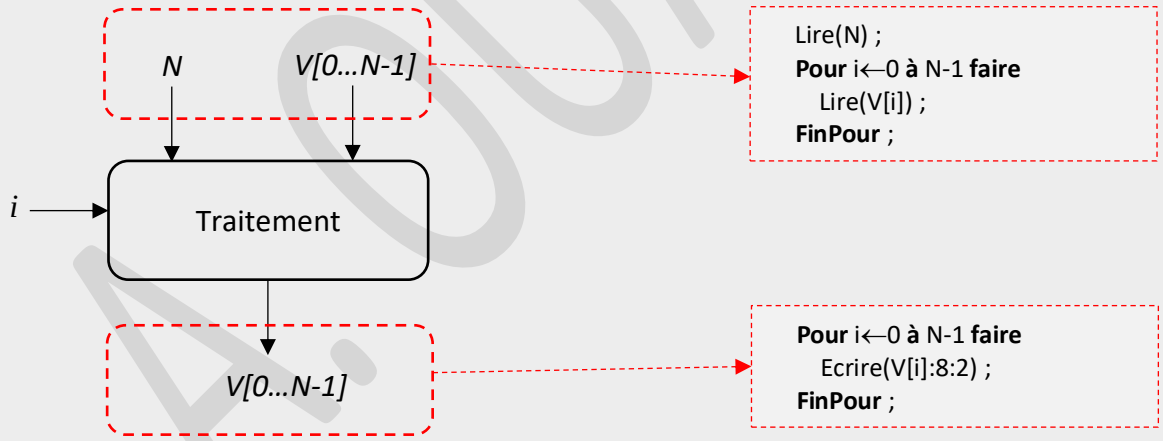
► Exercices avec corrigés :

Exercice 1: Lecture et Affichage d'un vecteur

Ecrire 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:



Remarques ☺

- Il faut noter que ce programme ne réalise pas de traitement, il ne contient que des entrées et des sorties.
- La variable ***i*** est une variable de traitement ou intermédiaire, utilisée pour parcourir le vecteur T.

Algorithme	Langage 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.**	

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(T[3])** pour afficher la valeur de la 4ème case.

Exercice 2 :

Ecrire un algorithme/programme C permettant de saisir 10 notes et de les afficher après avoir multiplié toutes ces notes par un coefficient fourni par l'utilisateur :

Solution :

Algorithme	Langage C
Algorithme tableau_note ; **Variables** Note : tableau [0..9] de réels ; Coef, i : entier ; **Début** Ecrire('Entrer le coefficient : ') ;	

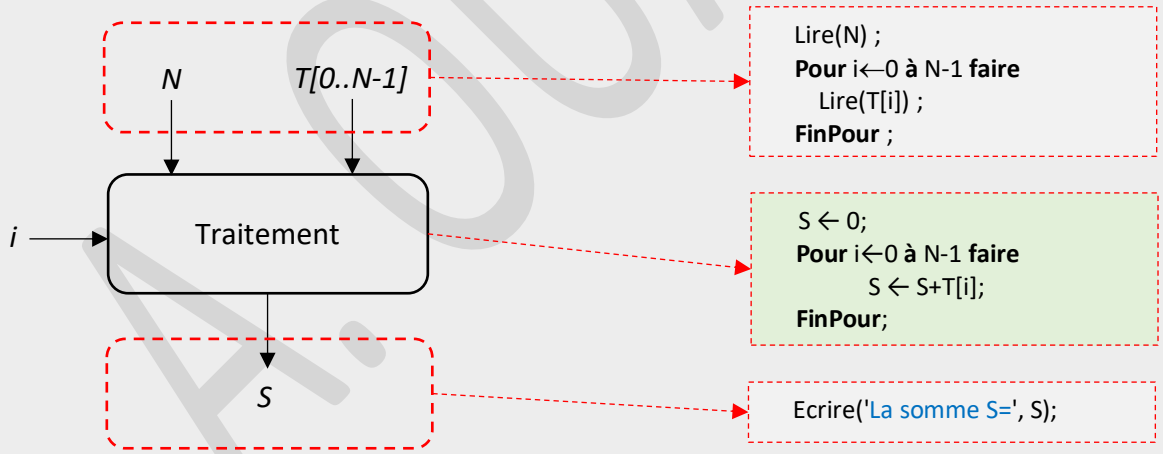
Lire(Coef) ; {Remplissage du tableau Note} **Pour** i ← 0 **à** 9 **faire** Ecrire('Entrer la valeur de la note ', i,' : ') ; Lire(Note[i]) ; **Fin-Pour ;** Ecrire('Affichage des notes*Coef : ') ; **Pour** i ← 0 **à** 9 **faire** Ecrire(Note[i]*coef) ; **Fin-Pour ;** **Fin.**	

Exercice 3 :

Ecrire un algorithme/programme C qui calcul la somme des éléments d'un tableau.

Solution :

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



On suppose que le nombre d'éléments du tableau ne dépasse pas 100.

Algorithme	Langage C
Algorithme Somme_tableau ; **Variable** T : tableau [0..99] de réels ; N, i : entier ; S : réel ; **Début** Ecrire('Entrer la taille du tableau : ') ;	

```
   Lire(N) ;
   Si (N>M) alors
      N ← M ;
   Fin-Si ;
   Ecrire('Remplissage du tableau : ') ;
   Pour i ← 0 à N-1 faire
      Lire(T[i]) ;
   Fin-Pour ;
   S ← 0 ;
   Pour i ← 0 à N-1 faire
      S ← S+T[i] ;
   Fin-Pour ;
   Ecrire('La somme des éléments du tableau
           est : ', S) ;
Fin.
```

Exercice 4 : (La recherche du plus petit et plus grand élément dans un tableau)

Ecrire un algorithme permettant de déterminer la valeur maximale et minimale dans un tableau de dix entiers, avec leurs positions dans le tableau. Traduire l'algorithme en programme en langage C.

Exercice 5 : (Le tri d'un tableau par sélection)

Ecrire un algorithme permettant de trier un tableau de dix entiers en ordre croissant. Traduire l'algorithme en programme en langage C.

Exercice 6 :

Ecrire un algorithme permettant de saisir un tableau de N éléments des valeurs réelles non nulles. Une fois la saisie terminée, le programme affichera le nombre de valeurs négatives et le nombre de valeurs positives.

Exercice 7 :

Soit T un tableau contenant N entiers (10≤N≤50). On propose d'écrire un programme en langage C qui permet d'éclater T en deux tableaux :

TN (contenant les éléments négatifs de T) et **TP** (contenant les éléments positifs de T).

Exemple :

T	2	7	-4	0	-7	4	3	0	-8	-1	3

TN	-4	-7	-8	-1

TP	2	7	0	4	3	0	3

Exercice 8 : (Somme de deux vecteurs T1 et T2)

Ecrire un algorithme permettant de faire la somme de deux vecteurs T1 et T2 de N éléments des valeurs réelles. Traduire l'algorithme en programme en langage C.

Exercice 9 : (Produit scalaire de deux vecteurs T1 et T2)

Ecrire un algorithme permettant de calculer et afficher le produit scalaire de deux vecteurs T1 et T2 de N éléments des valeurs réelles. Traduire l'algorithme en programme en langage C.

Exercice 10 : (Somme des éléments divisible par 3 d'un vecteur T)

Ecrire un algorithme permettant de calculer et afficher la somme des éléments divisible par 3 d'un vecteur T de type réel et de taille N. Traduire l'algorithme en programme en langage C.

Exercice 11 :

Ecrire un algorithme**/**programme C qui permet de lire un tableau de N éléments réels, qui calcule et affiche le nombre d'éléments égaux au dernier élément lu.

Exemple : pour le tableau suivant : le nombre = 5

4	-3	4	4	6	4	12	4	-9	4

Exercice 12 :

Ecrire un algorithme**/**programme C qui permet de lire un tableau T de N éléments réels, qui calcule et affiche le nombre d'éléments égaux au premier élément lu.

Exemple : pour le tableau suivant : le nombre = 5

4	-3	4	4	6	4	12	4	-9	4

à suivre

.....