

Faculté des Sciences  
كلية العلوم

Département d'Informatique  
Filière MIP - Semestre 2  
2024-2025

**Module : Informatique II ( Algorithmique II / Python )**

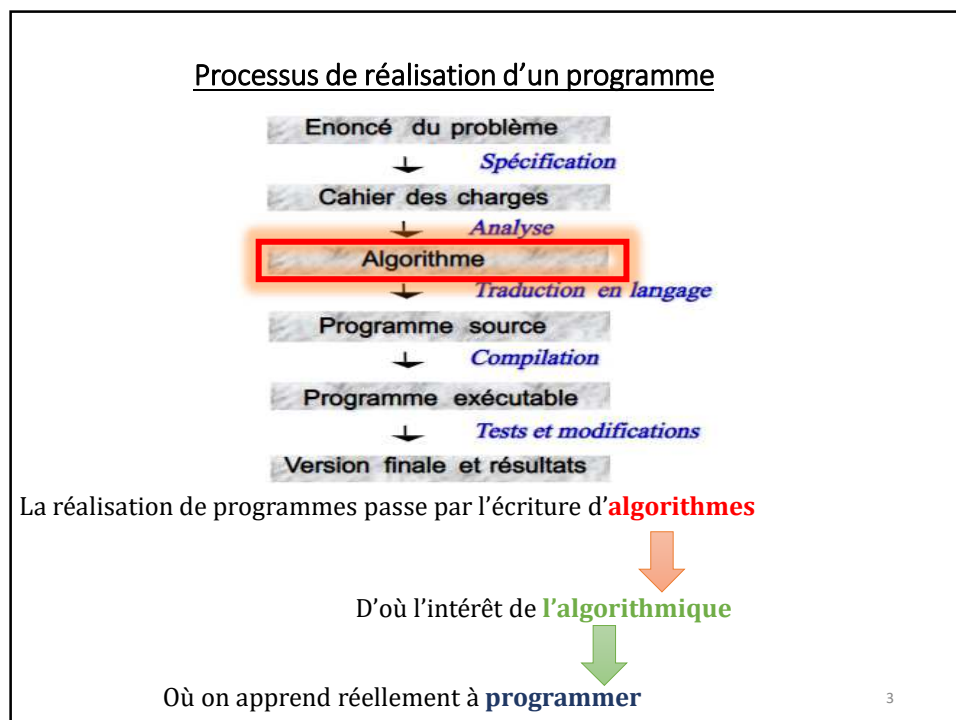
# Chapitre 1

# Introduction à l'algorithmique

Pr. Khadija Louzaoui

## Introduction à l'algorithmique

- ✚ Introduction
- ✚ Structure générale d'un algorithme
- ✚ Les variables et les constantes
- ✚ Les instructions élémentaires
- ✚ Les instructions conditionnelles ( les alternatives )
- ✚ Les instructions itératives ( Les boucles )
- ✚ Les tableaux



## Chapitre 1

## Introduction

**Algorithme**

- ❑ Le terme **algorithme** est employé en **informatique** pour décrire une méthode de résolution de problèmes programmables sur machine.
- ❑ Un **algorithme** décrit ce qui doit faire l'ordinateur pour arriver un but bien précis. Ce sont **les instructions** qu'on doit lui donner.
- ❑ Un **algorithme** est un ensemble d'instructions (nécessaires, ordonnées, précises, qui se terminent dans un temps fini) et qui agissent sur un ensemble de **données** pour avoir un **résultat**.
- ❑ La notion **d'algorithme** est à la **base** de toute la **programmation** informatique.

4


Chapitre 1

Introduction

Algorithme

❑ **L'algorithme** est un moyen pour le **programmeur** de présenter son approche d'un problème donné à d'autres personnes, dans un **langage** clair et compréhensible par l'être humain.

❑ C'est un **pseudo-langage** qui est conçu pour résoudre les problèmes et applications sans aucune contrainte due aux langages de programmation et aux spécificités de la machine.



5

Chapitre 1

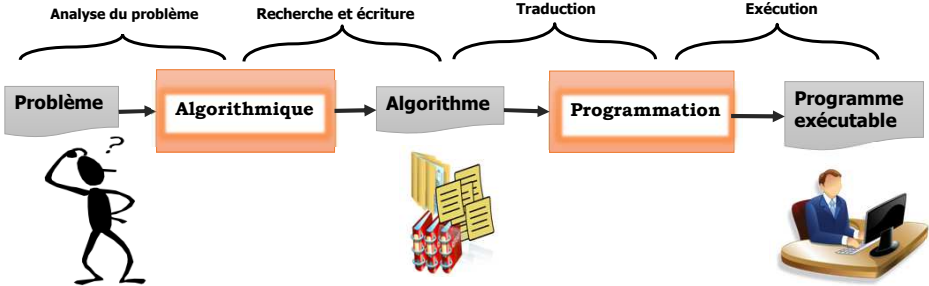
Introduction

Algorithmique

❑ **L'algorithmique** est **la science des algorithmes**. Elle s'intéresse à l'art de construire des algorithmes ainsi qu'à déterminer leur validité, leur robustesse, leur réutilisabilité, leur complexité ou leur efficacité.

❑ L'algorithmique permet ainsi de passer d'un **problème** à résoudre à un **algorithme** qui décrit la démarche de résolution du problème.

Analyse du problème      Recherche et écriture      Traduction      Exécution



Problème → Algorithmique → Algorithme → Programmation → Programme exécutable

Deux phases nécessaires pour obtenir un programme exécutable.

6

Chapitre 1	Introduction
<p><b>Phase d'algorithmique</b></p> <ul style="list-style-type: none"> <li>▪ <b>Phase d'algorithmique</b> qui implique la recherche et l'écriture d'un <b>algorithme</b>.</li> <li>▪ On doit définir les <b>données</b> qu'on dispose et les <b>objectifs</b> qu'on souhaite atteindre, ainsi que prévoir des réponses à tous les cas possibles.</li> </ul> <div style="display: flex; align-items: center; justify-content: space-around; margin: 20px 0;"> <div style="text-align: center;"> <div style="border: 1px solid gray; background-color: #d3d3d3; padding: 5px; width: 100px;">Donnée d'entrée</div> <div style="display: inline-block; width: 150px; height: 1px; background-color: black; margin: 0 10px;"></div> </div> <div style="text-align: center;"> <div style="border: 2px solid orange; padding: 5px; width: 100px;">Traitement</div> </div> <div style="text-align: center;"> <div style="border: 1px solid gray; background-color: #d3d3d3; padding: 5px; width: 100px;">Donnée de sortie</div> <div style="display: inline-block; width: 150px; height: 1px; background-color: black; margin: 0 10px;"></div> </div> </div> <div style="display: flex; justify-content: space-between;"> <div style="width: 30%;"> <p>Il s'agit de <b>repérer</b> les <b>données</b> nécessaire à la <b>résolution</b> du <b>problème</b>.</p> </div> <div style="width: 30%;"> <p>Il s'agit de <b>déterminer</b> toutes les <b>étapes</b> des <b>traitements</b> à faire et donc des " <b>instructions</b>" à <b>développer</b> pour arriver aux résultats.</p> </div> <div style="width: 30%;"> <p>Les <b>résultats</b> obtenus peuvent être <b>affichés</b> sur écran, ou <b>imprimés</b> sur papier, ou bien encore <b>conservés</b> dans un fichier.</p> </div> </div> <p style="text-align: center; margin-top: 20px;"><b>Principe du traitement automatisé</b></p>	

7

Chapitre 1	Introduction
<p><b>Phase d'algorithmique</b></p> <p>Lorsqu'on écrit un <b>algorithme</b>, les questions suivantes doivent être considérées :</p> <ul style="list-style-type: none"> <li>▪ Quel est le résultat attendu ? (données de sortie)</li> <li>▪ Quelles sont les données nécessaires (informations requises) ?</li> <li>▪ Comment faire (le traitement à réaliser) ?</li> </ul> <p><b>Exemple :</b> Résolution d'une équation du premier degré <b><math>ax+b=0</math></b>.</p> <ul style="list-style-type: none"> <li>▪ <b>Les données d'entrées</b> sont : a, b</li> <li>▪ <b>La donnée de sortie</b> est : x</li> <li>▪ <b>Traitement:</b> Etudier les cas suivants : <ul style="list-style-type: none"> <li>- a=0 et b≠0</li> <li>- a=0 et b=0</li> <li>- a ≠ 0</li> </ul> </li> </ul>	

8

Chapitre 1	Introduction
<p><b>Phase de programmation</b></p> <ul style="list-style-type: none"> <li>▪ <b>Phase de programmation</b> qui consiste à traduire l'algorithme obtenu en un programme à l'aide d'un <b>langage de programmation (code source)</b>.             <ol style="list-style-type: none"> <li>1-Le processeur <b>extraît</b> les <u>données</u> à traiter à partir de la source indiquée dans le programme (soit auprès de l'utilisateur qui devrait les introduire au moyen du clavier, soit en mémoire secondaire ou centrale).</li> <li>2- Il <b>exécute</b>, ensuite, la <u>série d'opérations</u> élémentaires de manière séquentielle (dans l'ordre prévu par le programme) et mémorise tous les résultats intermédiaires.</li> <li>3- Il <b>renvoie</b> enfin le ou les <u>résultats</u> attendus à la destination indiquée dans le programme.</li> </ol> </li> </ul>	
	9

Chapitre 1	Introduction
<p><b>Langage de programmation</b></p> <ul style="list-style-type: none"> <li>❑ Un <b>programme</b>, lorsqu'il est sous la forme de <b>code source</b>, est un texte qui exprime un <b>algorithme</b>, en vue de permettre l'<b>exécution</b> de ce dernier sur une machine.</li> <li>❑ Un <b>langage de programmation</b> permet d'écrire un <b>programme</b> (JavaScript, Python, Java, Typescript, C#, C++, PHP, C, Ruby...)</li> <li>❑ On peut distinguer deux grands types de langages : <b>les langages interprétés</b> et <b>les langages compilés</b>.             <ul style="list-style-type: none"> <li>▪ Dans un <b>langage interprété</b>, le même code source pourra marcher directement sur <b>tout ordinateur</b>. Avec un <b>langage compilé</b>, il faudra (en général) tout <b>recompiler</b> à chaque fois ce qui pose parfois des soucis.</li> <li>▪ Dans un <b>langage compilé</b>, le programme est directement <b>exécuté</b> sur l'ordinateur, donc il sera en général <b>plus rapide</b> que le même programme dans un <b>langage interprété</b>.</li> </ul> </li> </ul>	
	10

Chapitre 1	Introduction
<p><b>Langage de programmation Python</b></p> <p>Le langage de programmation Python a été créé en 1989 par Guido van Rossum.</p> <p>Ce langage de programmation présente de nombreuses caractéristiques intéressantes :</p> <ul style="list-style-type: none"> <li>➤ Multiplateforme.</li> <li>➤ Gratuit.</li> <li>➤ Langage de haut niveau.</li> <li>➤ Langage interprété.</li> <li>➤ Langage orienté objet.</li> <li>➤ Simple à prendre.</li> <li>➤ Très utilisé en bio-informatique et plus généralement en analyse de données et IA.</li> </ul> <p>Toutes ces caractéristiques font que Python est désormais enseigné dans de nombreuses formations, du lycée à l'enseignement supérieur.</p>	



11

Chapitre 1	Structure générale d'un algorithme
<p><b>Structure générale d'un algorithme</b></p> <p>Un algorithme est composé de trois parties principales :</p> <p><b>L'en-tête :</b> cette partie sert à donner un <u>nom</u> à l'algorithme et identifie le problème à résoudre.</p> <ul style="list-style-type: none"> <li>▪ Elle est précédée par le mot clé <b>Algorithme</b>.</li> </ul> <p><b>La partie déclarative :</b> cette partie rassemble les déclarations des différents objets non primitifs que l'algorithme utilise dans son traitement.</p> <ul style="list-style-type: none"> <li>▪ Elle est précédée par les mots <b>variables, constantes</b>, etc..</li> </ul> <p><b>Le corps de l'algorithme :</b> cette partie décrit le traitement de l'algorithme, elle est constituée d'une ou plusieurs séquences <b>d'instructions</b> faisant appel à des opérations de base à exécuter par l'ordinateur.</p> <ul style="list-style-type: none"> <li>▪ Elle est délimitée par les mots <b>Début</b> et <b>Fin</b>.</li> </ul>	

12

Chapitre 1

Structure générale d'un algorithme

Structure générale d'un algorithme

En-tête

Partie déclarative

Corps de l'algorithme

Algorithme

Variables

Constantes

Début

Fin

Nom Algorithme

Identificateur : type

Identificateur = valeur

Instruction 1  
Instruction 2  
.....  
Instruction n

13

Chapitre 1

Structure générale d'un algorithme

Structure générale d'un algorithme

Exemple : Un algorithme qui calcul la surface d'un cercle

Notation algorithmique

Algorithme

Variables

Constante

Début

Ecrire

Lire

$s \leftarrow r * r * \pi$

Ecrire

Fin

SurfaceCercle  
r,s : réel  
 $\pi = 3,141592$   
Donner la valeur du rayon:  
r  
La surface du cercle est : , s )

Python

In [1]:

```
pi = 3.141592
r = float(input("Donner la valeur du rayon : "))
s = r * r * pi
print("La surface du cercle est :", s)
```

Donner la valeur du rayon : 8

La surface du cercle est : 201.061888

Chapitre 1

Les variables et les constantes

Notion de variable

❑ Dans un programme, on va avoir en permanence besoin de stocker provisoirement des valeurs.

▪ Des données issues du disque dur, fournies par l'utilisateur (frappées au clavier).

▪ Des résultats obtenus par le programme, intermédiaires ou définitifs.

▪ Ces données peuvent être de plusieurs types : des nombres, du texte, etc.

❑ Toujours dès que l'on a besoin de stocker une information au cours d'un programme, on utilise une variable.

nom

âge

situation

note

vie

score

variables

15

Chapitre 1

Les variables et les constantes

Notion de variable

❑ Pour employer une image, une variable est une boîte, que le programme (l'ordinateur) va repérer par une étiquette (nom).

❑ Pour avoir accès au contenu de la boîte, il suffit de la désigner par son étiquette.

❑ On peut à chaque fois changer le contenu de la boîte.

Information

étiquette

Donnée

valeur

nom

❑ Une variable sert à stocker la valeur d'une donnée dans un langage de programmation.

❑ Une variable désigne un emplacement mémoire dont le contenu peut changer au cours d'un programme (d'où le nom de variable).

16

Pr. Louzaoui Khadija

8



Chapitre 1

Les variables et les constantes

Notion de variable

Variable et mémoire

❑ Une **variable** est un **espace mémoire** nommé de taille fixe, prenant au cours de déroulement du programme un nombre indéfini de **valeurs** différentes.

❑ L'**élément unitaire** de stockage de l'information est appelé **bit**. Un bit ne peut avoir que deux états distincts : **0** ou **1**.

❑ Dans la **mémoire** de l'ordinateur, les données sont manipulées par groupes de 8 bits (**octet,Byte**), ou plus (**mots** de 16, 32, 64 bits,...).

❑ Une **case mémoire** (cellule) est donc appelée **mot** (word).

❑ Pour que l'unité centrale puisse **stocker** une information et la **retrouver** dans la mémoire. Chaque mot a un **numéro** qui permet d'y faire référence de façon unique : c'est l'**adresse mémoire**.

0

0

0

bit

5248									
5249									
5250	1	0	0	1	1	0	1	0	
5251									
5252									
5253									
5254									
5245									
5256									
5257									

Adresse

Cellule mémoire

Mot

0

0

0

17

Chapitre 1

Les variables et les constantes

Notion de variable

Variable et mémoire

❑ Il y a donc deux façons de voir la mémoire centrale de l'ordinateur : côté **programmeur** et côté **ordinateur**.

❑ Dans la **programmation**, les **adresses** mémoire sont représentées par des **noms**. Le programmeur ne connaît pas donc l'**adresse** d'une case mais plutôt son **nom**.

Mémoire

Adresses	Mots
10000000	6
10000001	8
10000010	7
10000011	
...	

x

y

moyenne

Variables

Côté ordinateur

Côté programmeur

Pr. Louzaoui Khadija

9

Chapitre 1	Les variables et les constantes
<p><b>Déclaration des variables</b></p> <ul style="list-style-type: none"> <li>❑ La partie déclaration consiste à énumérer toutes les variables dont on aura besoin au cours de l'algorithme.</li> <li>❑ Chaque déclaration doit comporter le <b>nom</b> de variable (<b>identificateur</b>) et son <b>type</b>.             <ul style="list-style-type: none"> <li>❑ Un <b>identificateur</b> est le nom donné à une variable, une fonction, etc. Ce nom doit obligatoirement commencer par une lettre suivie d'une suite de lettres et les chiffres et il ne doit pas contenir d'espace.</li> </ul> </li> <li>❑ Les variables se déclarent au début de l'algorithme, avant le programme lui-même mais après le mot clé «<b>variable</b>» ou «<b>var</b>».</li> </ul> <div style="background-color: #ffffcc; padding: 10px; margin-top: 10px;"> <p><b>Syntaxe :</b></p> <p><b>variable</b> <b>identificateur</b> : <b>type</b></p> <p><b>variables</b> <b>liste d'identificateurs</b> : <b>type</b></p> </div>	
19	

Chapitre 1	Les variables et les constantes
<p><b>Déclaration des variables</b></p> <p style="text-align: center; color: red;"><b>Types de donnée</b></p> <ul style="list-style-type: none"> <li>❑ <b>Type de donnée</b> permet de déterminer :             <ul style="list-style-type: none"> <li>▪ L'ensemble des <b>valeurs</b> que peut prendre une variable.</li> <li>▪ L'ensemble des <b>opérations</b> qu'on peut les appliquer sur les variables.</li> <li>▪ Déterminer l'<b>espace mémoire</b> nécessaire au stockage de cette variable dans la mémoire.</li> </ul> </li> <li>❑ Les <b>types</b> les plus connus sont :             <ul style="list-style-type: none"> <li>▪ Le type <b>entier</b> : sert à manipuler les nombres entiers positifs ou négatifs</li> <li>▪ Le type <b>réel</b> : sert à manipuler les nombres à virgule.</li> <li>▪ Le type <b>caractère</b> : sert à manipuler des caractères alphabétiques et numériques.</li> <li>▪ Le type <b>chaîne de caractères</b> : sert à manipuler des chaînes de caractères permettant de représenter des mots ou des phrases.</li> <li>▪ Le type <b>booléen</b> : utilise les expressions logiques (vrai/ faux)</li> </ul> </li> </ul>	
20	

Chapitre 1

Les variables et les constantes

Déclaration des variables

Types de donnée

Type de données	Numérique		Alphanumérique		Booléen
Algorithmique	Entier	Réel	Caractères	Chaîne de caractères	Booléen
Exemples	-45 19	-5,6 9,2	'A','@' '6','?'	"bonjour" "G453929"	False True

Exemple:

Variables **a , b** : entier

**x** : réel

**preom** : chaine de caractères

**absent** : booléen

21

Chapitre 1

Les variables et les constantes

Déclaration des variables

Les opérations sur des variables

Type	Opérations	Symboles	Exemples
Entier	Addition	+	2 + 5 = 7
	Soustraction	-	8 – 5 = 3
	Multiplication	*	2 * 4 = 8
	Division	/	10 / 4 = 2,5
	Division entier	Div	10 Div 4 = 2
	Modulo (le reste de la division entier)	Mod	10 Mod 3 = 1
	Comparaison	≤ ≥ > < = ≠	7 > 2 vrai
Réel	Addition	+	2 + 5,4 = 7,4
	Soustraction	-	8,5 – 5 = 3,5
	Multiplication	*	2 * 4 = 8
	Division	/	10,6 / 4 = 2,65
	Comparaison	≤ ≥ > < = ≠	7 < 2 faux
Caractère	Comparaison	≤ ≥ > < = ≠	'B' < 'K' faux
Chaîne	Concaténation	& +	"ok " & " " & "by"
	Comparaison	≤ ≥ > < = ≠	'moh' < 'an ' vrai
Booléen	Logiques	et non ou	non( 7>1) faux

Chapitre 1

Les variables et les constantes

Déclaration des variables

Les opérations sur des variables

Priorité	Opérateur	Signification
La plus élevée	- (unaire)	Négation algébrique
	^	Puissance
	* / div mod	Multiplication, division, division entière et modulo
	+ -	Addition et soustraction
	&	Concaténation de chaînes
	< <= > >=	Opérateurs de comparaison
	= <>	Opérateurs d'égalité
	Non	Négation logique
	Et	Et logique
	Ou	Ou logique
La plus basse		

23

Chapitre 1

Les variables et les constantes

Déclaration des variables

Les opérations sur des variables

❑En cas de besoin, on utilise les parenthèses pour indiquer les opérations à effectuer en priorité.

❑à priorité égale, l'évaluation de l'expression se fait de gauche à droite.

Exemple :

7-5+2

7-(5+2)

7-5\*2

vaut : 4 --> ( 7-5 puis 2+2)

vaut : 0 --> (5+2 puis 7-7)

vaut: 3 --> (5\*2 puis 7-10 )

(a +b \*c ) /d \*e

1

2

3

4

24

Pr. Louzaoui Khadija

12

## Chapitre 1

## Les variables et les constantes

## Les constantes

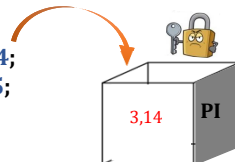
- ❑ Une **constante** est une variable dont la valeur ne doit pas changer au cours de l'exécution du programme. Par convention, on la nomme en MAJUSCULES.
- ❑ Une constante doit toujours **recevoir une valeur dès sa déclaration**, elle est caractérisée par un **identificateur** et une **valeur**
- ❑ En pseudo-code, la déclaration des constantes est effectuée après le mot clé «**constante**» ou «**const**» .

## Syntaxe :

**Constante** **identificateur** = valeur

## Exemple:

**Constante** **PI** = 3,14;  
**Constante** **COEF** = 5;



25

## Chapitre 1

## Les instructions élémentaires

- ❑ Un algorithme, par définition, est un ensemble **d'instructions** qui peuvent être simples ou complexes.
- ❑ Une **instruction** est une action élémentaire commandant à l'ordinateur un calcul, ou une communication avec l'un de ses périphériques d'entrées ou de sorties.
- ❑ Les **instructions de base** permettent la manipulation des variables telles que **l'affectation, la lecture et l'écriture**.

26

Chapitre 1

Les instructions élémentaires

Instruction d'affectation

❑ L'affectation est l'action élémentaire principale puisque c'est par son intermédiaire que l'on peut modifier la valeur d'une variable.

❑ L'affectation consiste à attribuer une valeur à une variable, elle est symbolisée en algorithmique par le signe « ← »

Var ← val

▪ val peut être une valeur, une autre variable ou une expression.

▪ Var et val doivent être de même type ou de types compatibles.

▪ l'affectation ne modifie que ce qui est à gauche de la flèche

Remarque:

❑ les déclarations suivantes sont invalides :

Var = val ou Var → val

❑ Une constante ne figure jamais à gauche d'une instruction d'affectation.

Exemple d'instruction fausse :

Constante z = 1 ;

▪ z ← 2 ; « Faux »

27

Chapitre 1

Les instructions élémentaires

Instruction d'affectation

Exemples:

5

x

Vrai

i

x ← 5

y ← x+10,5

i ← Vrai

j ← " MIP "

15,5

y

MIP

j

28

Pr. Louzaoui Khadija

14

Chapitre 1

Les instructions élémentaires

Instruction d'affectation

Après une affectation, l'ancien contenu d'une variable est substitué (écrasé) par le nouveau contenu.

3

A

0

B

Mémoire

5

+

3

8

U.C

A:Entier

B: Réel

A ← 5

B ← A+3

A ← 3

29

Chapitre 1

Les instructions élémentaires

Instruction d'affectation

Une instruction d'affectation doit se faire entre deux types compatibles.

Algorithme Affectations

Variables

i, j, k : entier;  
x, a, b, c, delta : réel;  
ok : booléen;  
ch1, ch2 : chaîne de caractères;

Début

i ← 1;  
j ← i;  
k ← i+j;  
x ← 10.3 ;  
ok ← FAUX ;  
ch1 ← "SMIA";  
ch2 ← ch1 ;  
x ← 4;  
x ← j  
delta ← b\*b - 4\*a\*c;

Fin

Exemples non valides:

i ← 10.3 ;  
OK ← "SMI";  
j ← x;  
ch1 ← delta;

30

Chapitre 1

Les instructions élémentaires

Instruction d'affectation

❑ Les langages de programmation C, Python, C++, Java, ... utilisent le signe égal = ou := pour l'affectation ←.

❑ Lors d'une affectation, l'expression de droite est évaluée et la valeur trouvée est affectée à la variable de gauche.  
Ainsi,  $A \leftarrow B$  ; est différente de  $B \leftarrow A$ ;

❑ l'affectation est différente d'une équation mathématique :

- $A+1 \leftarrow 3$  n'est pas possible en langages de programmation et n'est pas équivalente à  $A \leftarrow 2$ .
- Les opérations  $x \leftarrow x+1$  et  $x \leftarrow x-1$  ont un sens en programmation et se nomment respectivement **incrément** et **décrément**.
- Certains langages donnent des valeurs par défaut aux variables déclarées.
- Pour éviter tout problème, il est préférable d'initialiser les variables déclarées.

31

Chapitre 1

Les instructions élémentaires

Instruction d'affectation

Exercices

Faite le déroulement des algorithmes suivants en donnant la valeur finale de chaque variable :

Algorithme Test1

Variables a,b,c: booléens

Debut

a ← vrai

b ← faux

c ← b

fin

Algorithme Test2

Variables a,b,c: entiers

Debut

a ← 10

b ← 30

c ← a+b


b ← a+2

a ← a\*b

fin

32



Chapitre 1	Les instructions élémentaires
<p style="color: #A52A2A; font-weight: bold;">Les instructions d'entrée et de sortie</p> <p>L'algorithme a besoin de données en entrée, et fournit un résultat en sortie.</p> <div style="text-align: center; margin: 20px 0;">  <pre> graph LR     Entrée --&gt; Algorithme[Algorithme Suite d'instructions logiques]     Algorithme --&gt; Sortie             </pre> </div> <p>Lorsqu'on utilise un ordinateur:</p> <ul style="list-style-type: none"> <li>▪ le <b>clavier</b> permet de <b>saisir</b> les données</li> <li>▪ l'<b>écran</b> permet d'<b>afficher</b> un résultat ou des textes qui donnent des directives sur les données à fournir.</li> </ul>	
33	

Chapitre 1	Les instructions élémentaires
<p style="color: #A52A2A; font-weight: bold;">L'instruction d'entrée</p> <p>❑ <b>L'instruction de lecture</b> permet d'introduire (<b>entrer</b>) une ou plusieurs données à partir du clavier (<b>la saisie</b>), puis les sauvegarder dans leurs cases mémoires correspondantes.</p> <p>❑ L'instruction d'entrée donne la main à l'utilisateur pour <b>saisir</b> une donnée au clavier. La <b>valeur</b> saisie sera <b>affectée</b> à une <b>variable</b>.</p> <div style="background-color: #FFFF00; padding: 10px; margin: 10px 0;"> <p><i>Syntaxe :</i></p> <p style="color: #008000; font-weight: bold;">Lire ( identificateur )</p> <p style="color: #FF0000; font-weight: bold;">Lire ( identificateur1, identificateur2,...)</p> </div> <p><b>Exemple:</b></p> <ul style="list-style-type: none"> <li>▪ <b>Lire(x)</b> : <b>lit</b> et <b>stocke</b> <b>une</b> valeur donnée dans la case mémoire associée à <b>x</b>.</li> <li>▪ <b>Lire(x, y)</b> : <b>lit</b> et <b>stocke</b> <b>deux</b> valeurs, la première dans <b>x</b> et la deuxième dans <b>y</b>.</li> </ul>	
34	

Chapitre 1

Les instructions élémentaires

L'instruction d'entrée

Lire (var)

Se déroule en trois étapes :

1) Le programme s'arrête lorsqu'il rencontre une instruction Lire et ne se poursuit qu'après la saisie de l'entrée attendue par le clavier.

2) La touche Entrée signale la fin de la saisie.

3) La machine place la valeur entrée au clavier (ou saisie) dans la zone mémoire nommée var.

Mémoire

Variable	Valeur lue
x	5
y	7
...	

Lire(x, y) ;  
...

← input

57

Clavier

Côté ordinateur

Côté programmeur

35

Chapitre 1

Les instructions élémentaires

L'instruction d'entrée

Remarques:

❑ Lire une valeur ne correspondant pas au type de la variable où elle doit être stockée.

❑ Conseil: Avant de lire une variable, il est fortement conseillé d'écrire des messages à l'écran, afin de prévenir l'utilisateur de ce qu'il doit taper (sinon longue attente).

❑ Attention une constante n'est jamais lue.

36

Pr. Louzaoui Khadija

18

Chapitre 1

Les instructions élémentaires

L'instruction d'entrée

Exercices

lire(5)	Erreur
lire(R)	Lire et stocker la valeur saisie au clavier dans la variable R
lire('a')	erreur
lire(a,b)	Lire et stocker respectivement les deux valeurs saisies au clavier dans les variables a et b
Lire (a+b)	erreur
lire (x←5)	erreur
lire("le montant est ",M)	erreur

37

Chapitre 1

Les instructions élémentaires

L'instruction de sortie

❑

L'instruction d'écriture permet d'écrire en sortie (output) les données résultant d'un traitement effectué par l'algorithme (valeur, texte, ...) en les affichant par exemple sur un périphérique de sortie tel que l'écran.

❑

Cette donnée à afficher peut être :

- Un texte (un commentaire ou un message).
- Une constante.
- Le contenu d'une variable.
- Mélange de texte et de valeurs.
- Le résultat d'une expression arithmétique.
- Le résultat d'une expression logique.

Syntaxe :

Ecrire ( Expression )

38

Pr. Louzaoui Khadija

19

Chapitre 1

Les instructions élémentaires

L'instruction de sortie

Exemple :

Ecrire (" texte à afficher")

Ecrire (var)

Ecrire (var1,var2,...)

Ecrire (exp)

Ecrire (exp1,exp2,...)

Ecrire (var,exp)

Ecrire (" texte à afficher" ,var, exp)

La virgule sépare les chaines de caractères et la variable.

Tout le texte contenu entre des guillemets est écrit à l'écran, alors que lorsqu'une variable apparait dans l'instruction Ecrire c'est sa valeur qui est afficher.

Dans le cas d'écriture d'une expression, c'est le résultat d'évaluation de cette expression qui est affiché et non pas l'expression elle-même.

39

Chapitre 1

Les instructions élémentaires

L'instruction de sortie

Mémoire

Variable	Valeur lue
x	5
y	7
...	

Ecrire(x, y) ;  
...

output

5  
7

Côté ordinateur

Côté programmeur

40

Chapitre 1

Les instructions élémentaires

L'instruction de sortie

Exemple:

Ecrire(Nom)

▪Signifie : Affiche en sortie le contenu de la variable Nom

Ecrire("Entrer votre Nom : ")

▪Signifie : afficher sur l'écran le message : « Entrer votre Nom »

Ecrire("Votre Nom est : ", Nom)

▪Affiche en sortie le message : « Votre Nom est : le contenu de la variable Nom »

Ecrire(x+y)

▪Affiche en sortie le résultat d'addition de x et y  
(soient x=5 et y=7, le résultat est 12)

41

Chapitre 1

Les instructions élémentaires

L'instruction de sortie

Exercices

Algorithme	Affichage après Exécution
Ecrire(" Bonjour ")	Bonjour
Ecrire(R)	La valeur de R
Ecrire(3+2)	5
Ecrire("3+2=7")	3+2=7
Ecrire("3+2=3+2")	3+2=3+2
Ecrire("3+2=",3+2)	3+2=5
Ecrire(X+Y)	La valeur de X+Y
Ecrire("X+Y")	X+Y
Ecrire ("la somme est ",S)	La somme est : la valeur de S
Ecrire (a, b+3," message")	La valeur de a , la valeur de (a+b) , message

42

Chapitre 1

Les instructions élémentaires

L'instruction d'entrée et de sortie

→

→

→

**Remarque :**

il faut bien comprendre la lecture et l'écriture du côté de la machine et non coté utilisateur ça veut dire :

**La lecture** pour la machine c'est **l'écriture** pour l'utilisateur.

**L'écriture** pour la machine c'est **la lecture** pour l'utilisateur.

Saisir la valeur de A

Lire(A)

lecture

Écriture

Lire la valeur de A

écrire(A)

Écriture

lecture

43

Chapitre 1

Les instructions élémentaires

Les commentaires

❑

Lorsqu'un algorithme devint long, il est conseillé d'ajouter des lignes de **commentaires** dans l'algorithme, c'est-à-dire des lignes qui ont pour but de donner des indications sur les instructions effectuées et d'expliquer le fonctionnement d'algorithme (programme) sans que le compilateur ne les prenne en compte.

❑

Un **commentaire** commence toujours par deux symboles « /\* » et se termine par les symboles « \*/ » sur plusieurs lignes.

❑

Ou bien commence par le symbole « // » sur une seul ligne.

Exemple:

▪

/\* Ceci est un commentaire sur plusieurs lignes \*/

▪

// Ceci est un commentaire sur une ligne

Remarque :

Parfois on utilise les commentaires pour annuler l'action de quelques instructions dans un algorithme ou un programme au lieu de les effacer :

Variable i: Entier

// Variable j: Réel

44

Pr. Louzaoui Khadija

22

## Chapitre 1

## Les instructions élémentaires

**Exemple :**

Ecrire un algorithme qui **demande** un nombre à l'utilisateur, puis qui **calcule** et **affiche** le **carré** de ce nombre.

**Algorithme** Exe2

**Variables** nb, carr : Entier;

**Début**

**Ecrire** ("Entrez un nombre :");

**Lire** (nb);

carr ← nb \* nb;

**Ecrire** ("Son carré est : ", carr);

*/\* En fait, on pourrait tout aussi bien économiser la variable carr en remplaçant les deux avant-dernières lignes par \*/*

**Ecrire** ("Son carré est : ", nb \* nb);

**Fin**

45

## Chapitre 1

## Les instructions conditionnelles

**Les structures alternatives**

❑ Contrairement au traitement séquentiel, **la structure alternative** ou **conditionnelle** permet **d'exécuter** ou non une série d'instructions selon la valeur d'une **condition**.

❑ Une **condition** est une expression logique ou une variable logique évaluée à **Vrai** ou **Faux**.

▪ Si la condition est **vérifiée** (sa valeur est : **Vrai**).

▪ Si la condition est **n'est pas vérifiée** (sa valeur est : **Faux**).

❑ Une **condition** est une **comparaison**, elle est composée de trois éléments :

▪ une valeur

▪ un opérateur de comparaison

▪ une autre valeur

Les valeurs peuvent être a priori de n'importe quel type (numériques, caractères...). Mais si l'on veut que la comparaison ait un sens, il faut que les deux valeurs de la comparaison soient du **même** type !

46

Chapitre 1	Les instructions conditionnelles
<p><b>Structure alternative simple ( un choix)</b></p> <p>Dans cette forme, une <b>action</b> qui correspond à une ou plusieurs <b>instructions</b>, est <b>exécuté</b> <b>si</b> une <b>condition</b> est <b>vérifiée</b>. <b>Sinon</b> l'algorithme passe directement au bloc d'instruction qui suit immédiatement le bloc conditionnel.</p> <p><u><b>Syntaxe :</b></u></p> <div style="background-color: yellow; padding: 10px; margin: 10px auto; width: fit-content; text-align: center;"> <b>Si Condition Alors</b>        Instructions  <b>Fin Si</b> </div> <p><b>Remarque :</b>        La condition évaluée après l'instruction « <b>Si</b> » est <b>une variable</b> ou <b>une expression booléenne</b> qui, à un moment donné, est <b>Vraie</b> ou <b>Fausse</b>. par exemple : <math>x=y</math> ; <math>x \leq y</math> ; ...</p>	
47	

Chapitre 1	Les instructions conditionnelles
<p><b>Structure alternative simple ( un choix)</b></p> <p><u><b>Exemple:</b></u></p> <pre style="margin-left: 40px;"> x ← 5 y ← 9 <b>Si (x = y) Alors</b>   Ecrire ("x est égale à y") <b>FinSi</b>           </pre> <p>le message « x est égale à y » ne sera pas affiché puisque la condition (<math>x = y</math>) <b>n'est pas vérifiée</b>.</p>	
48	



Chapitre 1

Les instructions conditionnelles

Structure alternative complète ( deux choix)

Cette forme permet de choisir entre deux actions selon qu’une condition est vérifiée ou non.

**Syntaxe :**

Si Condition Alors

Instructions 1

Si non

Instructions 2

Fin Si

Si la condition est **vraie** alors le bloc **d’instructions1** sera **exécuté**, et le bloc d’instructions2 sera **ignoré**, **sinon** le bloc **instructions2** sera **exécuté** et le bloc d’instructions1 sera **ignoré**.

49

Chapitre 1

Les instructions conditionnelles

Structure alternative complète ( deux choix)

**Exemple :**

x ← 5

y ← 9

Si (x = y) Alors

Ecrire ("x est égale à y")

Sinon

Ecrire ("x est différente de y")

FinSi

On peut traiter les deux cas possibles. Si la condition (x=y) est **vérifiée**, le **premier message est affiché**, si elle n’est **pas vérifiée**, le **deuxième message est affiché**.

50

Chapitre 1	Les instructions conditionnelles
<p><b>Structure alternative imbriquée ( multiple choix)</b></p> <p>❑ La forme complète permet de choisir entre plusieurs actions en imbriquant des formes simples selon la syntaxe suivante:</p> <p><b>Syntaxe :</b></p> <pre style="background-color: #ffffcc; padding: 10px; margin: 10px 0;"> <b>Si condition1 Alors</b>   Instructions 1 <b>Sinon</b>   <b>Si condition2 Alors</b>     Instructions 2   <b>Sinon</b>     Instructions 3   <b>Fin si</b> <b>Fin si</b></pre>	
51	

Chapitre 1	Les instructions conditionnelles
<p><b>Structure alternative imbriquée ( multiple choix)</b></p> <p><b>Exemple :</b></p> <pre style="background-color: #ffffcc; padding: 10px; margin: 10px 0;"> Algorithme Nature_Nombre variables   n: réel Début   Ecrire("Algorithme qui détermine la nature d'un nombre:")   Ecrire("Veuillez entrer un nombre :")   Lire n   <b>Si (n &gt; 0) alors</b>     Ecrire("Ce nombre est positif")   <b>Sinon</b>     <b>Si (n = 0) alors</b>       Ecrire("Ce nombre est nul")     <b>Sinon</b>       Ecrire("Ce nombre est négatif")     <b>FinSi</b>   <b>FinSi</b> Fin</pre>	
52	

Chapitre 1	Les instructions conditionnelles
<p><b>Structure alternative imbriquée ( multiple choix)</b></p> <p>L'utilisation de tests imbriqués permet de :</p> <ul style="list-style-type: none"> <li>❑ <b>Simplifier le (pseudo-code) :</b> à travers l'imbrication nous n'avons utilisé que deux conditions simples au lieu de trois conditions dont une est composée.             <ul style="list-style-type: none"> <li>→ Un algorithme (ou programme) plus simple et plus lisible.</li> </ul> </li> <li>❑ <b>Optimiser le temps d'exécution :</b> dans le cas où la première condition est vérifiée, l'algorithme passe directement à la fin, sans tester le reste qui est forcément faux.             <ul style="list-style-type: none"> <li>→ Un algorithme (ou programme) plus performant à l'exécution.</li> </ul> </li> </ul>	
53	

Chapitre 1	Les instructions conditionnelles
<p><b>Structure alternative imbriquée ( multiple choix)</b></p> <p>→ Les tests peuvent avoir un degré quelconque d'imbrications</p> <div style="background-color: #ffffcc; padding: 10px; margin: 10px 0;"> <pre> Si (condition1) Alors instruction(s) 1 Sinon     Si (condition2) Alors instruction(s) 2     Sinon         Si (condition3) Alors instruction(s) 3         ...         Sinon instruction(s) N     FinSi FinSi FinSi           </pre> </div> <p><b>Conseil :</b>          utiliser les tests imbriqués pour <b>limiter le nombre de tests</b> et <b>placer d'abord les conditions les plus probables</b>.</p>	
54	

Chapitre 1	Les instructions conditionnelles
<p><b>Conditions composées</b></p> <p>Certains problèmes exigent parfois de formuler des conditions qui ne peuvent pas être exprimées sous la forme d'une simple comparaison, par exemple:</p> <ul style="list-style-type: none"> <li>❑ La condition <math>x \in [0, 1[</math> s'exprime par la combinaison de deux conditions <b><math>x \geq 0</math> ET <math>x &lt; 1</math></b> qui doivent être vérifiées en même temps.</li> <li>❑ le cas « <b>A est inclus entre 5 et 8</b> », A une variable de type entier, elle revient à dire que « <b>A est supérieur à 5 ET A est inférieur à 8</b> ».</li> </ul> <p>Il y a donc bien là deux conditions, reliées par ce qu'on appelle un <b>opérateur logique</b>.</p>	
55	

Chapitre 1	Les instructions conditionnelles
<p><b>Conditions composées</b></p> <ul style="list-style-type: none"> <li>❑ Une condition <b>composée</b> est une condition formée de plusieurs conditions <b>simples reliées</b> par des <b>opérateurs logiques</b>: <b>ET</b>, <b>OU</b>, <b>OU exclusif</b> (XOR) et <b>NON</b>.</li> <li>❑ Ordre de priorité des opérateurs logiques             <ol style="list-style-type: none"> <li>1. NON</li> <li>2. ET</li> <li>3. OU</li> <li>4. OU exclusif</li> </ol> </li> </ul> <p><b>Exemples :</b></p> <ul style="list-style-type: none"> <li>• x compris entre 2 et 6 : <b><math>(x \geq 2)</math> ET <math>(x \leq 6)</math></b></li> <li>• n divisible par 3 ou par 2 : <b><math>(n \% 3 == 0)</math> OU <math>(n \% 2 == 0)</math></b></li> <li>• deux valeurs et deux seulement sont identiques parmi a, b et c : <b><math>(a=b)</math> XOR <math>(a=c)</math> XOR <math>(b=c)</math></b></li> <li>• <math>(A = \text{faux}) \Leftrightarrow \text{non } A</math></li> <li>• <math>(A = \text{vrai}) \Leftrightarrow A</math></li> </ul>	
56	

Chapitre 1

Les instructions conditionnelles

Conditions composées

Tables de vérité

L'évaluation d'une condition composée se fait selon des règles présentées généralement dans ce qu'on appelle tables de vérité.

C1	C2	C1 ET C2
VRAI	VRAI	VRAI
VRAI	FAUX	FAUX
FAUX	VRAI	FAUX
FAUX	FAUX	FAUX

C1	C2	C1 OU C2
VRAI	VRAI	VRAI
VRAI	FAUX	VRAI
FAUX	VRAI	VRAI
FAUX	FAUX	FAUX

C1	C2	C1 XOR C2
VRAI	VRAI	FAUX
VRAI	FAUX	VRAI
FAUX	VRAI	VRAI
FAUX	FAUX	FAUX

C1	NON C1
VRAI	FAUX
FAUX	VRAI

57

Chapitre 1

Les instructions conditionnelles

Conditions composées

Nous avons les équivalences suivantes :

▪ **NON (A ET B) ⇔ NON A OU NON B**

▪ **NON (A OU B) ⇔ NON A ET NON B**

Ainsi, toute structure de test avec l'opérateur logique **ET** peut être exprimée d'une manière équivalente avec l'opérateur logique **OU** et vice-versa. Par conséquent, les deux alternatives suivantes sont équivalentes :

Si **A ET B** Alors

Instructions 1

Sinon

Instructions 2

Fin si

↔

Si **Non A OU Non B** Alors

Instructions 1

Sinon

Instructions 2

Fin si

58

Pr. Louzaoui Khadija

29

Chapitre 1

Les instructions conditionnelles

Structure sélective ( Conditionnelle à choix multiples)

Exemple:

Algorithme jours

variables j: entier

Début

Ecrire("Entrer un nombre entre 1 et 7:")

lire j

Si (j = 7) alors

écrire("Dimanche")

Sinon

Si (j = 6) alors

écrire("Samédi")

Sinon

Si (j = 5) alors

écrire("Vendredi")

Sinon

Si (j = 4) alors

écrire("Jeudi")

Sinon

Si (j = 3) alors

écrire("Mercredi")

Sinon

Si (j = 2) alors

écrire("Mardi")

Sinon

Si (j = 1) alors

écrire("Lundi")

FinSi

FinSi

FinSi

FinSi

FinSi

Fin

59

Chapitre 1

Les instructions conditionnelles

Structure sélective ( Conditionnelle à choix multiples)

➤Plusieurs conditions à traiter dans un algorithme.

➤Comparer une même variable avec plusieurs valeurs.

➤Imbrication des alternatives devient importante.

⇓

La **structure sélective** est la solution la plus appropriée car elle est plus facile à représenter.

➤ La **structure sélective** est une représentation simplifiée des conditions imbriquées (au lieu d'exprimer beaucoup d'imbrications de conditions Si - Sinon).

➤La **structure sélective**

➤ou **structure conditionnelle à choix multiples**

➤ou encore **structure Selon**

➤appelée parfois **structure Cas**

Pr. Louzaoui Khadija

30

Chapitre 1

Les instructions conditionnelles

Structure sélective ( Conditionnelle à choix multiples)

Syntaxe :

Selon sélecteur Faire

valeur 1 : Traitement 1

valeur 2 : Traitement 2

valeur 3 : Traitement 3

valeur 4 : Traitement 4

...

valeur n : Traitement n

Sinon : autre Traitement

FinSelon

ou

Cas sélecteur Vaut

valeur 1 : Traitement 1

valeur 2 : Traitement 2

valeur 3 : Traitement 3

valeur 4 : Traitement 4

...

valeur n : Traitement n

Sinon : autre Traitement

FinCas

61

Chapitre 1

Les instructions conditionnelles

Structure sélective ( Conditionnelle à choix multiples)

Exemple:

Algorithme jours

variables j: entier

Début

Ecrire("Algorithme qui affiche le jour :")

Ecrire("Veuillez entrer un nombre entre 1 et 7:")

Lire j

selon j faire

1:Ecrire("lundi")

2:Ecrire("Mardi")

3:Ecrire("Mercredi")

4:Ecrire("Jeudi")

5:Ecrire("Vendredi")

6:Ecrire("Samedi")

7:Ecrire("Dimanche")

Sinon: écrire("Entrer un nombre entre 1 et 7")

FinSelon

Fin

62

Pr. Louzaoui Khadija

31

Chapitre 1	Les instructions conditionnelles
<p><b>Exercice:</b></p> <p>1- Ecrire un algorithme qui permet de lire deux variables numériques a et b, de les afficher avant et après leur permutation. Par exemple, avant : a=15 et b=6, après : a=6 et b=15.</p> <p>2- Proposer un algorithme qui réalise la permutation de deux variables numériques sans avoir utiliser une troisième variable.</p> <p>3- Ecrire un algorithme qui permet la lecture de trois entiers a, b et c et qui détermine le minimum des trois nombres.</p> <p>4- Ecrire un algorithme qui permet la résolution de l'équation <math>ax+b=0</math>.</p> <p>5- Ecrire un algorithme qui permet de résoudre l'équation du deuxième degré <math>ax^2+bx+c=0</math>.</p>	
63	

Chapitre 1	Les instructions itératives
<p><b>Boucle</b></p> <p><input type="checkbox"/> Dans un algorithme on a souvent besoin de répéter un même bloc d'instructions plusieurs fois.</p> <p><input type="checkbox"/> Au lieu d'effectuer cette répétition manuellement on utilise les structures itératives ou <b>répétitives (boucles)</b>.</p> <p><input type="checkbox"/> Une <b>boucle</b> (ou <b>itération</b>) est une instruction de contrôle qui permet de répéter plusieurs fois un ensemble d'instructions.</p> <p><input type="checkbox"/> Les boucles participent à ce qu'on appelle la <b>factorisation</b> du code. Elles permettent de n'écrire qu'une fois un morceau d'algorithme qui peut néanmoins être exécuté plusieurs fois.</p>	
64	



Chapitre 1

Les instructions itératives

Boucle

❑ En algorithmique il existe trois types principaux de structures répétitives à savoir:

▪ la boucle **Pour** qui permet de répéter une instruction un certain nombre de fois.

▪ la structure **Tant que ... Faire** qui permet d'effectuer une instruction tant qu'une **condition** est satisfaite.

▪ la boucle **Répéter ... jusqu'à** qui permet de répéter une instruction jusqu'à ce qu'une **condition** soit satisfaite.

Nombre de répétitions est connu

→

Boucle **Pour**

Nombre de répétitions dépend d'une condition

→

→

Boucle **Tant que**

Boucle **Répéter ... Jusqu'à ...**

Chapitre 1

Les instructions itératives

Structure POUR

❑ La boucle **Pour** permet d'exécuter une séquence d'instructions **un nombre de fois connu fixé** à l'avance.

❑ Elle utilise une variable **indice (compteur)** de contrôle d'itérations caractérisé par: **valeur initiale**, **valeur finale** , **pas** de variation

Syntaxe :

**Pour** **indice** variant de **initial** à **final** **pas valP**

Instructions

**FinPour**

❑ **indice** : est une variable compteur de **type entier** (ou caractère). Elle doit être **déclarée**.

❑ **valP** : est un **entier** qui peut être **positif** ou **négatif**.

▪ **pas** peut ne pas être mentionné, car par défaut sa valeur est égal à **1**. Dans ce cas

▪ **le nombre d'itérations** est égal à ( **final - initial+1** ).

❑ **initial** et **final** peuvent être des **valeurs**, des **variables** définies avant le début de la boucle ou des **expressions** de même type que **indice**.

66

Pr. Louzaoui Khadija

33

Chapitre 1	Les instructions itératives
<p><b>Structure POUR</b></p> <p><u><b>Exemple</b></u></p> <div style="display: flex; align-items: center; justify-content: space-around;"> <div style="background-color: #ffffcc; padding: 10px; border: 1px solid black; width: 40%;"> <p><b>Algorithme Message</b></p> <p><b>Début</b></p> <p>    écrire("Bonjour")</p> <p>    écrire("Bonjour")</p> <p>    écrire("Bonjour")</p> <p>    écrire("Bonjour")</p> <p>    écrire("Bonjour")</p> <p>    écrire("Bonjour")</p> <p>    écrire("Bonjour")</p> <p>    écrire("Bonjour")</p> <p>    écrire("Bonjour")</p> <p><b>Fin</b></p> </div> <div style="font-size: 2em; color: #f4a460;">➔</div> <div style="background-color: #ffffcc; padding: 10px; border: 1px solid black; width: 40%;"> <p><b>Algorithme Message</b></p> <p><b>Variable i:entier</b></p> <p><b>Début</b></p> <div style="border: 1px solid red; padding: 2px; margin: 5px 0;"> <b>Pour i de 0 à 9 pas 1</b> </div> <p>    écrire("Bonjour")</p> <p><b>FinPour</b></p> <p><b>Fin</b></p> </div> </div>	

67

Chapitre 1	Les instructions itératives
<p><b>Structure POUR</b></p> <p><u><b>Remarques</b></u></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Il faut éviter de modifier la valeur du <b>compteur (indice)</b> (et de <b>final</b>) à l'intérieur de la boucle.</li> <li><input type="checkbox"/> En effet, une telle action : <ul style="list-style-type: none"> <li><input type="checkbox"/> perturbe le nombre d'itérations prévu par la boucle Pour</li> <li><input type="checkbox"/> rend difficile la lecture de l'algorithme</li> <li><input type="checkbox"/> présente le risque d'aboutir à une boucle infinie</li> </ul> </li> </ul> <p><b>Exemple :</b></p> <div style="background-color: #ffffcc; padding: 10px; border: 1px solid black; margin: 10px auto; width: 60%;"> <p><b>Pour i de 1 à 5 pas 1</b></p> <p>    i ← i-1;</p> <p>    écrire(" i = ", i);</p> <p><b>FinPour</b></p> </div>	

68

Chapitre 1

Les instructions itératives

Structure Tantque ... Faire

❑ La boucle **Tant que ... Faire**, permet de **tester une condition** et **répéter** le traitement associé tant que cette condition est **vérifiée**.

❑ Une fois cette condition là devient **fausse** alors on **quitte** la boucle pour **poursuivre** l'exécution du reste du traitement.

Syntaxe :

Tantque condition Faire

Instructions

Fintantque

69

Chapitre 1

Les instructions itératives

Structure Tant que ... Faire

Remarques:

▪ Il est possible que les instructions à **répéter** ne soient jamais exécutées.

▪ Le nombre **d'itérations** dans une boucle TantQue **n'est pas connu** au moment d'entrée dans la boucle. Il **dépend** de l'évolution de la valeur de la condition.

▪ Une des instructions du corps de la boucle doit absolument **changer la valeur de la condition** de **vrai** à **faux** (après un certain nombre d'itérations), sinon le programme va tourner indéfiniment (boucles infinies).

70

Chapitre 1

Les instructions itératives

Structure Tant que ... Faire

Exemple ( Message )

Algorithme Message

Début

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

Fin

?????????????

Boucles infinies

Condition

Point de Sortie de la boucle

Algorithme Message

Début

Tantque Vrai Faire

écrire("Bonjour")

FinTantque

Fin

71

Chapitre 1

Les instructions itératives

Structure Tant que ... Faire

Exemple ( Message )

Algorithme Message

Début

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

écrire("Bonjour")

Fin

Déclarer un Compteur

Initialiser le Compteur

{

Vérifier la condition

Incrémenter du Compteur

}

Répéter 10 fois

Sortir de la boucle

Algorithme Message

Variable i:entier

Début

i ← 0

Tantque i < 10 Faire

écrire("Bonjour")

i ← i+1

FinTantque

Fin

72

Chapitre 1

Les instructions itératives

Structure Tant que ... Faire

Exemple

Algorithmme Message  
Variable **i:entier**  
Début  
**i ← 0**  
Tantque **i < 10** Faire  
    écrire("Bonjour")  
    **i ← i+1**  
FinTantque  
Fin

	<b>i ← 0</b>	Ecrire("Bonjour")	<b>i ← i+1</b>
Itération 1:	i 0	Bonjour	i 1
Itération 2:	i 1	Bonjour	i 2
Itération 3:	i 2	Bonjour	i 3
Itération 4:	i 3	Bonjour	i 4
...	...	...	...
Itération 10:	i 9	Bonjour	i 10
Fin tanque	i 10	Point de sortie la boucle	

73

Chapitre 1

Les instructions itératives

Structure Tant que ... Faire

Exemple

Algorithmme Message  
Variable **i:entier**  
Début  
**i ← 0**  
Tantque **i < 10** Faire  
    écrire("Bonjour")  
    **i ← i+1**  
FinTantque

Algorithmme Message  
Variable **i:entier**  
Début  
**i ← 1**  
Tantque **i ≤ 10** Faire  
    écrire("Bonjour")  
    **i ← i+1**  
FinTantque  
Fin

Tantque **i > 0** Faire  
    écrire("Bonjour")  
    **i ← i-1**  
FinTantque  
Fin

Tantque **i ≥ 0** Faire  
    écrire("Bonjour")  
    **i ← i-1**  
FinTantque  
Fin

74

Pr. Louzaoui Khadija

37

Chapitre 1	Les instructions itératives
<p><b>Structure Tant que ... Faire</b></p> <p><u><b>Boucles Infinie</b></u></p> <div style="display: flex; justify-content: space-around; margin-top: 20px;"> <div style="background-color: #ffffcc; padding: 10px; border: 1px solid black; width: 45%;"> <pre> <b>Algorithme</b> Message <b>Variable</b> <i>i</i>:entier <b>Début</b> <i>i</i> ← 0 <b>Tantque</b> <i>i</i> &lt; 10 <b>Faire</b>     écrire("Bonjour") <b>FinTantque</b> <b>Fin</b> </pre> </div> <div style="background-color: #ffffcc; padding: 10px; border: 1px solid black; width: 45%;"> <pre> <b>Algorithme</b> Message <b>Variable</b> <i>i</i>:entier <b>Début</b> <i>i</i> ← 1 <b>Tantque</b> <i>i</i> &gt; 0 <b>Faire</b>     écrire("Bonjour")     <i>i</i> ← <i>i</i>+1 <b>FinTantque</b> <b>Fin</b> </pre> </div> </div>	

75

Chapitre 1	Les instructions itératives
<p><b>Structure Tant que ... Faire</b></p> <p><u><b>Exemple ( contrôle de saisie )</b></u></p> <p>Contrôle de <b>saisie</b> d'une <b>lettre</b> alphabétique jusqu'à ce que le <b>caractère</b> <b>entré</b> soit <b>valable</b></p> <div style="background-color: #ffffcc; padding: 10px; border: 1px solid black; margin-top: 20px;"> <pre> <b>Algorithme</b> Contrôle_saisie <b>variables</b> <i>C</i> : Caractère <b>Début</b>     écrire("Entrer une lettre :")     lire <i>C</i> <b>Tantque</b> ((<i>C</i> &lt; 'A') ou (<i>C</i> &gt; 'Z')) <b>Faire</b>     écrire ("Saisie erronée. Recommencez")     Lire <i>C</i> <b>FinTantque</b>     écrire ("La saisie est valable ") <b>Fin</b> </pre> </div>	

76

Chapitre 1

Les instructions itératives

Lien entre Pour et Tant que ... Faire

❑ La boucle **Pour** est un cas particulier de **Tantque** (cas où le nombre d'itérations est connu et fixé).

❑ Tout ce qu'on peut écrire avec **Pour** peut être remplacé avec **Tantque** (**la réciproque est fausse**)

**Pour** **i** de **initial** à **final** **pas** **1**

Instructions

**FinPour**

**i** ← **initial**

**Tantque** **i** <= **final** **Faire**

Instructions

**i** ← **i+1**

**FinTantque**

77

Chapitre 1

Les instructions itératives

Lien entre Pour et Tant que ... Faire

Exemple ( Puissance )

un algorithme qui permet de calculer et d’affiche  $x^n$  (  $x$  est un réel ,  $n$  est un entier positive).

Algorithme puissance

variables  $x, p$  : réel

$n, i$  : entier

Début

écrire(" Donner la base x:" )

lire  $x$

écrire(" Donner l'exposant n:" )

lire  $n$

$p \leftarrow 1$

**Pour** **i** de **1** à **n** **pas** **1**

$p \leftarrow p * x$

**FinPour**

écrire( $x$ , " à la puissance ",  $n$ , " est:",  $p$  )

**Fin**

$p \leftarrow 1$

$i \leftarrow 1$

**Tantque** ( $i \leq n$ ) **Faire**

$p \leftarrow p * x$

$i \leftarrow i + 1$

**FinTantque**

78

Pr. Louzaoui Khadija

39

Chapitre 1

Les instructions itératives

Structure Répéter ... Jusqu'à ...

❑ La boucle **Répéter... Jusqu'à...** , permet de **répéter** un bloc d'instructions jusqu'à ce qu' une **condition** soit **vérifiée**.

Syntaxe :

Répéter

Instructions

Jusqu'à condition

▪ La **vérification** de la **condition** s'effectue après l'exécution des instructions.

▪ les **instructions** entre **Répéter** et **jusqu'a** sont **exécutées au moins une fois** et leur exécution est répétée jusqu'à ce que la **condition** devienne **fausse**.

79

Chapitre 1

Les instructions itératives

Structure Répéter ... Jusqu'à ...

Exemple

Ecrire un algorithme qui permet de demander à l'utilisateur un nombre N et qui calcule la somme des N entiers positifs .  
Par exemple, si l'utilisateur entre le nombre 5, l'algorithme doit calculer 1+2+3+4+5 .

Algorithme Message

Variable N,s,i:entier

Début

Écrire ("Donner N:")

Lire N

**i ← 1**

**s ← 0**

Répéter

**s ← s+i**

**i ← i+1**

Jusqu'à (**i > N**)

Écrire ("la somme est :",s)

Fin

Déclarer un Compteur

**Initialiser le Compteur**

Exécuter le traitement pour la 1 fois

Exécuter le traitement à répéter

Incrémenter du Compteur

Vérifier la condition

**Répéter N fois**

Sortir de la boucle

80

Pr. Louzaoui Khadija

40



Chapitre 1	Les instructions itératives
<p style="color: #A52A2A; margin: 0;">Structure Répéter ... Jusqu'à ...</p> <p style="margin: 0;"><u>Exemple ( Division de deux nombres )</u></p> <div style="background-color: #FFFFE0; padding: 10px; margin: 10px 0;"> <pre> Algorithme Division variables a,b,c  : réel Début     écrire(" Entrer le nombre a : " )     lire a     Répéter     écrire(" Donner le nombre b:" )     lire b     Jusqu'à (b &lt;&gt; 0)     c ← a/b     écrire(" à la résultat de la division est:",c) Fin </pre> </div>	
81	

Chapitre 1	Les instructions itératives
<p style="color: #A52A2A; margin: 0;">Tantque ... Faire et Répéter... jusqu'à...</p> <p style="margin: 0;">Différences entre les boucles Tant que... Faire et Répéter ... Jusqu'à...</p> <ul style="list-style-type: none"> <li>▪ La séquence d'instructions est exécutée au moins <b>une fois</b> dans la boucle <b>Répéter ... Jusqu'à...</b>, alors qu'elle peut ne pas être exécutée dans le cas du <b>Tantque ... Faire</b>.</li> <li>▪ Dans les deux cas, la séquence d'instructions est exécutée si la <b>condition</b> est <b>vraie</b>.</li> <li>▪ Dans les deux cas, la séquence d'instructions doit nécessairement <b>faire évoluer</b> la <b>condition</b>, faute de quoi on obtient une boucle infinie.</li> </ul>	
82	

Chapitre 1

Les instructions itératives

Tant que ... Faire et Répéter... jusqu'à...

Exemple

Ecrire un algorithme qui demande à l'utilisateur de saisir un entier supérieur strictement à 1. et qui calcule la somme des entiers jusqu'à ce nombre.  
Par exemple, si l'utilisateur entre le nombre 5, l'algorithme doit calculer 1+2+3+4+5 .

Algorithme Message

Variable N,s,i:entier

Début

Écrire ("Donner N:")

Lire N

i ← 1

s ← 0

Répéter

s ← s+i

i ← i+1

Jusqu'à (i > N)

Écrire ("la somme est :",s)

Fin

i ← 1

s ← 0

Tantque (i<=N) Faire

s ← s+i

i ← i+1

FinTantque

83

Chapitre 1

Les instructions itératives

Notion du compteur

- Un **compteur** est une variable associée à la boucle dont la valeur est incrémentée de un à chaque itération. Elle sert donc à **compter le nombre d'itérations** (répétitions) de la boucle.
- La notion du **compteur** est associée particulièrement aux deux boucles : « Répéter...jusqu'à » et « Tant que...faire ». Par contre, dans la boucle « Pour », c'est l'indice qui joue le rôle du compteur.
- L'utilisation du **compteur** dans les deux premières boucles est exprimée ainsi :

Bloc de la boucle

compt ← 0

Répéter

Instructions

...

compt ← compt+1

Jusqu'à condition

compt ← 0

Tantque condition Faire

Instructions

...

compt ← compt+1

FinTantque

84

Pr. Louzaoui Khadija

42

Chapitre 1

Les instructions itératives

Notion du compteur

Remarque :

▪ Il faut toujours initialiser le **compteur** avant de commencer le comptage.

▪ La variable « **compt** » a été initialisée à zéro (0) avant le début de chaque boucle.

▪ L'instruction « **compt ← compt +1** » incrémente la valeur de « **compt** » de un (1). Elle peut être placée n'importe où à l'intérieur du bloc de la boucle.

Exemple :

compt ← 0

Répéter

Ecrire(i)

compt ← compt+1

Jusqu'à (i=5)

Résultat d'exécution : 0,1,2,3,4

compt ← 0

Tantque (i<5) Faire

Ecrire(i)

compt ← compt+1

FinTantque

Résultat d'exécution : 0,1,2,3,4

85

Chapitre 1

Les instructions itératives

Boucles imbriquées

▪ Les boucles peuvent être **imbriquées** les unes dans les autres. Deux ou plusieurs boucles imbriquées peuvent être aussi les mêmes ou différentes.

Exemples :

Pour i de 1 à 2 pas 1

Ecrire("i=",i)

Pour j de 1 à 3 pas 1

Ecrire("j=",j)

FinPour

FinPour

boucle 2

boucle 1

Résultat d'exécution :

i=1

j=1

j=2

j=3

i=2

j=1

j=2

j=3

Algorithme boucle\_imbriquée

Variables i,j:entier

Début

Pour i de 1 à 5 pas 1

Pour j de 1 à i

écrire("O")

FinPour

écrire("K")

FinPour

Fin

Résultat d'exécution :

OK

OOK

OOOK

OOOOK

OOOOK

86

Pr. Louzaoui Khadija

43

Chapitre 1

Les instructions itératives

Choix d'un type de boucle

- Si on peut déterminer le **nombre d'itérations** avant l'exécution de la boucle, il est plus naturel d'utiliser la boucle **Pour**.
- S'il **n'est pas** possible de **connaître** le nombre d'itérations avant l'exécution de la boucle, on fera appel à l'une des boucles **Tant que... Faire** ou **Répéter ... Jusqu'à...**

Nombre d'itérations connu?

Oui

Pour

Non

Traitement exécuté au moins une fois

Oui

Répéter ...Jusqu'a...

Non

Tantque... Faire

87

Chapitre 1

Les tableaux

Activité

❑ Supposons qu'on veut calculer la moyenne des notes d'une classe de 5 étudiants :

J'ai 5 Etudiants

Stocker la note de chaque étudiants

Calculer la moyenne

Etudiant	Note
Amina	10,5
Ahmed	9
Meryem	14
Mouad	15
Rachid	2


88

Chapitre 1

Les tableaux

Activité

Jusqu'ici, nous avons employé des **variables** pour **stocker** une seule valeur de types primitifs.



Stocker la note des 5 étudiants

Etudiant	Variable	Note
Amina	N1	10,5
Ahmed	N2	9
Meryem	N3	14
Mouad	N4	15
Rachid	N5	2

89


Chapitre 1

Les tableaux

Activité

Imaginons maintenant le cas pour une promotion de 320 étudiants, Vous pouvez remarquer que c'est un peu **lourd** de déclarer, manipuler une centaine de variables (avec 320 fois de lecture/écriture distinctes) et calculer la moyenne donnera obligatoirement une atrocité du genre :

Moyenne ← (N1+N2+N3+...+N319+N320)/320



J'ai 320 étudiants

Stocker la note des 320 étudiants???

Etudiant	Variable	Note
Amina	N1	10,5
Ahmed	N2	9
Meryem	N3	14
...	...	...
Mouad	N319	15
Rachid	N320	2

90

Chapitre 1	Les tableaux
<p><b>Activité</b></p> <p>❑ Jusqu'à présent, le seul moyen pour le faire:</p> <ul style="list-style-type: none"> <li>• <b>Déclarer</b> 320 variables désignant les notes <b>N1, ..., N320</b>.</li> <li>• La <b>saisie</b> de ces notes nécessite 320 instructions <b>lire(Ni)</b>.</li> <li>• Le <b>calcul</b> de la moyenne est : <b>l'addition</b> de 320 notes diviser par 320</li> <li>• Le calcul du nombre des <b>notes &gt; 10</b> se fait par une suite de tests de 320 instructions <b>Si</b> :           <ul style="list-style-type: none"> <li>▪ <math>\text{nbre} \leftarrow 0</math></li> <li>▪ <b>Si</b> (<math>N1 &gt; 10</math>) <b>alors</b> <math>\text{nbre} \leftarrow \text{nbre} + 1</math> <b>FinSi</b></li> <li>▪ ...</li> <li>▪ <b>Si</b> (<math>N320 &gt; 10</math>) <b>alors</b> <math>\text{nbre} \leftarrow \text{nbre} + 1</math> <b>FinSi</b></li> </ul> </li> </ul> <p style="text-align: center; color: red; font-weight: bold; margin: 10px 0;">Cette façon n'est pas très pratique</p> <div style="text-align: center; margin: 10px 0;"> </div> <p style="text-align: center;">C'est pourquoi la notion de <b>tableau</b> a été alors inventée. <small>91</small></p>	

Chapitre 1	Les tableaux
<p><b>Tableaux</b></p> <p>❑ En algorithmique (et en programmation), on peut <b>regrouper</b> toutes ces <b>variables</b> en une <b>seule structure de donnée</b> qui s'appelle <b>tableau</b>.</p> <p>❑ Un <b>tableau</b> est un ensemble de <b>variables</b> de <b>même type</b> ayant toutes le même <b>identificateur</b>.</p> <p style="text-align: center; font-weight: bold; margin: 10px 0;">Comment peut-on différencier entre des variables ayant le même nom ?</p> <div style="text-align: center; margin: 10px 0;"> </div> <p style="text-align: center; font-weight: bold; margin: 10px 0;">Chaque élément du <b>tableau</b> est repéré par un <b>indice</b>. Ce dernier est un numéro qui permet de différencier chaque élément du tableau des autres.</p> <p>❑ Les éléments du <b>tableau</b> ont tous le même <b>nom</b>, mais pas le même <b>indice</b>. Pour accéder à un élément d'un tableau, on utilise le nom du tableau suivi de l'indice de l'élément entre crochets [...].</p> <p style="text-align: right; font-size: small; margin-top: 10px;"><small>92</small></p>	

Chapitre 1

Les tableaux

Tableaux

T

5

10

-1

19

50

indices

1

2

3

4

5

Valeurs des éléments

T[1] = 5

T[2] = 10

T[3] = -1

T[4] = 19

T[5] = 50

Remarque :

Lorsqu'un tableau est créé, il prend un espace contigu en mémoire : les cases sont les unes à la suite des autres, Il n'y a pas de « trou » au milieu.

93

Chapitre 1

Les tableaux

Types tableaux

Les éléments d'un tableau sont rangés selon un ou plusieurs axes appelés dimensions du tableau.

Tableau à une dimension.

Tableau à plusieurs dimensions.

94

Chapitre 1

Les tableaux

Tableau à une dimension

❑ Dans les **tableaux à une dimension** (qui permettent de représenter par exemple des vecteurs au sens mathématique du terme), chaque élément est accessible pour lecture ou modification par un seul **indice**.

❑ Un **tableau à une dimension** est un **vecteur** d'éléments de même **type**.

❑ Il peut être représenté sous la forme suivante :

T

T[1]	T[2]	T[3]	T[4]	...	T[n]
------	------	------	------	-----	------

▪ Dimension du tableau : **1**

▪ Taille du tableau : **n**

▪ Les **T[i]**, i=**1, 2, ..., n** doivent être de même type

95

Chapitre 1

Les tableaux

Déclaration

❑ La **déclaration** d'un tableau à une dimension s'effectue en précisant le **type** de ses éléments et sa **taille** (le nombre de ses éléments)

Syntaxe :

Tableau identificateur[**taille**] : type

identificateur

--	--	--	--	--	--

Indice

123

Taille

❑ Exemple :

Tableau **note[20] : réel**

Note

10,5	9	14	4	10	2	..	..	...	15
------	---	----	---	----	---	----	----	-----	----

Note[1]

Note[5]

Note[6]

Note[20]

96

Pr. Louzaoui Khadija

48



Chapitre 1

Les tableaux

Accéder aux éléments d'un tableau

Tableau T[10] : entier

T

5	-45	10	2	75	-12	9	60	-7	25
1	2	3	4	5	6	7	8	9	10

☐ Ce tableau est de longueur 10, car il contient 10 emplacements.

☐ Chacun des dix nombres du tableau est repéré par son rang, appelé **indice**.

☐ Pour accéder à un élément du tableau, il suffit de préciser entre crochets **l'indice de la case contenant cet élément**.

☐ Pour accéder au 5<sup>ème</sup> élément (75), on écrit : **T[5]**

97

Chapitre 1

Les tableaux

Accéder aux éléments d'un tableau

Tableau T[10] : entier

T

5	-45	10	2	75	-12	9	60	-7	25
0	1	2	3	4	5	6	7	8	9

☐ Selon les langages de programmation, le premier indice du tableau est soit 0, soit 1. Le plus souvent c'est **0**.

☐ Pour accéder au **i<sup>ème</sup>** élément, on écrit **T[i-1]** (avec 0<=i<10)

☐ Dans ce cas, **T[i]** désigne l'élément **i+1** du tableau **T**.

Une fois déclaré, un tableau peut être manipulé comme un ensemble de **variables** simples. Les trois manipulations de base sont **l'affectation**, la **lecture** et **l'écriture**.

98

Chapitre 1

Les tableaux

Affectation

L'affectation d'une valeur **v** à un élément **i** d'un tableau **T** se fait par :

$T[i] \leftarrow v$

Syntaxe d'affectation:  
**identificateur[indice] ← valeur**

**Exemple:**

- Affectation du nombre 16 au 4<sup>ème</sup> élément du tableau:

T

5	-45	10	2	75	-12	9	60	-7	25
1	2	3	4	5	6	7	8	9	10

$T[4] \leftarrow 16$

- Cette instruction a modifié le contenu du 4<sup>ème</sup> élément du tableau (16 au lieu de 2).

T

5	-45	10	16	75	-12	9	60	-7	25
1	2	3	4	5	6	7	8	9	10

- La variable **x** prend la valeur du premier élément du tableau (x vaut 5).

$x \leftarrow T[1]$

99

Chapitre 1

Les tableaux

Lecture

Il est possible aussi d'affecter des valeurs aux éléments d'un tableau par une instruction de **lecture**.

Syntaxe de lecture:  
**Lire( identificateur [indice] )**

**Exemple :**

- Utilisation de la lecture pour saisir un nombre au 6<sup>ème</sup> élément du tableau

**Lire(T[6])**

T

5	-45	10	2	75	-12	9	60	-7	25
1	2	3	4	5	6	7	8	9	10

- Cette instruction initialise le contenu du 6<sup>ème</sup> élément du tableau par la valeur 16

T

5	-45	10	16	75	16				
1	2	3	4	5	6	7	8	9	10

100

Chapitre 1

Les tableaux

Ecriture

De même que la lecture, l'écriture de la valeur d'un élément du tableau s'écritra comme suit :

Ecrire **T[i]**

Cette instruction permet d'afficher la valeur de l'élément **i** du tableau **T**.

Syntaxe d'écriture:

Ecrire( identificateur[indice] )

Exemple :

- affichage de la valeur du nombre du dernier élément du tableau

Ecrire(**T[10]**)

T

5	-45	10	2	75	-12	9	60	-7	25
1	2	3	4	5	6	7	8	9	10

- Cette instruction affiche le contenu du 10<sup>ème</sup> élément du tableau (la valeur 25).

T

5	-45	10	16	75	-12	9	60	-7	25
1	2	3	4	5	6	7	8	9	10

↑

101

Chapitre 1

Les tableaux

Tableaux et boucles

☐ Un grand **avantage** des tableaux est qu'on peut traiter les données qui y sont stockées de façon simple en utilisant des **boucles**.

☐ Les **boucles** sont extrêmement utiles pour les algorithmes associés aux tableaux, pour **parcourir** les éléments du tableau selon l'ordre croissant (ou décroissant) des indices on utilise les boucles.

☐ Le traitement de chacun des éléments étant souvent le même, seule la valeur de l'indice est amenée à changer, une **boucle** est donc parfaitement adaptée à ce genre de traitements.

102

Chapitre 1

Les tableaux

initialisation d'un tableau

❑ Pour initialiser un tableau on peut utiliser par exemple les instructions suivantes :

T1[4] = { 2, 25, -7, 3}

T2[9] = { 10, 17, -5, 43, 55, -26, 70, 88, 19 }

103

Chapitre 1

Les tableaux

Saisie et affichage des éléments d'un tableau

Algorithme Tableau

Variables **n, i : entier**

**Tableau T[100]: réel**

Début

Ecrire(" Donner la taille du tableau : ")

Lire(**n**)

Pour i de 1 à n pas 1

Ecrire("Saisie de l'élément ", i, " : ")

Lire(**T[i]**)

FinPour

Pour i de 1 à n pas 1

Ecrire ("T[" ,i,"]=", **T[i]** )

FinPour

fin

Déclaration d'un tableau avec une taille maximale de 100

Lire la taille effective du tableau

Lire les éléments du tableau un par un

Afficher les éléments du tableau un par un

104

Pr. Louzaoui Khadija

52

Chapitre 1

Les tableaux

Remarque

Ne pas confondre :

- Taille maximale : une constante ( capacité )  
Constante Max=100  
Variables tableau T[Max]: réel

- Taille effective : nombre de cases réellement utilisées lors de la manipulation du tableau (une variable)

Exemple:

Algorithme Tableau  
Variables n, i : entier  
Tableau T[100]: réel  
  
Début  
Ecrire(" Donner la taille du tableau:" )  
Lire(n)  
pour i ← 1 à n pas 1 faire  
...  
FinPour  
Fin

Taille maximale : 100  
Taille effective : n

105

Chapitre 1

Les tableaux

Exemple 1

Algorithme note  
Variables n, i : entier  
moyenne, somme :réel  
tableau Note[50]: réel  
  
Début  
Ecrire(" Entrer le nombre des notes:" )  
Lire(n)  
  
Pour i de 1 à n pas 1  
Ecrire("Donner la note de l'étudiant num ", i, ":" )  
Lire(Note[i])  
FinPour  
somme ← 0  
Pour i de 1 à n pas 1  
somme ← somme+ Note[i]  
FinPour  
Moyenne ← somme/n  
Ecrire("la moyenne des notes est :", moyenne )  
Fin

106

Chapitre 1

Les tableaux

Exemple 2

Un algorithme qui permet de calculer le nombre d'étudiants ( 20 étudiants ) ayant une note supérieure à 12 avec les tableaux.

Algorithme note

Variables **tableau Note[20]**: réel  
**nbr, i** : entier

Début

Nbr ← 0

Pour i ← 1 à 20 pas 1

Ecrire("Donner la note de l'étudiant num ", i, ":" )

Lire(**Note[i]**)

Si (**Note[i] > 12**) alors

nbr ← nbr+1

FinSi

FinPour

Ecrire("le nombre des notes supérieur à 12 est :", nbr )

Fin

107

Chapitre 1

Les tableaux


Activité

J'ai 5 étudiants  
J'enseigne 3 matières

Stocker

Moyenne

Max et Min



Etudiant	Analyse	Algèbre	Informatique
Amina	10,5	15	19
Ahmed	9	5	10
Meryem	14	10	4
Mouad	15	8	12
Rachid	2	4	13

108

Chapitre 1

Les tableaux

Activité

Tableau Analyse[5]:réel

Tableau Algèbre[5]:réel

Tableau Informatique[5]:réel

Etudiant	Analyse	Algèbre	Informatique	Moyenne
Amina	10,5	15	16	13,8
Ahmed	9	5	10	8
Meryem	14	10,5	4	9,5
Mouad	15	8	12	11,6
Rachid	2	4	13,5	6,5
Max	15	15	16	
Min	2	4	4	

L'utilité d'un **tableau à deux dimensions** réside dans la possibilité de déclarer un **seul** tableau au lieu de déclarer **plusieurs** tableaux identiques.

109

Chapitre 1

Les tableaux

Tableaux à deux dimension

Un tableau à plusieurs dimensions est une **matrice** d'éléments de même type.

❑ Les langages de programmation permettent de déclarer des tableaux dans lesquels les valeurs sont repérées par deux **indices**.

- Le premier indice représente le numéro de **ligne**
- Le deuxième indice représente le numéro de **colonne**

A

A[1,1]	A[1,2]	A[1,3]	A[1,4]	A[1,5]	A[1,6]
A[2,1]	A[2,2]	A[2,3]	A[2,4]	A[2,5]	A[2,6]
A[3,1]	A[3,2]	A[3,3]	A[3,4]	A[3,5]	A[3,6]
A[4,1]	A[4,2]	A[4,3]	A[4,4]	A[4,5]	A[4,6]
A[5,1]	A[5,2]	A[5,3]	A[5,4]	A[5,5]	A[5,6]
A[6,1]	A[6,2]	A[6,3]	A[6,4]	A[6,5]	A[6,6]

- Dimension du tableau : **2**
- Taille du tableau : **L,C**
- Les **A[i][j]**, **i=1, 2, ...,L** , **j=1, 2, ...,C** doivent être de même type

110

Pr. Louzaoui Khadija

55

Chapitre 1

Les tableaux

Déclaration

☐ En pseudo code :

**Tableau** **identificateur**[NbrLigne ] [NbrColonne] : **type**

☐ Exemple :

**Tableau** Notes[5][3] : **réel**

Notes

	1	2	3
1	10,5	15	16
2	9	5	10
3	14	10,5	4
4	15	8	12
5	2	4	13,5

111

Chapitre 1

Les tableaux

Accéder aux éléments de la matrice

☐ Les éléments sont rangées dans un tableau à deux entrées.

A

	1	2	3	4	5	6
1	15	28	44	9	90	5
2	23	20	51	12	3	19
3	36	21	60	65	18	10

☐ Ce tableau a 3 lignes et 6 colonnes.

☐ Les éléments du tableau sont repérés par leur numéro de **ligne** désignés en bleu et leur numéro de colonne désignés en **vert** .

Par exemple A[2][4] vaut 12.

A[i][j] permet d'accéder à l'élément de la matrice qui se trouve à l'**intersection** de la ligne **i** et de la colonne **j**

112

Pr. Louzaoui Khadija

56



## Les tableaux

## Accéder aux éléments de la matrice

- ❑ Selon les langages de programmation, les premiers indices de la matrice est 0, soit 1. Le plus souvent c'est 0.

<b>A</b>		0	1	2	3	4	5
	0	15	28	44	9	90	5
	1	23	20	51	12	3	19
	2	36	21	60	65	18	10

- ❑ Ce tableau a **3** lignes et **6** colonnes.
- ❑ Les éléments du tableau sont repérés par leur numéro de **ligne** désignés en bleu et leur numéro de colonne désignés en **vert** .
  - Par exemple **A[1][3]** vaut **12**.

**A[i][j]** permet d'accéder à l'élément de la matrice qui se trouve à l'**intersection** de la ligne **i+1** et de la colonne **j+1**

- ❑ Un tableau à deux dimensions est manipulé de la même façon qu'un tableau simple (à une seule dimension) que ce soit pour **l'affectation**, la **lecture** ou **l'écriture**.

113

## Les tableaux

## Affectation

- ❑ Syntaxe d'affectation:

$$A[NbrLigne][NbrColone] \leftarrow \text{valeur}$$

- Example:

- Affectation de la note 16 à l'étudiant num 4 dans la matière num 2.

**A[4][2] ← 16**

	1	2	3
<b>1</b>	10,5	15	16
<b>2</b>	9	5	10
<b>3</b>	14	10,5	4
<b>4</b>	15	<b>16</b>	12
<b>5</b>	2	4	13,5

114

Chapitre 1

Les tableaux

Lecture

Syntaxe de lecture:

**Lire (A[NbrLigne ][ NbrColone]**

Exemple :

- saisie de la note 10 à l'étudiant num 3 dans la matière num 1.

**Lire ( A[3][1] )**

1

2

3

1

2

3

4

5

**A**

1	2	3
10,5	15	16
9	5	10
<b>10</b>	10,5	4
15	16	12
2	4	13,5

115

Chapitre 1

Les tableaux

Ecriture

Syntaxe d'écriture:

**Ecrire (A[NbrLigne ][ NbrColone]**

Exemple :

- Afficher de l'étudiant num 2 dans la matière num 2.

**Ecrire ( A[2][2] )**

1

2

3

1

2

3

4

5

**A**

1	2	3
10,5	15	16
9	<b>5</b>	10
10	10,5	4
15	16	12
2	4	13,5

116

Pr. Louzaoui Khadija

58

Chapitre 1

Les tableaux

Saisie et affichage des éléments de la matrice

Algorithme Matrice

Variables **m, n, i, j** : entier  
**Tableau M[100][100]**: entier

Début

Ecrire(" Entrez le nombre de lignes de la matrice:")  
Lire(**m**)  
Ecrire(" Entrez le nombre de colonnes de la matrice:")  
Lire(**n**)

Pour i ← 1 à m pas 1  
  Pour j ← 1 à n pas 1  
    Ecrire ("Entrez l'élément : ligne ", i, " et colonne ", j)  
    Lire (**M[i][j]**)  
  FinPour  
FinPour

Pour i ← 0 à m pas 1  
  Pour j ← 0 à n pas 1  
    Ecrire ("M[" ,i, ", ",j, "]= ", **M[i][j]**)  
  FinPour  
FinPour

Fin

Déclaration d'une matrice avec une taille maximale de (100, 100)

Lire la taille effective de la matrice m et n

Lire les éléments de la matrice

Afficher les éléments de la matrice

117

Chapitre 1

Les tableaux

initialisation d'une matrice

❑ Pour initialiser une matrice on peut utiliser par exemple les instructions suivantes :

T1[3][3] = {{1,2,3}, {4,5,6}, {7,8,9}}

T2[3][3] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 }

118

Pr. Louzaoui Khadija

59

Chapitre 1

Les tableaux

Exemple

Ecrire un algorithme qui permet de demander à l'utilisateur de saisir la notes des étudiants ( 5 étudiants ) dans chaque matière ( 3 matières ).  
Cet algorithme permet de calculer et afficher la moyenne de chaque étudiant.

Etudiant	Analyse	Algèbre	Informatique	Moyenne
Amina	10,5	15	16	13,8
Ahmed	9	5	10	8
Meryem	14	10,5	4	9,5
Mouad	15	8	12	11,6
Rachid	2	4	13,5	6,5

119

Chapitre 1

Les tableaux

Algorithme Moyenne

Variables m,n,i,j : entier

moyenne, somme :réel

Tableau M[50][50]: réel

Début

Ecrire(" Entrer le nombre de lignes de la matrice:")

Lire(m)

Ecrire(" Entrer le nombre de colonnes de la matrice:")

Lire(n)

pour i ← 1 à m pas 1

pour j ← 1 à n pas 1

Ecrire ("Donner la note de l' étudiant num ; ", i, " dans la matière num : ", j)

lire (M[i][j])

FinPour

Fin pour

somme ← 0

Moyenne ← 0

pour i ← 1 à m pas 1

pour j ← 1 à n pas 1

somme← somme+ M[i][j]

FinPour

Moyenne ← somme/n

Ecrire("la moyenne de l'étudiant num : ",i, " est :", moyenne )

somme←0

FinPour

Fin

120