Algorithmique #1

Pr. Abdelali El Gourari

Introduction

Solution d'une équation du second degré: $ax^2 + bx + c = 0$

Calculer le discriminant :

Étape 1: $\Delta = b^2 - 4ac$

Déterminer le signe du discriminant :

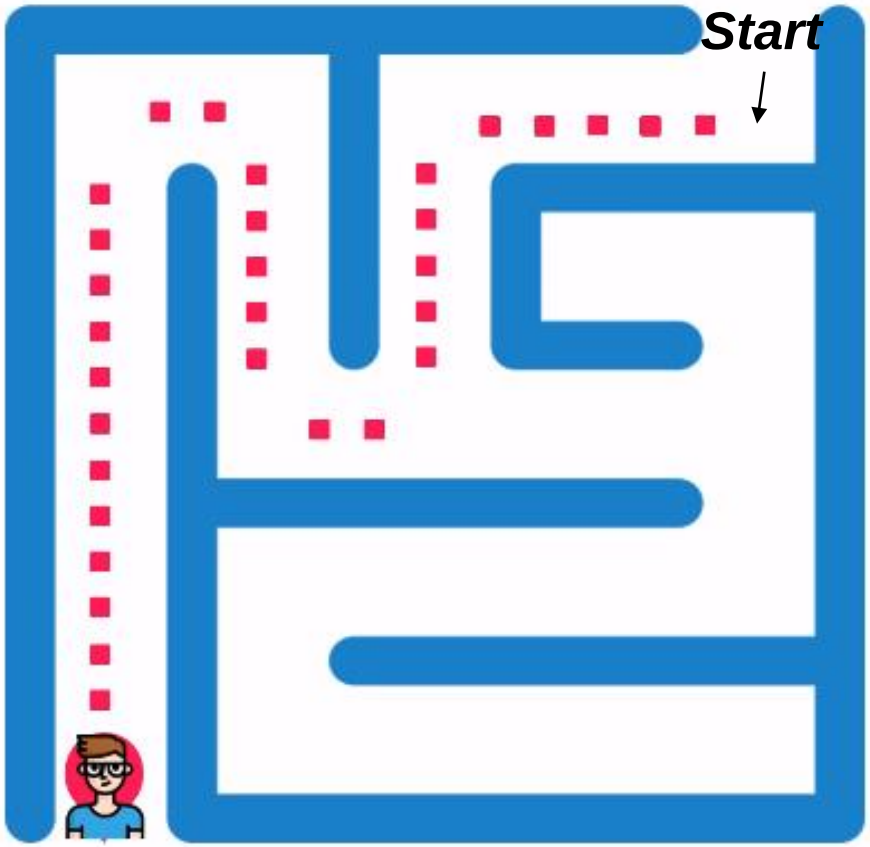
Étape 2: $x = \frac{-b}{2a} \quad si\ \Delta = 0$

Étape 3: $x_1 = \frac{-b - \sqrt{\Delta}}{2a}\ et\ x_2 = \frac{-b + \sqrt{\Delta}}{2a} \quad si\ \Delta > 0$

Étape 4: $Pas\ de\ solution\ \ sinon$

Introduction

Le meilleur trajet pour sortir du labyrinthe



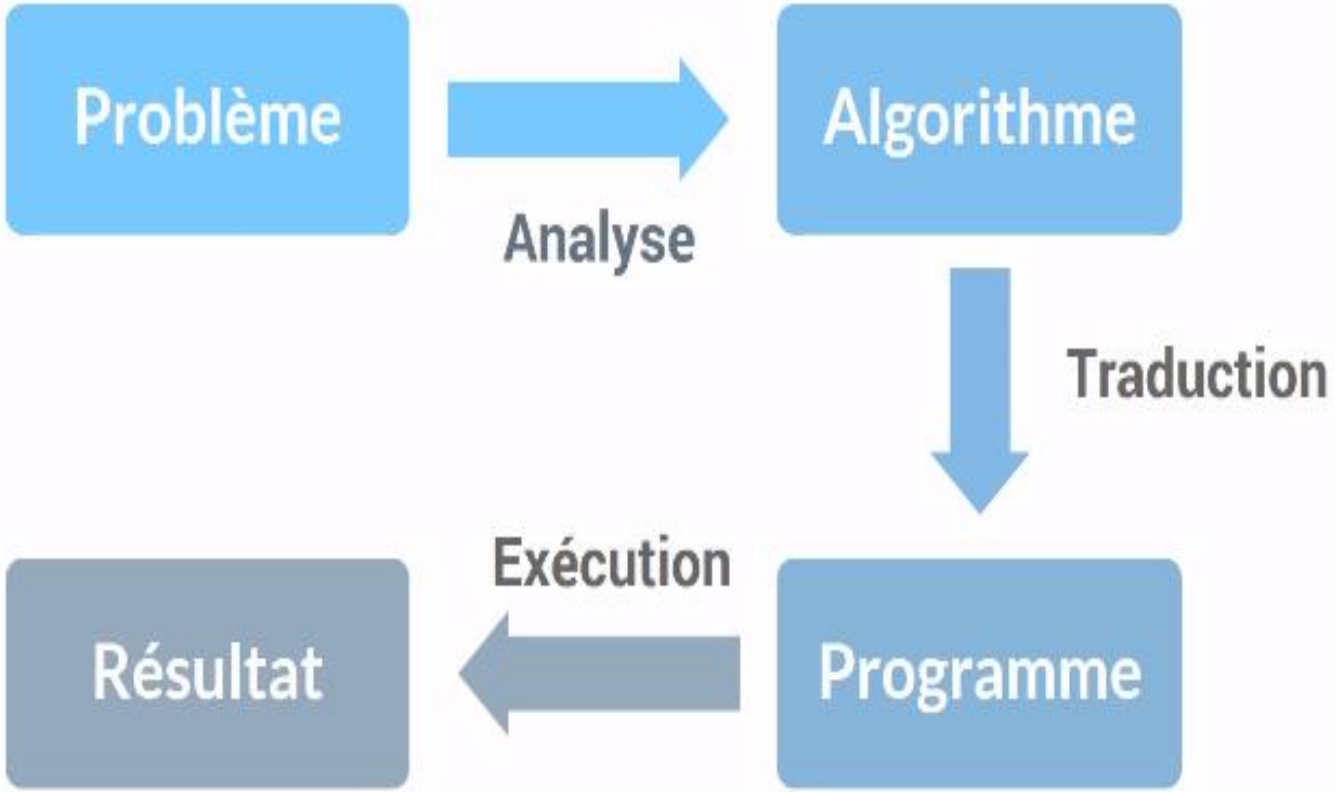
1 – Avancer 5 pas
2 – Tourner à gauche
3 – Avancer 5 pas
4 – Tourner à droite
5 – Avancer 2 pas
6 – Tourner à droite
7 – Avancer 5 pas
8 – Tourner à gauche
9 – Avancer 2 pas
10 – Tourner à gauche
11 – Avancer 12 pas

Définition

Un algorithme est une suite **d'actions** ou **d'instructions** élémentaires qui doivent être exécutées dans **un ordre bien** déterminé pour résoudre un problème (ou réaliser un travail).

Algorithmique

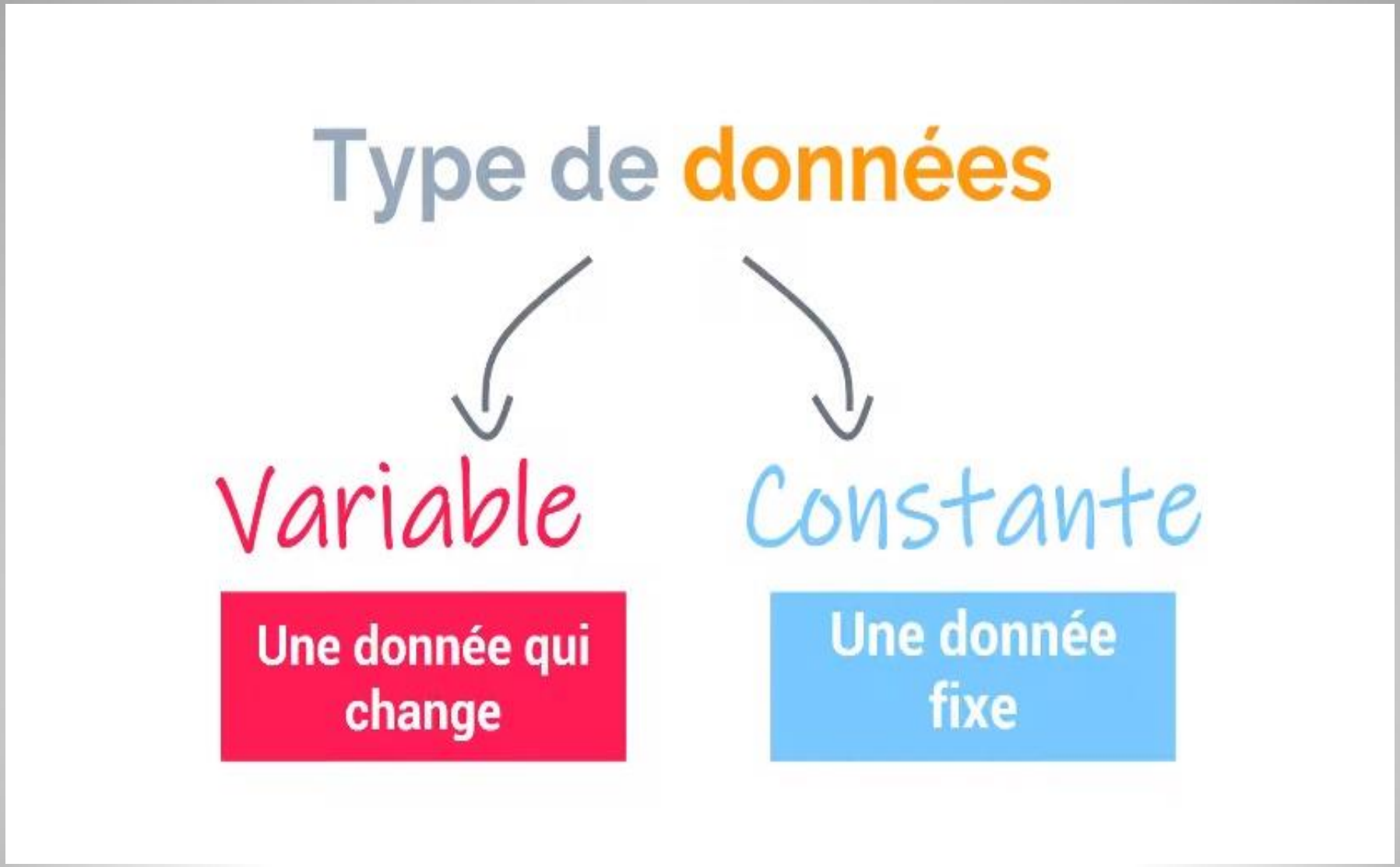
La résolution informatique d'un tel problème comporte les phases suivantes :



Notion de variable et de constante

	Gains	
Jours	Salaire	Pourboire
Lundi	150	92
Mardi	150	64
Mercredi	150	147
Jeudi	150	88
Vendredi	150	204
Samedi	150	190
Dimanche	150	0

Notion de variable et de constante



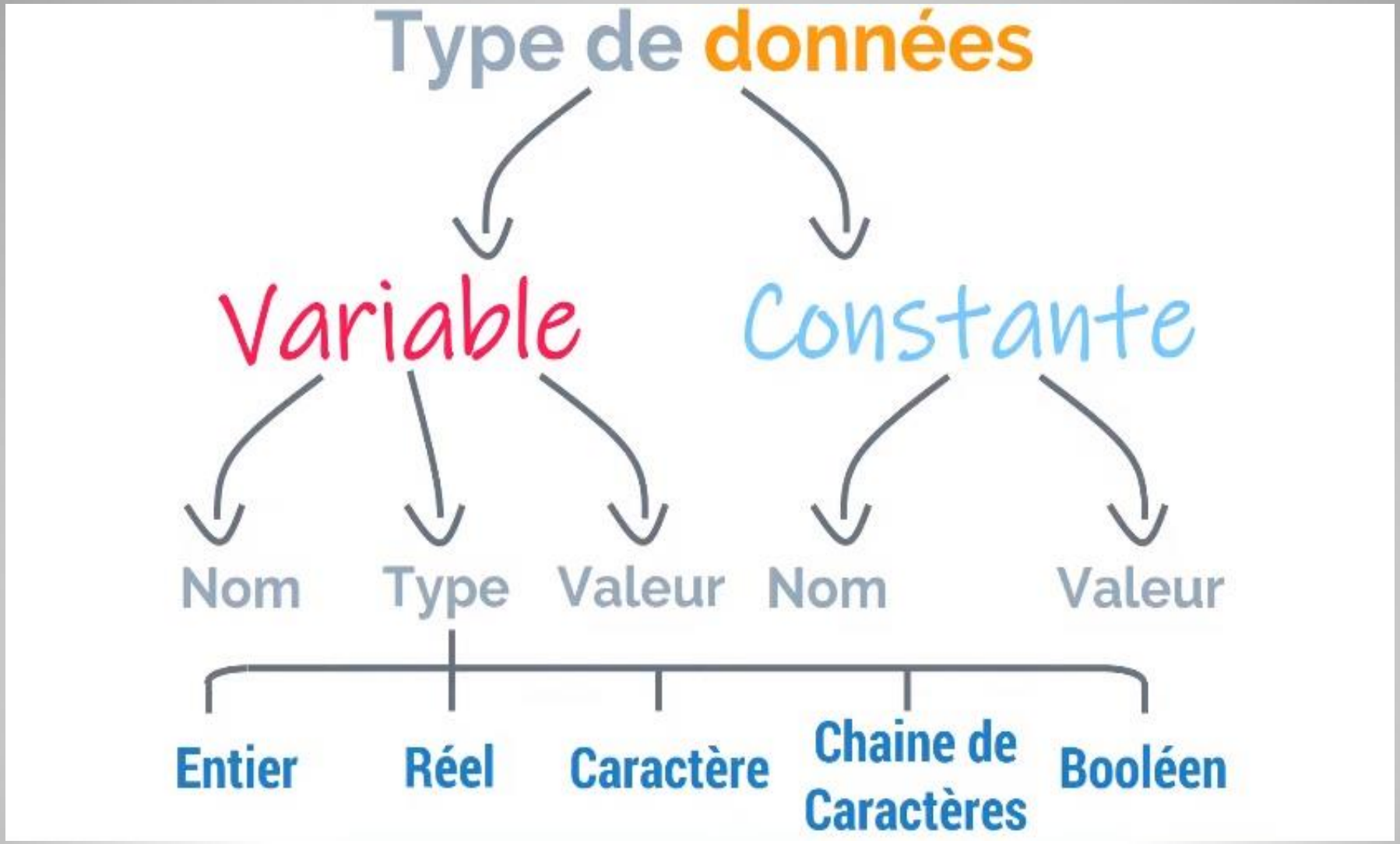
Notion de variable et de constante

Relevé de notes			
Nom étudiant		Ali	
N°	300	Sexe	H
Nbr étudiants		400	

Matière		Note	Valide
Algorithmique #1		18	Oui
Algorithmique #2		15.5	Oui
Programmation C		08	Non
Programmation C++		17	Oui
	Moyenne	15.37	
	Mention	Bien	

Variables
Nom étudiant
N°
Sexe
Matière
Note
Valide
Moyenne
Mention

Notion de variable et de constante



Notion de variable et de constante

Définition:

Les données sont des informations nécessaires au déroulement d'un algorithme. On distingue deux catégories :

- **Constante** : une donnée fixe qui **ne varie pas** durant l'exécution d'un algorithme.
- **Variable** : une donnée dont le contenu peut être **modifié** par une action durant l'exécution d'un algorithme.

Notion de variable et de constante

Types de données

Type de données		Exemples
Numérique	Entier	-345 178 2019
	Réel	-158.54 3 $4.5x10^{38}$
Alphanumérique	Caractère	"A" "2" "$"
	Chaine de caractères	"ENCGM" "Adam Smith" "35887" "Bonjour"
Booléen		Vrai Faux

Notion de variable et de constante

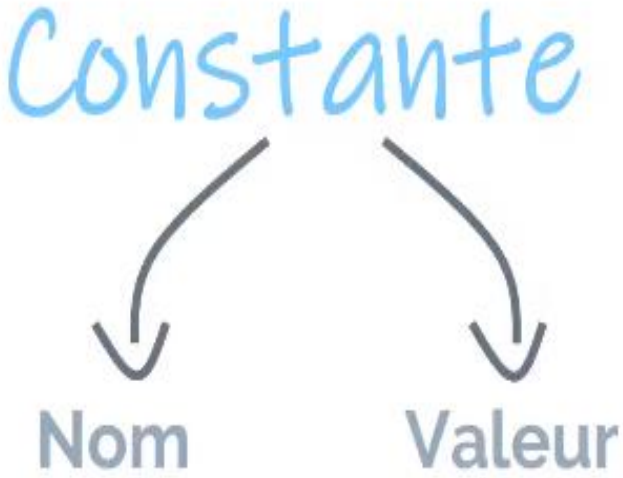
Exercice 1

Donnez le type des données suivantes :

Donnée	Type
" Bienvenue au Maroc "	
-300	
"8"	
2506.5	
" @ "	
Vrai	
" Faux "	

Déclaration des variables et des constantes

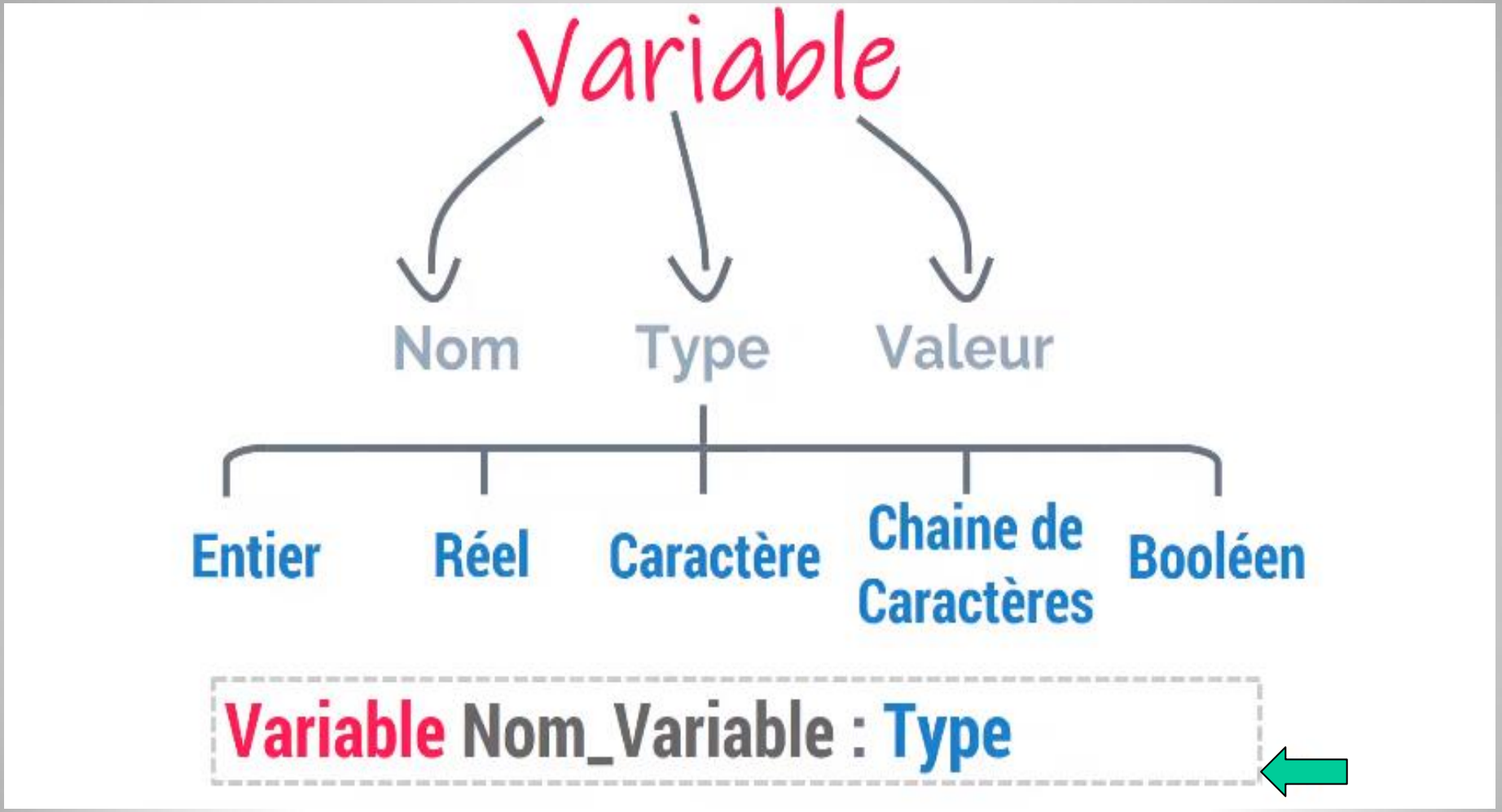
Syntaxe de déclaration d'une constante




```
Constante Nom_Constante = valeur
```

Déclaration des variables et des constantes

Syntaxe de déclaration d'une variable



Déclaration des variables et des constantes

Définition:

La déclaration permet d'informer l'ordinateur l'existence d'une donnée.

C'est-à-dire demander à l'ordinateur la permission de réserver un espace de la mémoire où l'on peut **stocker** et **récupérer** l'information.

Déclaration des variables et des constantes

Syntaxe de déclaration d'une constante

Une constante est caractérisée par son **nom** et sa **valeur** (**fixe**)

Syntaxe de déclaration:

```
Constante Nom_Constante = valeur
```

Exemples :

```
Constante Pi = 3.14
Constante nbr_Mois = 12
```

Déclaration des variables et des constantes

Syntaxe de déclaration d'une variable

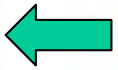
Une variable est caractérisée par son **nom**, sa **valeur** et son **type**.

Syntaxe de déclaration:

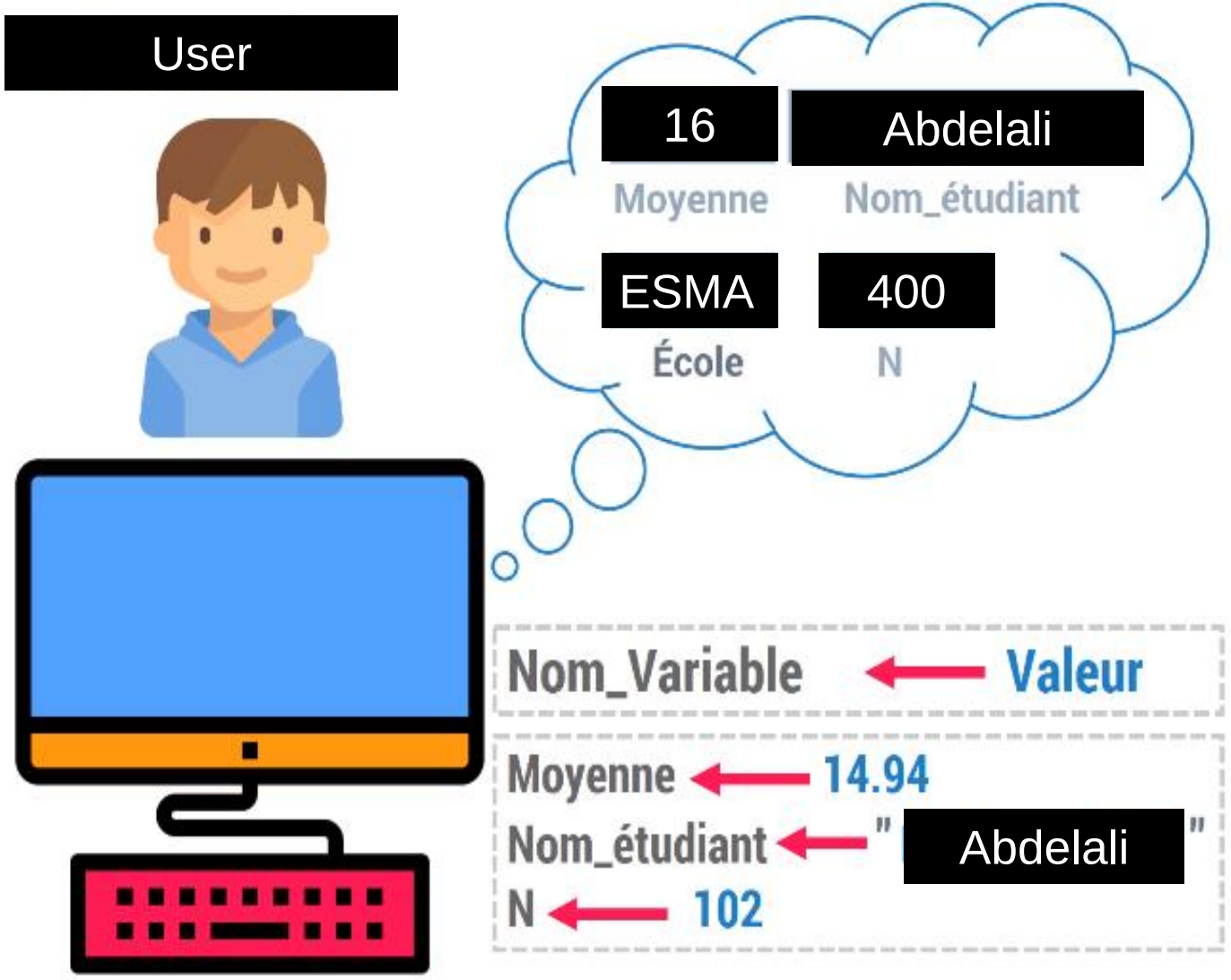
```
Variable Nom_Variable : Type
```

Exemples :

```
Variable num_Etudiant : entier
Variable note : réel
Variable prénom : chaine de caractères
Variable admis : booléen
Variable opérateur : caractère
```



Affectation



Affectation

Définition:

L'affectation est une opération qui consiste à attribuer à une variable :

- soit une valeur **particulière**,
- soit une valeur **contenue** dans une autre variable
- soit une valeur **calculée** à l'aide d'opérateurs arithmétiques.

Elle est représentée par une flèche orientée à gauche ←

Affectation

Syntaxe d'affectation:

```
Nom_Variable ← Valeur
```

Exemples :

```
A   ← 2
B   ← A
C   ← A + B
Nom ← "Ali"
```

la variable A reçoit la valeur 2
la variable B reçoit le contenu de A
la variable C reçoit le résultat de A plus B
la variable Nom reçoit la valeur Ali

Affectation

Exercice 2

Soient trois variables A, B et C tels que :

A est de type entier

B est de type chaine de caractères

C est type logique

1- Comment on déclare les variables A, B et C dans ce cas ?

Variable A : entier

Variable B : chaines de caractères

Variable C : booléen

Affectation

Exercice 3

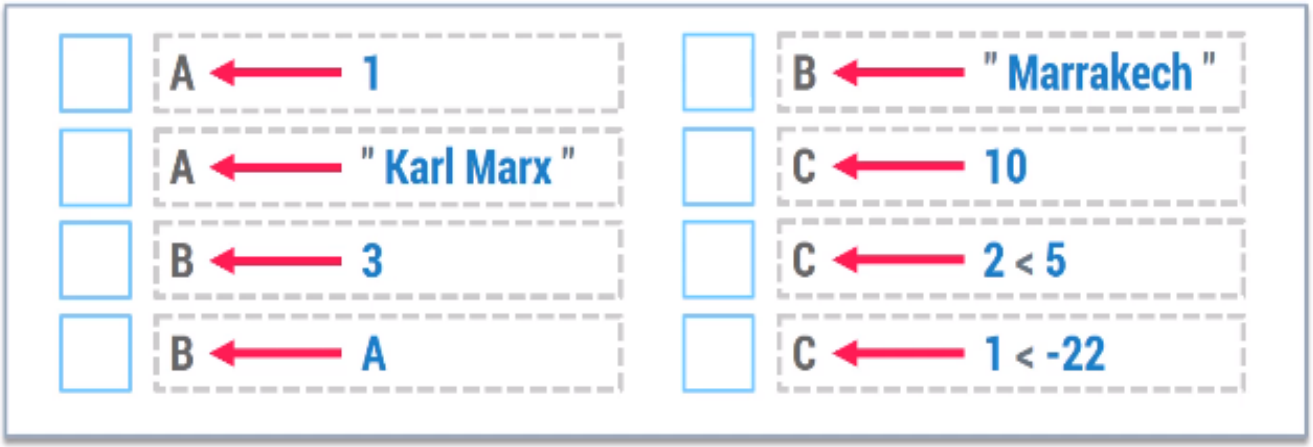
Soient trois variables A, B et C tels que :

A est de type entier

B est de type chaine de caractères

C est type logique

2- Cochez ce qui est juste :



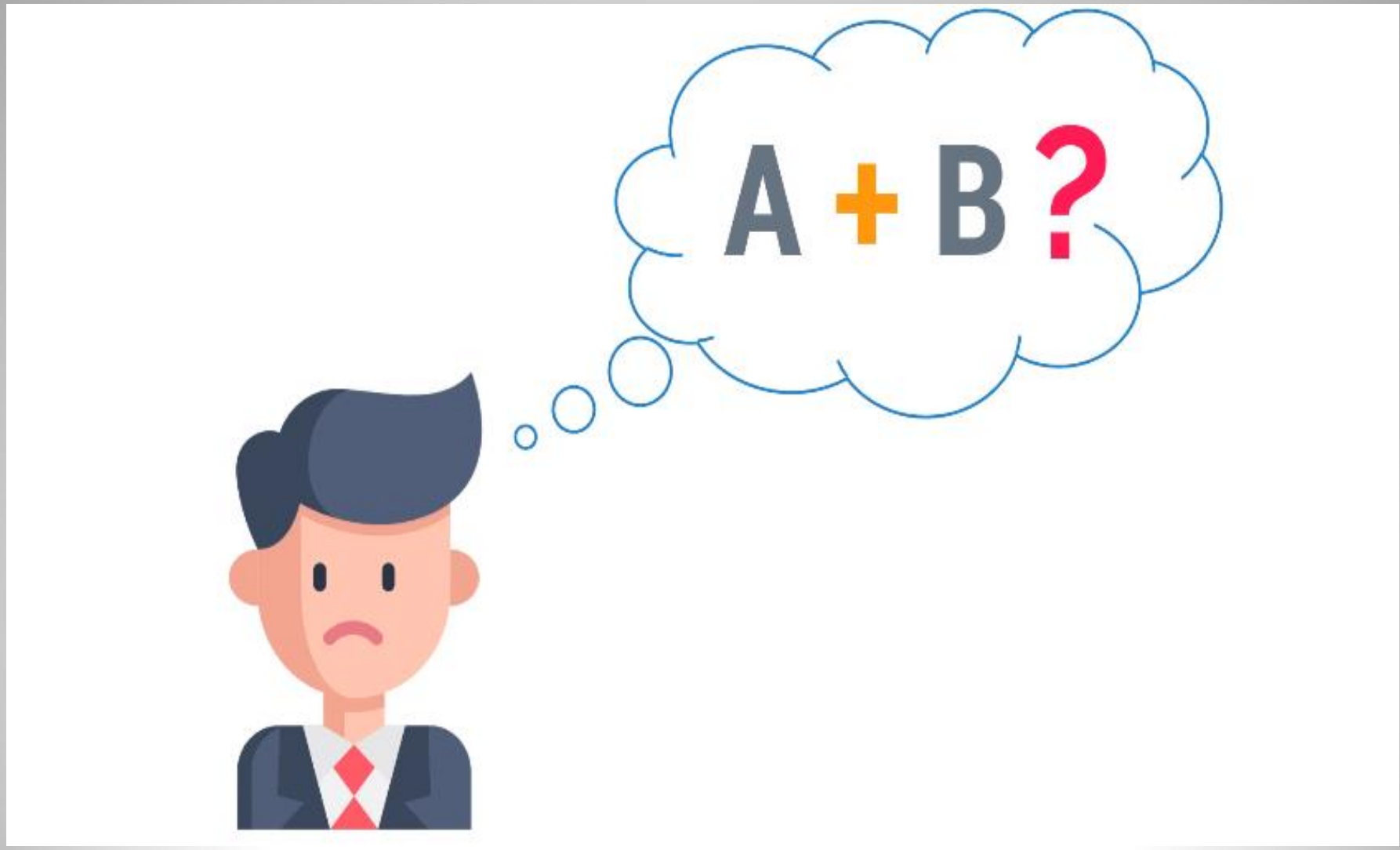
Affectation

Exercice

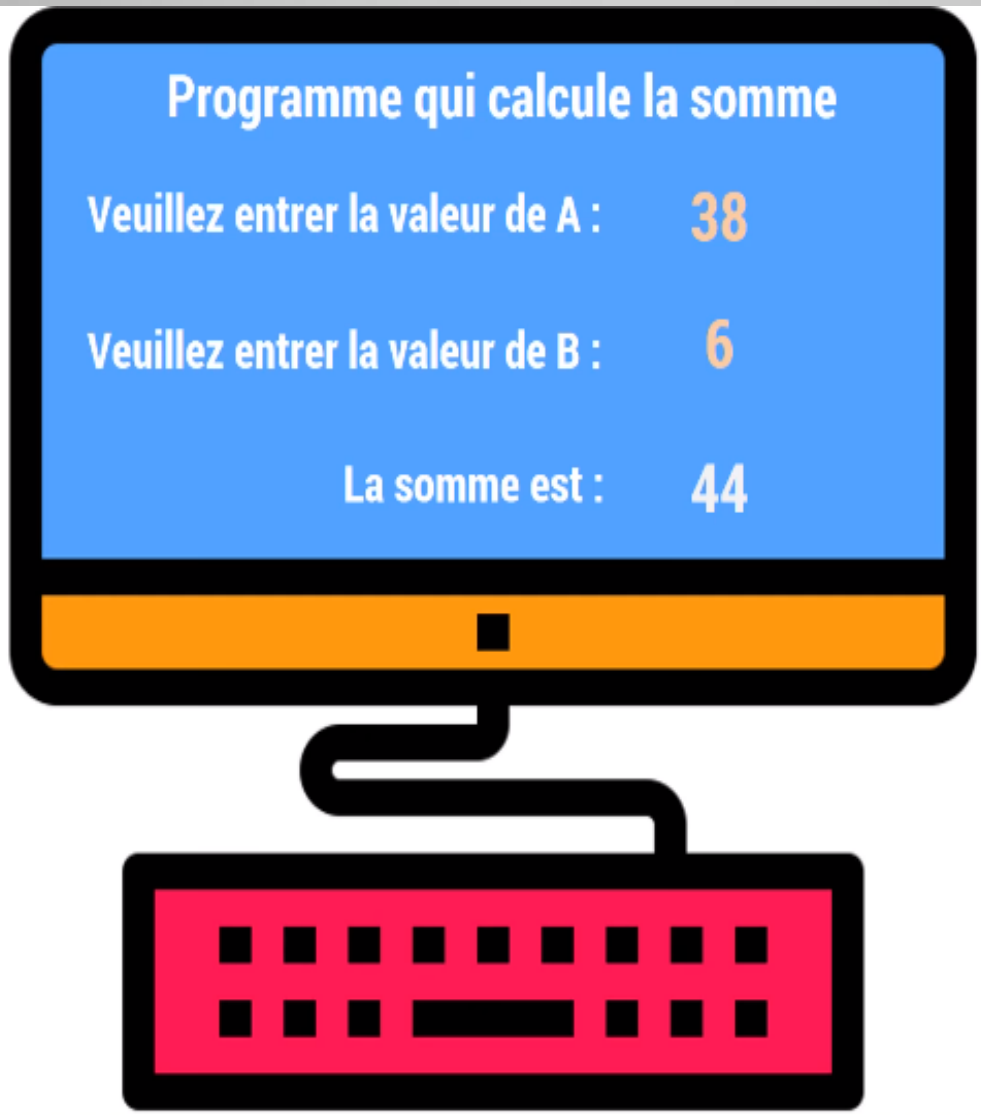
Complétez le tableau suivant :

	Variables			
Instructions	**A**	**B**	**C**	**D**
B ← 2				
C ← B + 10				
A ← 4				
D ← A				
B ← B * D				
C ← B + 5				
A ← 10 + 4 + C				
C ← A + B + D				

Ecrire et Lire



Ecrire et Lire



Définition

L'instruction Ecrire permet d'**afficher** la valeur d'une expression sur un périphérique de sortie (écran). Une expression peut être

- Un **nombre**,
- Une **variable** numérique,
- Un **résultat** d'une opération entre plusieurs variables,
- Une **chaîne de caractères**, dans ce cas, il est nécessaire de mettre la chaîne de caractères entre deux apostrophes.

Ecrire (Opération de sortie)

Syntaxe :

```
Ecrire ( variable )
Ecrire ( " Message " )
Ecrire ( " Message " , variable )
```

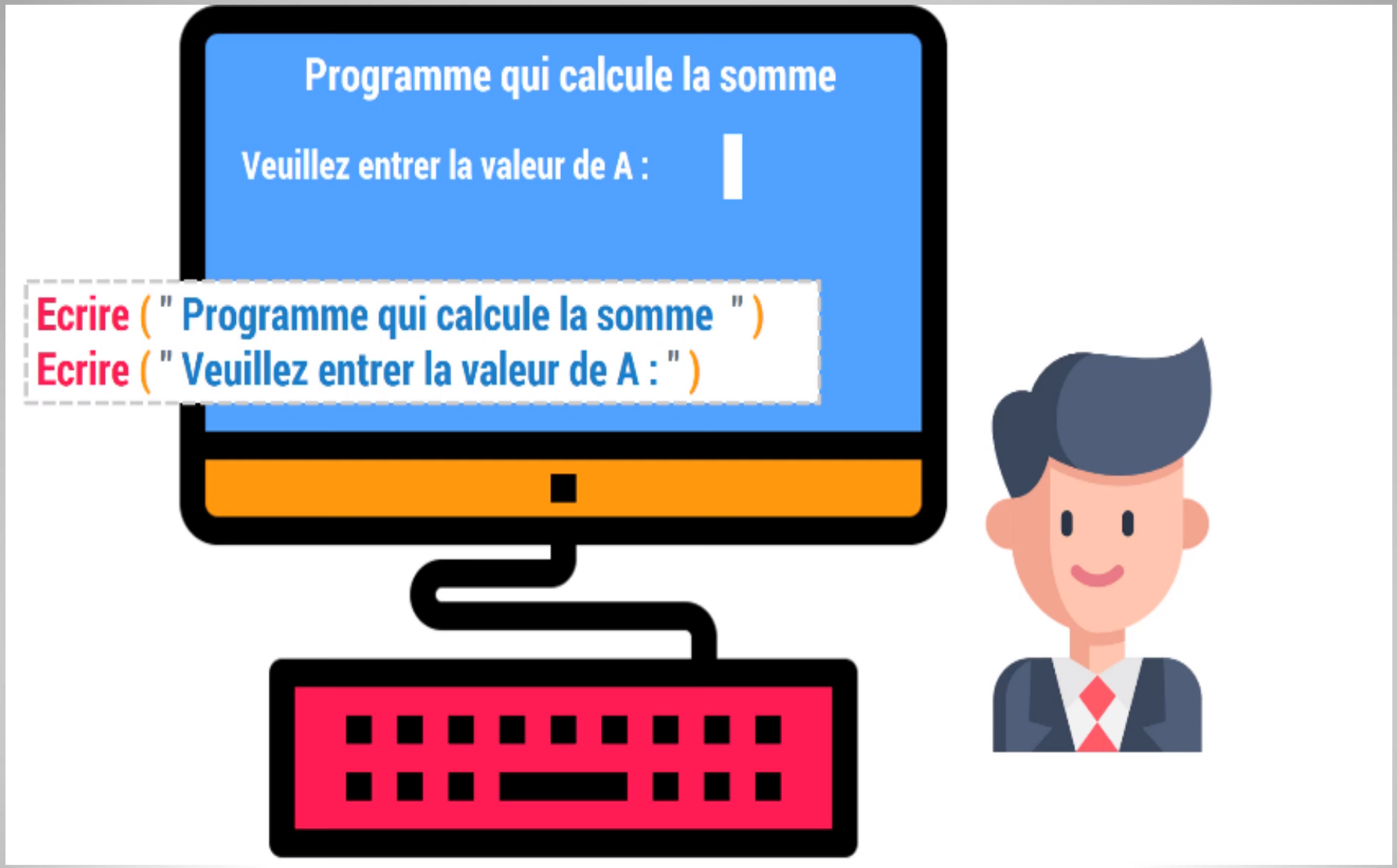
Exemples :

```
Ecrire ( A )

Ecrire ( " Maroc " )

Ecrire ( " A = " , A )
```

Signifie **affiché** sur l'écran le **contenu** de la variable A.

Signifie affiché sur l'écran le **message** suivant : **Maroc**

Ecrire et Lire



Lire (Opération d'entrée)

L'instruction Lire permet de **demander** à l'utilisateur de fournir des informations. Chaque information donnée par l'utilisateur est stockée dans une variable (attention au type !).

Syntaxe :

```
Lire ( variable1 )
Lire ( variable1 , variable2 , ... )
```

Lire (Opération d'entrée)

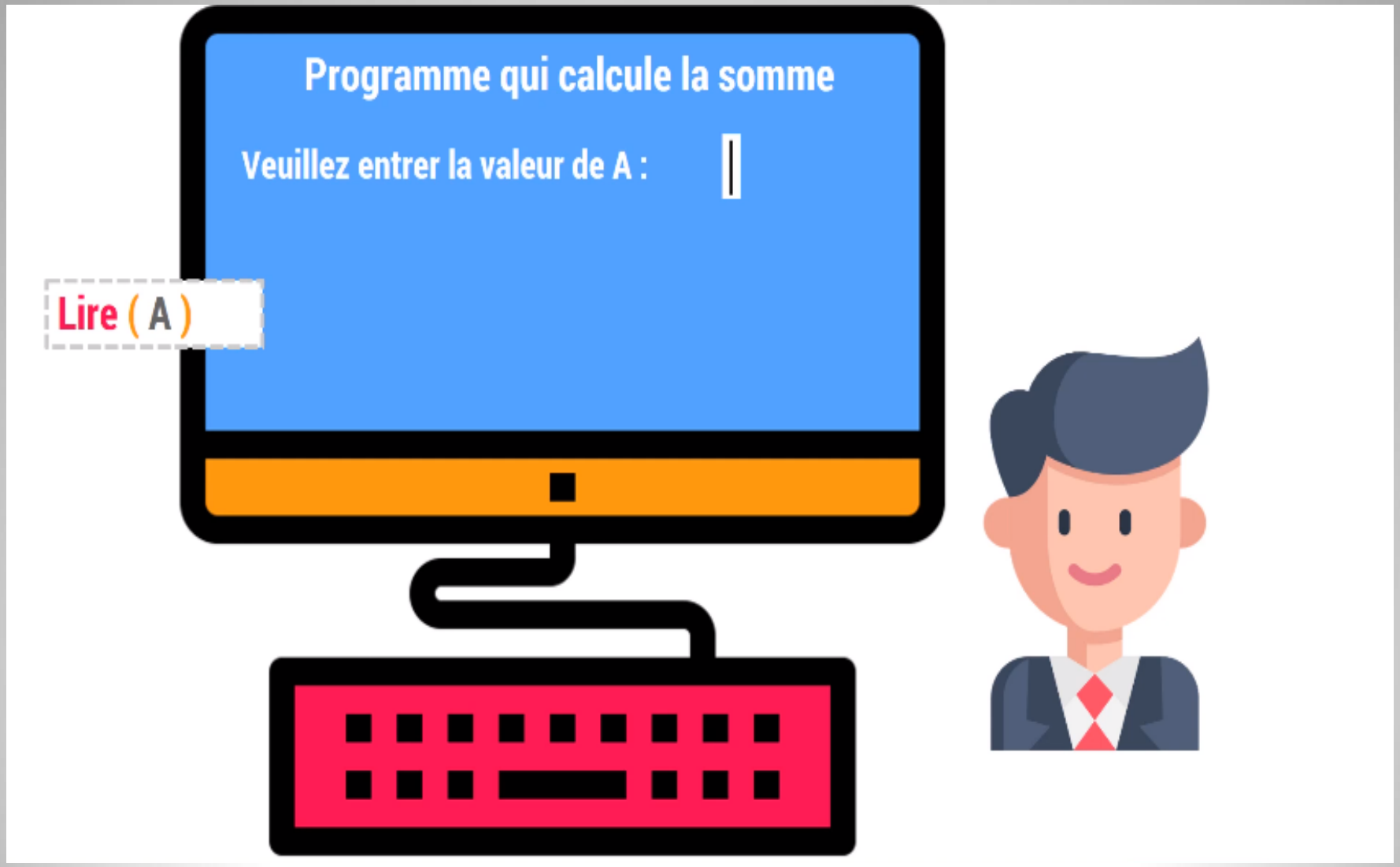
Exemples :

Lire (x)	Pour demander la valeur de x
Lire (nom , prénom , âge)	Pour demander le nom, prénom et âge.

Remarque :

Lors de l'exécution de l'instruction Lire la machine **attend** que l'utilisateur lui fournisse une valeur afin de pouvoir **continuer** à exécuter l'algorithme.

Ecrire et Lire



Exercice 1

Soient X, Y et Z trois variables :

X ⟵ 150

Y ⟵ 100

Z ⟵ " DH "

Qu'affichent les instructions suivantes ?

Instruction	Résultat
Ecrire (" Hello World ! ")	
Ecrire (X)	
Ecrire (" Z ")	
Ecrire (X * 2)	
Ecrire (" Prix : " , X + Y , Z)	

Exercice 2

Nous voulons écrire un algorithme qui calcule l'aire d'un cercle.

1 - Donner les instruction de déclaration.

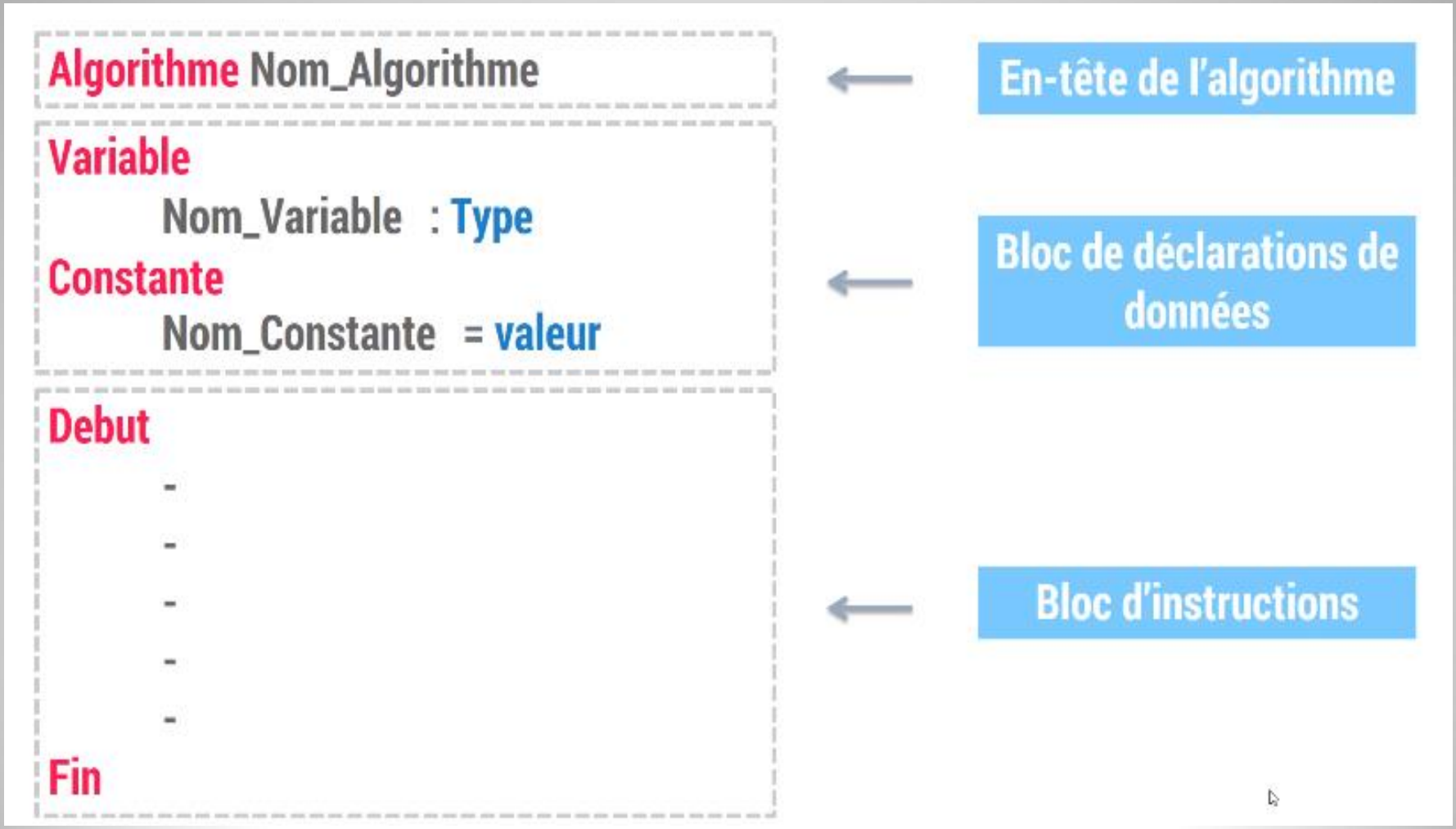
2 - Donner les instructions qui demandent à l'utilisateur de taper les valeur des données.

3 - Donner les instructions de traitement

4 - Donner les instructions qui permettent d'afficher le résultat

```
Variable Rayon , Surface  : réel
Constante Pi = 3.14
Ecrire ( " Veuillez entrer la valeur du rayon de cercle : " )
Lire ( Rayon )
Surface ⟵ Rayon * Rayon * Pi
Ecrire ( " L'aire de cercle est : " , Surface )
```

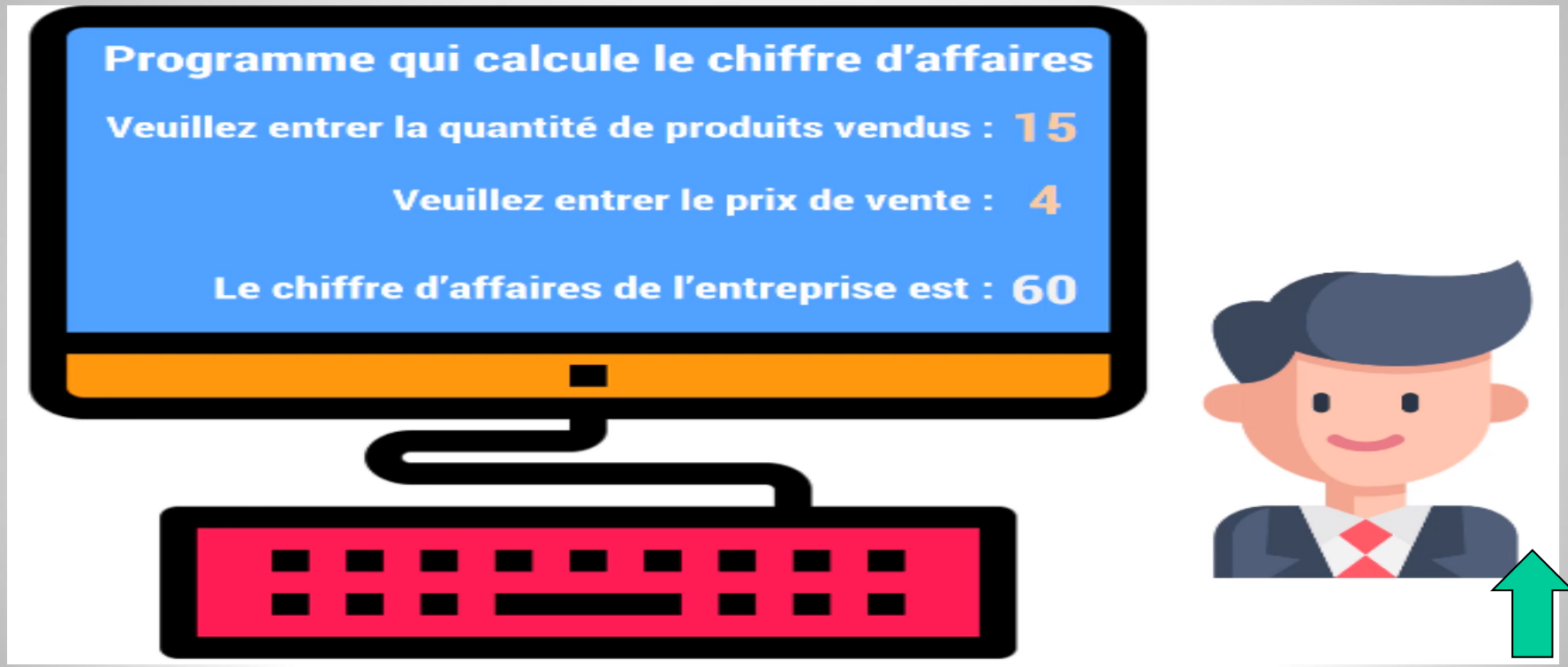
La structure d'un algorithme



La structure d'un algorithme

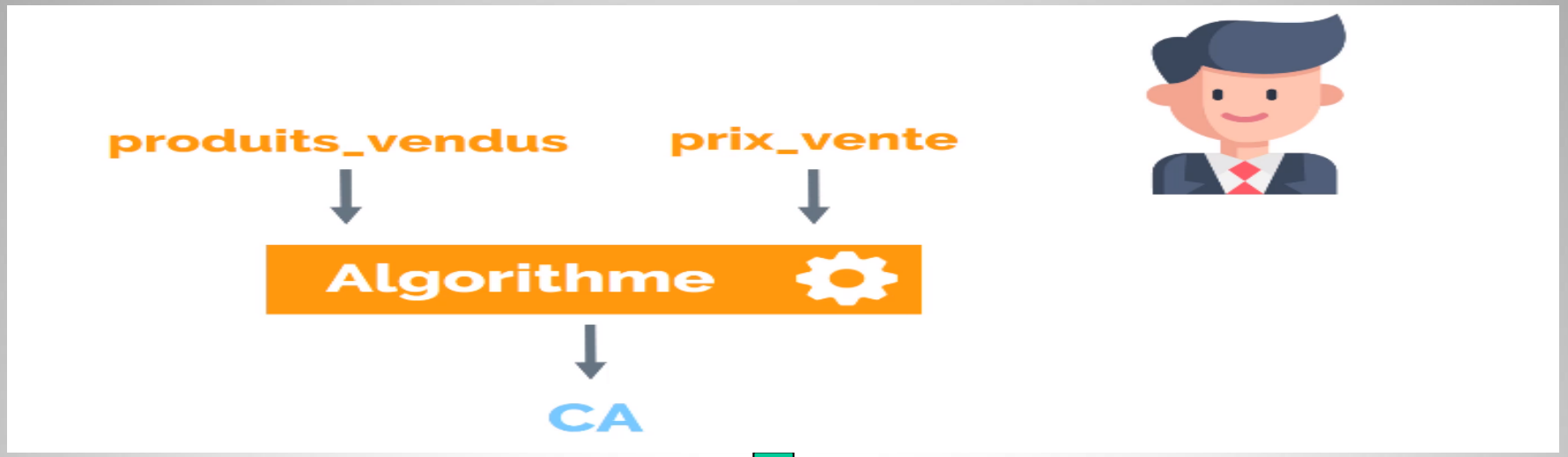
Exercice 1

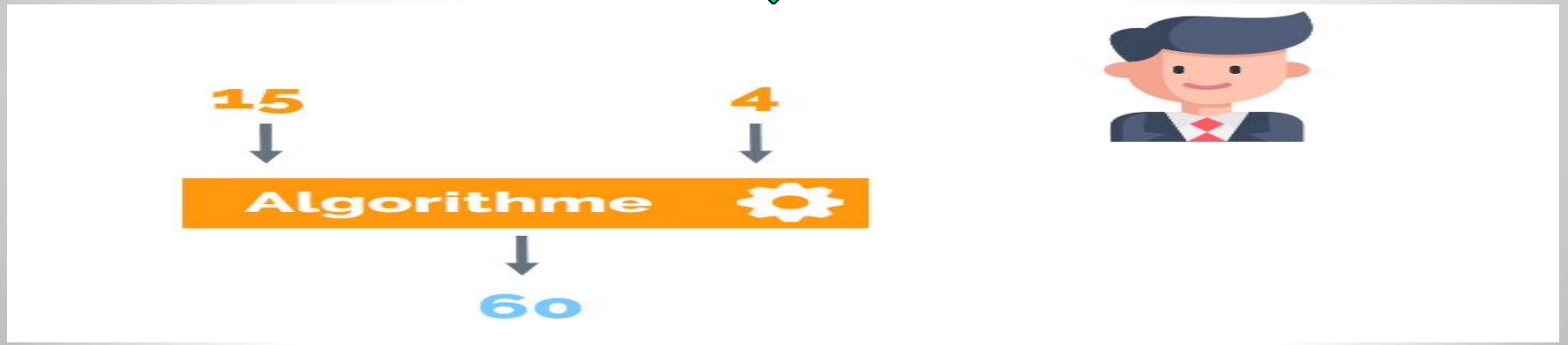
Ecrire un algorithme qui demande à l'utilisateur de taper la quantité de produits vendus, le prix de vente et qui affiche le chiffre d'affaires de l'entreprise.



La structure d'un algorithme

Exercice 1: Explication





La structure d'un algorithme

Solution de l'exercice

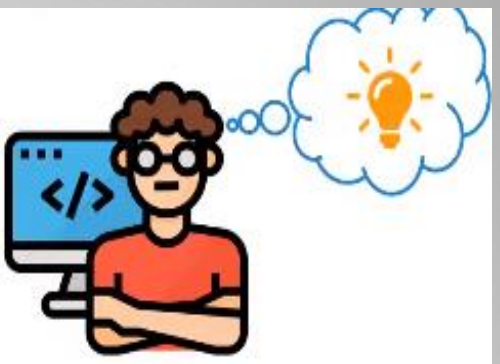
```
Algorithme chiffre_Affaires
Variables
        produits_vendus : Entier
        prix_vente , CA : Réel
Début
        Ecrire ( " Veuillez entrer la quantité de produits vendus : " )
        Lire ( produits_vendus )
        Ecrire ( " Veuillez entrer le prix de vente : " )
        Lire ( prix_vente )
        CA ← produits_vendus * prix_vente
        Ecrire ( " Le chiffre d'affaires de l'entreprise est : " , CA )
Fin
```

Les commentaires

```
Algorithme S
Variables
    SB , CS , SN : Réel
Début
        Ecrire ( " Veuillez entrer SB : " )
        Lire ( SB )
        Ecrire ( " Veuillez entrer CS : " )
        Lire ( CS )
        SN ⟵ SB – CS
        Ecrire ( " SN = " , SN )
Fin
```

Les commentaires

Commentaires



```
Algorithme S % Algorithme qui calcule le salaire net %
Variables
    SB , CS , SN : Réel % SB : salaire brut, CS : cotisations sociales,  SN : salaire net %
Début
        Ecrire ( " Veuillez entrer SB : " )
        Lire ( SB )
        Ecrire ( " Veuillez entrer CS : " )
        Lire ( CS )
        SN ⟵ SB – CS % Calcul de salaire net %
        Ecrire ( " SN = " , SN ) % Affichage de résultat %
Fin
```

Les commentaires

Définition

- Pour **accentuer la lisibilité** de l'algorithme et faciliter sa compréhension par les humains, on peut introduire des commentaires, qui ne sont pas des instructions destinées à la machine mais des indications données au lecteur de l'algorithme.
- Les commentaires sont **encadrés** par les symboles %.

Les commentaires

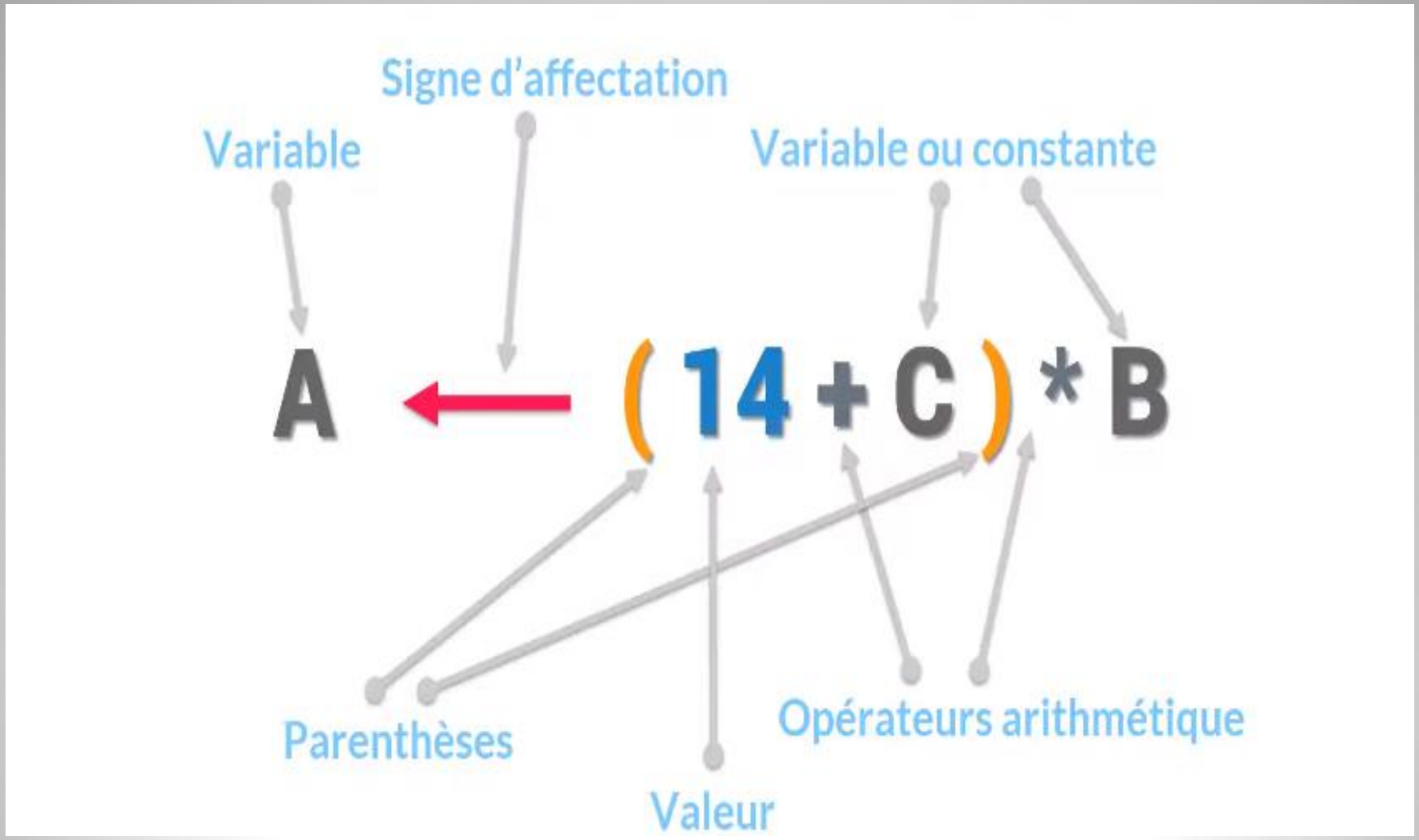
Syntaxe :

```
% Commentaire %
```

Exemple :

```
% Déclaration de la variable pays %

Variable pays : chaine de caractères

% Affectation de la valeur Maroc à la variable pays %

pays ← " Maroc "
```

Expressions arithmétiques

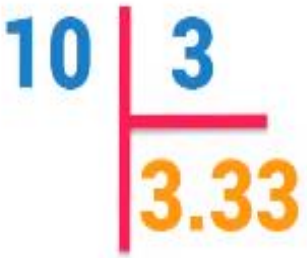


Expressions arithmétiques

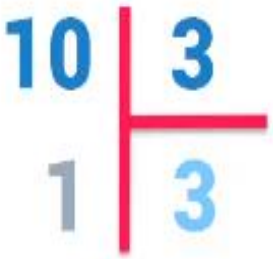
Définition

- Une expression (située à droite de la flèche) est un ensemble de valeurs, reliées par des opérateurs, et équivalent à une seule valeur.
- **Une Expression arithmétique** est formée par des combinaisons d'objets numériques (entier et réel) et des opérateurs arithmétiques.
- Une expression arithmétique donne un **résultat numérique** dont le type est entier ou réel.

Expressions arithmétiques



A ← 10 / 3



B ← 10 div 3

C ← 10 mod 3

10 ^ 3 = 10^3 = 1000

D ← 10 ^ 3

Expressions arithmétiques

Les opérateurs arithmétiques usuels sont :

Opérateur	Signification
+ , -	Addition , Soustraction
* , /	Multiplication , Division
div	Division entière
mod	Reste de la division entière
^	puissance

Expressions arithmétiques

Donner les valeurs des variables après l'exécution de chaque instruction:

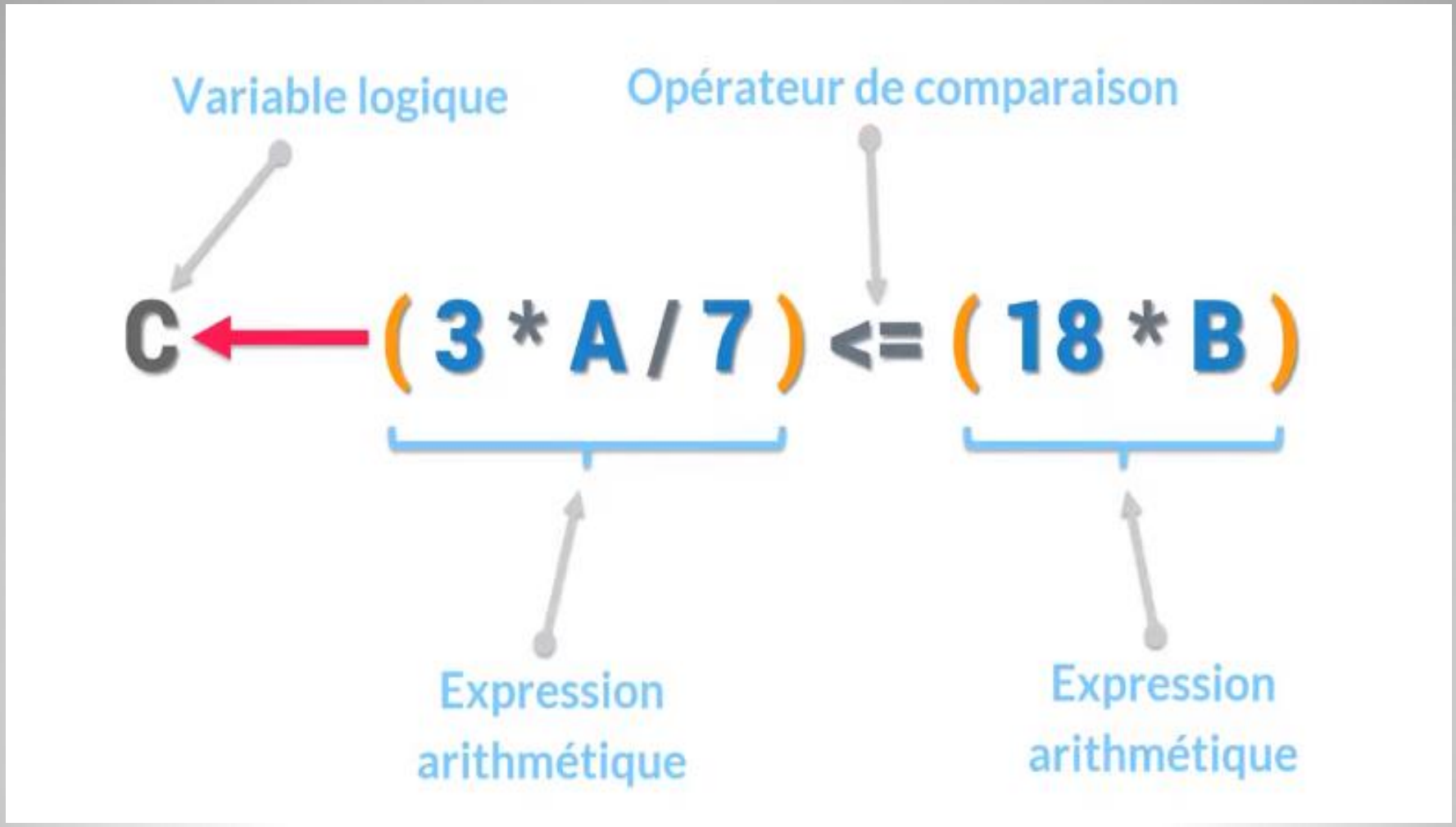
Instruction	Résultat
A ← 4 ^ 2	
B ← 8 * 5	
N ← A + B	
P ← N - 20	
R ← B / 3	
X ← B mod 3	
Y ← B div 3	

Expressions arithmétiques

Donner les valeurs des variables après l'exécution de chaque instruction:

Instruction	Résultat
C ← "Face"	
D ← "book"	
E ← C + D	

Expressions de comparaison

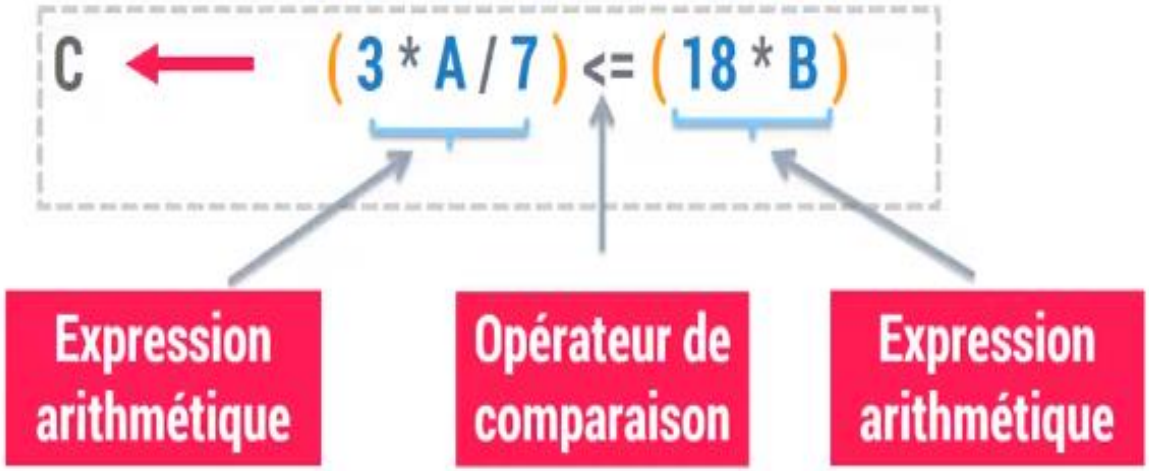


Expressions de comparaison

Elle est une comparaison entre deux expressions arithmétiques. Une expression de comparaison donne un **résultat booléen** (vrai ou faux).

Les opérateurs de comparaison usuels sont : **>** , **=** , **<** , **>=** , **<=** , **<>**

Exemple :

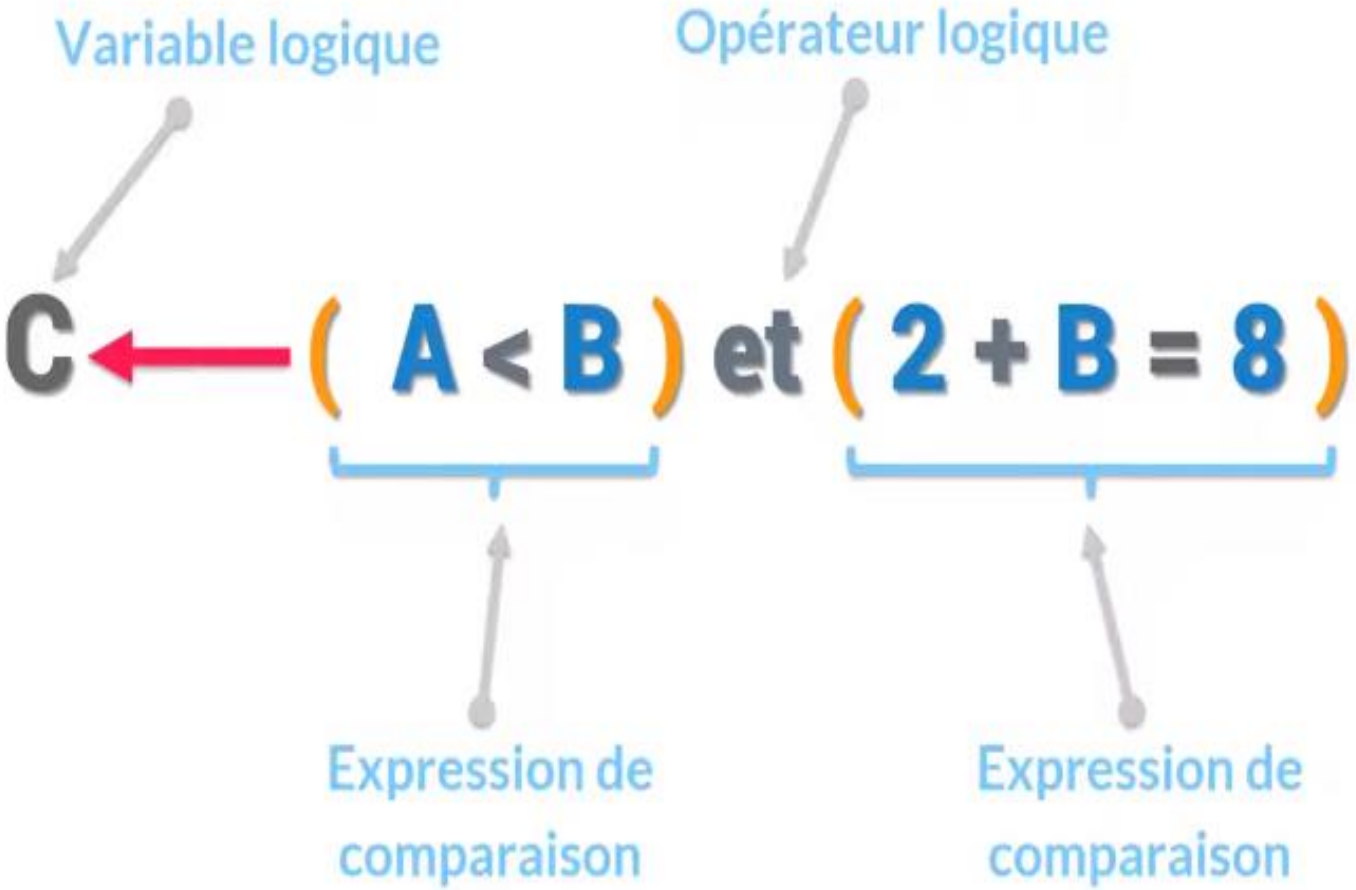


Expressions de comparaison

Donner les valeurs des variables après l'exécution de chaque instruction:

Instruction	Résultat
F ⟵ 3 < 5	
G ⟵ 2 <> 4	
H ⟵ 1 > 8	
X ⟵ 4 + 5	
K ⟵ (X mod 3) = (X – 8)	
J ⟵ (X * 10) >= 90	
M ⟵ (X * X + X) > (X ^ 3)	

Expressions logiques



Expressions logiques

Soit **A** et **B** deux variables booléennes. Le tableau suivant illustre les différentes valeurs de vérité que l'on obtient en combinant les valeurs de **A** et **B** à l'aide des opérateurs logiques.

A	B	A et B	A ou B	non A
Faux	Faux	Faux	Faux	Vrai
Faux	Vrai	Faux	Vrai	Vrai
Vrai	Faux	Faux	Vrai	Faux
Vari	Vrai	Vrai	Vrai	Faux

Expressions logiques

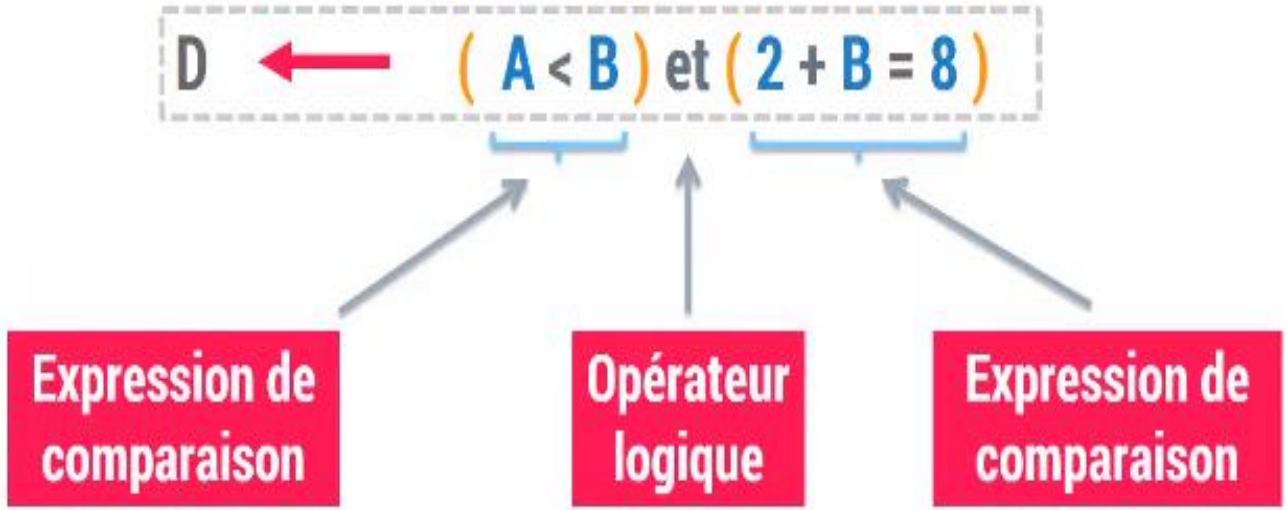
Exemples :

Instruction	Résultat
A ← 5	5
B ← 6	6
C ← (A < B) Et (3 > B)	
D ← (A = 5) Ou (B > 10)	
E ← Non (C)	
F ← C Ou (E Et D)	
G ← (Non (E) Et F) Ou (C Et D)	

Expressions logiques

Une expression logique est la composée d'expressions de comparaisons par les opérateurs logiques. Une expression logique donne un **résultat booléen** (vrai ou faux). Les opérateurs logiques usuels sont : **et** , **ou** , **non**

Exemple :



Expressions logiques

Soit A une variable logique, quelle sera la valeur de A dans chacun des cas suivants :

Instruction	Résultat
A ← Non (3 = 5)	
A ← (2 <> 2) Ou (3 > 4)	
A ← ((30 / 3) = 10) Et (3 > 30)	
A ← ((2 > 4) Et (8 <= 8)) Ou (3 <> 4)	
A ← (40 >= 30) Et ((10 ^ 2) <> 50)	
A ← Non ((82 mod 8) / 2 = 1)	

Structure conditionnelle Si ... Alors ... Fin Si

```
Algorithme division
Variables
        A , B : Réel
Début
        Ecrire ( " Veuillez entrer le dividende" )
        Lire ( A )
        Ecrire ( " Veuillez entrer le diviseur" )
        Lire ( B )
        Ecrire ( " Le résultat est : " , A / B )
Fin
```

Structure conditionnelle Si ... Alors ... Fin Si

La structure séquentielle

Un algorithme qui suit une structure séquentielle est un algorithme dont **toutes** les instructions sont exécutées **l'une après l'autre** de façon à ce que l'ordre des instructions est respecté.

Structure conditionnelle Si ... Alors ... Fin Si

La structure conditionnelle est une structure dont les instructions sont exécutées selon **les réponses des conditions**.

1 - Structure conditionnelle Simple (un choix)

Syntaxe :

```
Si Condition Alors
        Instructions
Fin Si
```

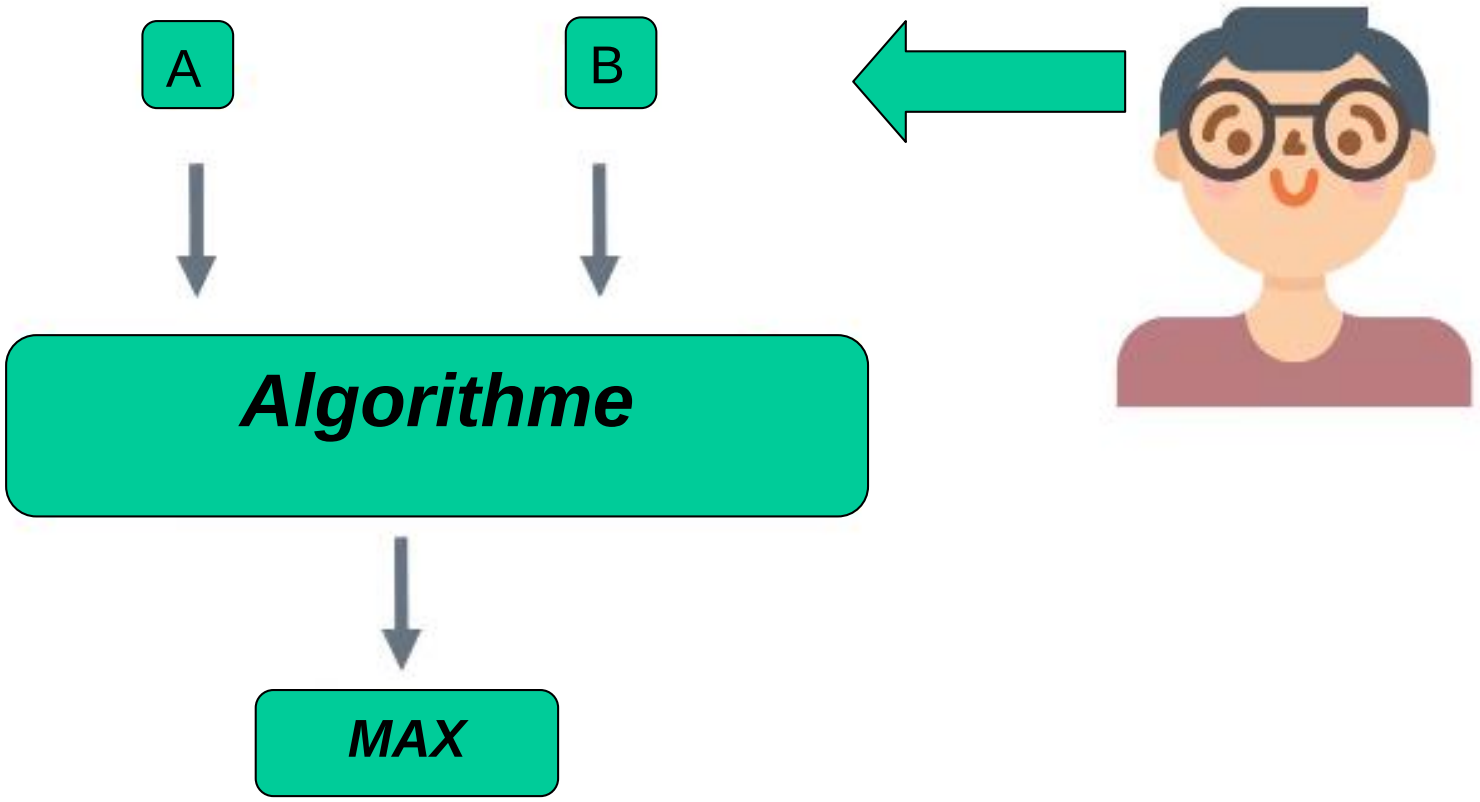
Si la condition vaut **Vrai** alors le bloc d'instructions sera exécuté, si non il sera ignoré.

Structure conditionnelle Si ... Alors ... Fin Si

```
Algorithme division
Variables
        A , B : Réel
Début
        Ecrire ( " Veuillez entrer le dividende" )
        Lire ( A )
        Ecrire ( " Veuillez entrer le diviseur" )
        Lire ( B )
        Si  B <> 0 Alors
                Ecrire ( " Le résultat est : " , A / B )
        Fin si
Fin
```

Structure conditionnelle Si ... Alors ... Fin Si

Ecrire un algorithme qui permet de calculer le maximum de deux nombres réels saisies par l'utilisateur.



Structure conditionnelle Si ... Alors ... Fin Si

Ecrire un algorithme qui permet de calculer le maximum de deux nombres réels saisies par l'utilisateur.

```
Algorithme Maximum
Variables
        A , B , Max : Réel
Début
        Ecrire ( " Entrez les valeurs de A et de B: " )
        Lire ( A , B )
        Max ⟵ A
        Si  Max < B Alors
                Max ⟵ B
        Fin si
        Ecrire ( " Le maximum est égale à : " , Max )
Fin
```

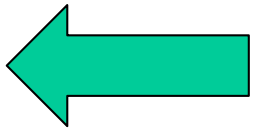
Structure conditionnelle Si ... Alors ... Fin Si

2 - Structure alternative (deux choix)

Syntaxe :

```
Si Condition Alors
        Instructions1
Sinon
        Instructions2
Fin Si
```

Si la condition vaut **Vrai** alors le bloc **d'instructions1** sera exécuté, et le bloc d'instructions2 sera ignoré, **sinon** le bloc d'**instructions2** sera exécuté et le bloc d'instructions1 sera ignoré.

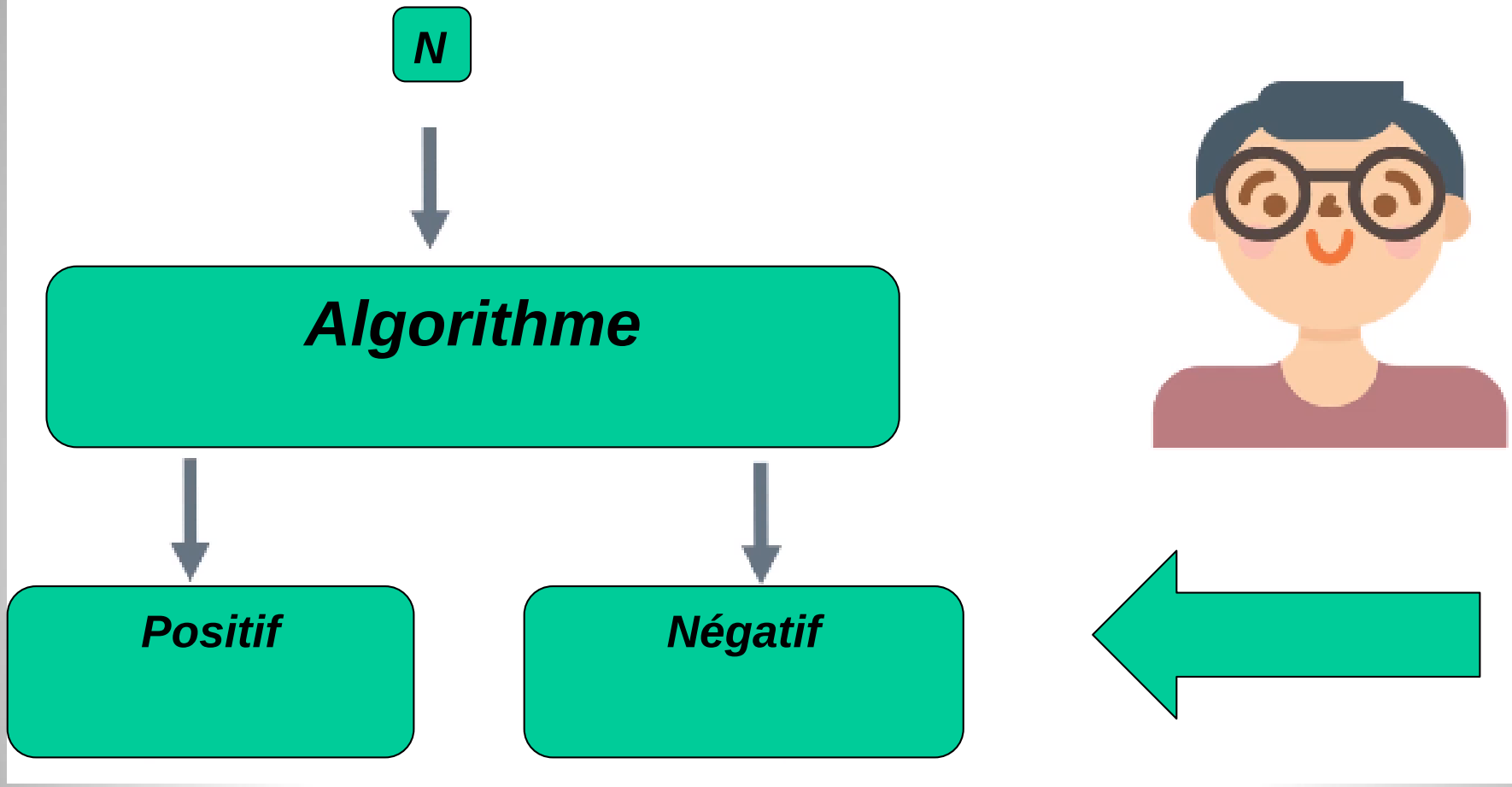


Structure conditionnelle Si ... Alors ... Fin Si

```
Algorithme division
Variables
        A , B : Réel
Début
        Ecrire ( " Veuillez entrer le dividende" )
        Lire ( A )
        Ecrire ( " Veuillez entrer le diviseur" )
        Lire ( B )
        Si  B <> 0 Alors
                Ecrire ( " Le résultat est : " , A / B )
        Sinon
                Ecrire ( " La division par 0 est impossible " )
        Fin si
Fin
```

Structure conditionnelle Si ... Alors ... Fin Si

Ecrire un algorithme qui permet de demander un nombre entier à l'utilisateur, et l'informe ensuite si ce nombre est positif ou négatif.



Structure conditionnelle Si ... Alors ... Fin Si

Ecrire un algorithme qui permet de demander un nombre entier à l'utilisateur, et l'informe ensuite si ce nombre est positif ou négatif.

```
Algorithme nature_Nombre
Variables
        n : Entier
Début
        Ecrire ( " Programme qui détermine la nature d'un nombre : " )
        Ecrire ( " Veuillez entrer un nombre : " )
        Lire ( n )
        Si  n > 0 Alors
                Ecrire ( " Ce nombre est positif ")
        Sinon
                Ecrire ( " Ce nombre est négatif " )
        Fin si
Fin
```

Structure alternative Si imbriquée

3 - Structure alternative imbriquée (multiple choix)

Syntaxe :

```
Si Condition1 Alors
        Si Condition2 Alors
                Instructions1
        Sinon
                Instructions2
        Fin Si
Sinon
        Instructions3
Fin Si
```

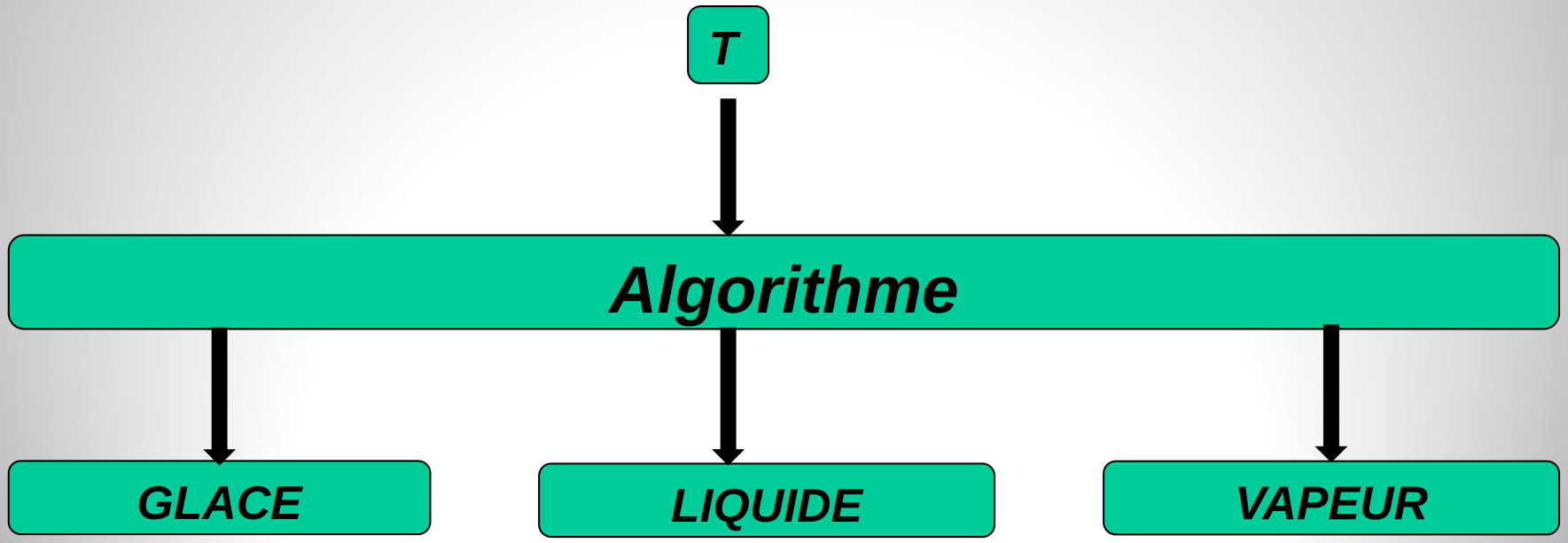
Structure alternative Si imbriquée

```
Algorithme nature_Nombre
Variables
        n : Entier
Début
        Ecrire ( " Programme qui détermine la nature d'un nombre : " )
        Ecrire ( " Veuillez entrer un nombre : " )
        Lire ( n )
        Si  n > 0 Alors
                Ecrire ( " Ce nombre est positif ")
        Sinon
                Si  n = 0 Alors
                        Ecrire ( " Ce nombre est nul ")
                Sinon
                        Ecrire ( " Ce nombre est négatif " )
                Fin si
        Fin si
Fin
```

Structure alternative Si imbriquée

Ecrire un algorithme qui permet de lire la valeur de la température de l'eau et d'afficher son état :

- **GLACE** Si la température est inférieure à 0
- **LIQUIDE** Si la température est strictement supérieure à 0 et strictement inférieure à 100
- **VAPEUR** Si la température est supérieure à 100.

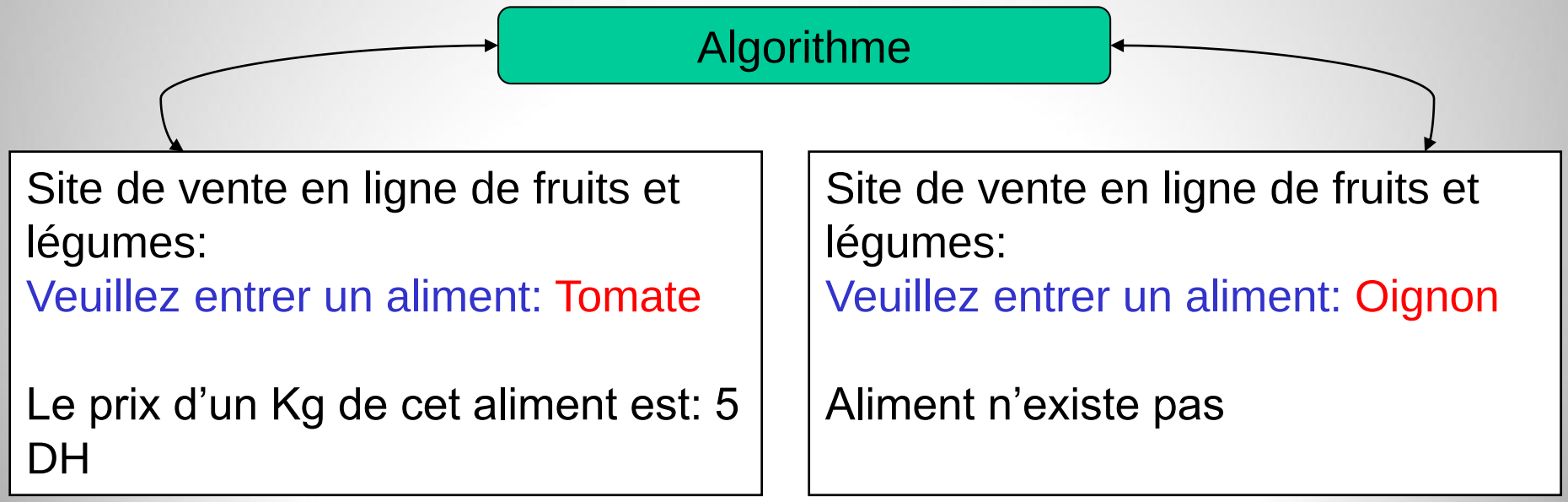


Structure alternative Si imbriquée

```
Algorithme état_Eau
Variables
        T : Réel
Début
        Ecrire ( " Veuillez entrer la température : " )
        Lire ( T )
        Si  T <= 0 Alors
                Ecrire ( " L'état de l'eau est GLACE ")
        Sinon
                Si  T >= 100 Alors
                        Ecrire ( " L'état de l'eau est VAPEUR " )
                Sinon
                        Ecrire ( " L'état de l'eau est LIQUIDE " )
                Fin si
        Fin si
Fin
```

Structure conditionnelle à choix multiple cas ou selon

Aliment	Prix par Kg (DH)
Banane	11
Avocat	18
Pomme	14
Tomate	5
Carotte	2



Structure conditionnelle à choix multiple cas ou selon

```
Algorithme prix_aliment
Variables
        A : chaine de caractères
Début
  Ecrire ( " Veuillez entrer un aliment " )
  Lire ( A )
    Si  A = "Banane" Alors
      Ecrire ( " Le prix d'un Kg de cet aliment est : 9DH ")
    Sinon
      Si A = "Pomme" Alors
        Ecrire ( " Le prix d'un Kg de cet aliment est : 8DH " )
      Sinon
        Si A = "Avocat" Alors
          Ecrire ( " Le prix d'un Kg de cet aliment est : 22DH " )
        Sinon
          Si A = "Oignon" Alors
            Ecrire ( " Le prix d'un Kg de cet aliment est : 4DH " )
          Sinon
            Si A = "Tomate" Alors
              Ecrire ( " Le prix d'un Kg de cet aliment est : 3DH" )
            Sinon
              Si A = "Carotte" Alors
                Ecrire ( " Le prix d'un Kg de cet aliment est : 6DH " )
              Sinon
                Ecrire ( " Aliment n'existe pas " )
              Fin Si
            Fin Si
          Fin Si
        Fin Si
      Fin Si
    Fin Si
  Fin Si
Fin
```

Structure conditionnelle à choix multiple cas ou selon

4 - Structure à choix multiple

Lorsque l'imbrication des alternatives devient importante, l'utilisation de la structure à choix multiple devient nécessaire.

Syntaxe :

```
Cas Variable Vaut
        Valeur 1 : Instruction 1
        Valeur 2 : Instruction 2
            ...  :  ...
        Valeur n : Instruction n
        Sinon : Autre instruction
Fin Cas
```

Structure conditionnelle à choix multiple cas ou selon

```
Algorithme prix_aliment
Variables
        A : chaine de caractères
Début
        Ecrire ( " Veuillez entrer un aliment " )
        Lire ( A )
        Cas  A vaut
                "Banane" : Ecrire ( " Le prix d'un Kg de cet aliment est : 9DH ")
                "Pomme" : Ecrire ( " Le prix d'un Kg de cet aliment est : 8DH " )
                "Avocat"  : Ecrire ( " Le prix d'un Kg de cet aliment est : 22DH " )
                "Oignon"  : Ecrire ( " Le prix d'un Kg de cet aliment est : 4DH " )
                "Tomate" : Ecrire ( " Le prix d'un Kg de cet aliment est : 3DH" )
                "Carotte" : Ecrire ( " Le prix d'un Kg de cet aliment est : 6DH " )
                Sinon :     Ecrire ( " Aliment n'existe pas " )
        Fin cas
Fin
```

Structure conditionnelle à choix multiple cas ou selon

Ecrire un algorithme qui permet de demander a utilisateur de saisir un entier entre 1 et 7 au clavier, et afficher le nom du jour correspondant.

```
Algorithme jour_Semaine
Variables
        n : Entier
Début
        Ecrire ( " Veuillez entrer un nombre entre 1 et 7 : " )
        Lire ( n )
        Cas  n vaut
                1 : Ecrire ( " Lundi " )
                2 : Ecrire ( " Mardi " )
                3 : Ecrire ( " Mercredi " )
                4 : Ecrire ( " Jeudi " )
                5 : Ecrire ( " Vendredi " )
                6 : Ecrire ( " Samedi " )
                7 : Ecrire ( " Dimanche " )
                Sinon : Ecrire ( " Le nombre est incorrect " )
        Fin cas
Fin
```

Table de multiplication de 7

0 * 7 = 0
1 * 7 = 7
2 * 7 = 14
3 * 7 = 21
4 * 7 = 28
5 * 7 = 35
6 * 7 = 42
7 * 7 = 49
8 * 7 = 56
9 * 7 = 63
10 * 7 = 70

Structures répétitives

```
Algorithme table_7
Variables
        M : Entier
Début
        M ⟵ 0 * 7
        Ecrire ( " 0 * 7 = " , M )
        M ⟵ 1 * 7
        Ecrire ( " 1 * 7 = " , M )
        M ⟵ 2 * 7
        Ecrire ( " 2 * 7 = " , M )
        M ⟵ 3 * 7
        Ecrire ( " 3 * 7 = " , M )
        M ⟵ 4 * 7
        Ecrire ( " 4 * 7 = " , M )
        M ⟵ 5 * 7
        Ecrire ( " 5 * 7 = " , M )
        M ⟵ 6 * 7
        Ecrire ( " 6 * 7 = " , M )
        M ⟵ 7 * 7
        Ecrire ( " 7 * 7 = " , M )
        M ⟵ 8 * 7
        Ecrire ( " 8 * 7 = " , M )
        M ⟵ 9 * 7
        Ecrire ( " 9 * 7 = " , M )
        M ⟵ 10 * 7
        Ecrire ( " 10 * 7 = " , M )
Fin
```

Structures répétitives

La **structure répétitive** (**Boucle**) permet d'exécuter plusieurs fois une séquence d'instructions.

Dans une boucle, le nombre de répétitions peut être connu, **fixé à l'avance**, comme il peut **dépendre d'une condition** permettant l'arrêt et la sortie de cette boucle.

Boucle Pour ← Le nombre de répétitions peut être connu

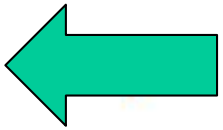
Boucle Tant que ← Le nombre de répétitions dépend d'une condition

Boucle Répéter ... Jusqu'à ...

1 - La boucle Pour

Cette boucle permet d'exécuter une séquence d'instructions un nombre de fois connu fixé à l'avance. Elle utilise une variable (indice) de contrôle d'itérations caractérisée par :

- sa valeur initiale,
- sa valeur finale,
- son pas de variation.



Structures répétitives: ***La Boucle Pour***

Syntaxe :

```
Pour I ← N à M pas 1 Faire
        Instructions
Fin Pour
```

I : variable
N : valeur initiale
M : valeur finale
Le pas de variation égale à 1

```
Algorithme table_7
Variables
        M , I : Entier
Début
        Pour I ← 0 à 10 pas 1 Faire
                M ← I * 7
                Ecrire ( I , " * 7 = " , M )
        Fin Pour
Fin
```

Structures répétitives: ***La Boucle Pour***

```
Algorithme table_7
Variables
        M , I : Entier
Début
        Pour I ⟵ 0 à 10 pas 1 Faire
                M ⟵ I * 7
                Ecrire ( I , " * 7 = " , M )
        Fin Pour
Fin
```

Solution avec une boucle

```
Algorithme table_7
Variables
        M : Entier
Début
        M ⟵ 0 * 7
        Ecrire ( " 0 * 7 = " , M )
        M ⟵ 1 * 7
        Ecrire ( " 1 * 7 = " , M )
        M ⟵ 2 * 7
        Ecrire ( " 2 * 7 = " , M )
        M ⟵ 3 * 7
        Ecrire ( " 3 * 7 = " , M )
        M ⟵ 4 * 7
        Ecrire ( " 4 * 7 = " , M )
```

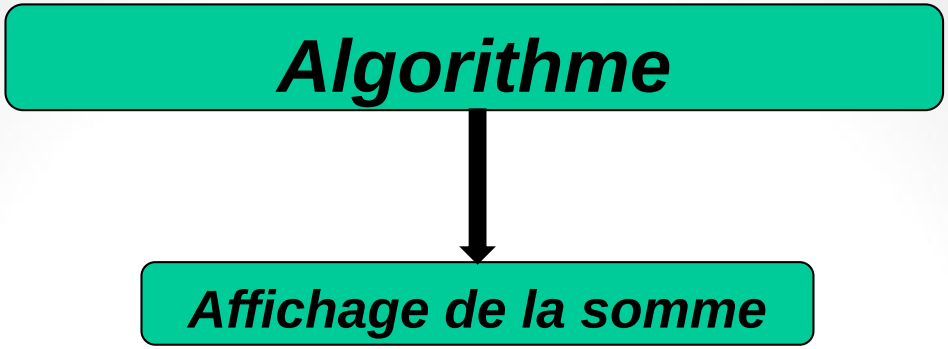
```
        M ⟵ 5 * 7
        Ecrire ( " 5 * 7 = " , M )
        M ⟵ 6 * 7
        Ecrire ( " 6 * 7 = " , M )
        M ⟵ 7 * 7
        Ecrire ( " 7 * 7 = " , M )
        M ⟵ 8 * 7
        Ecrire ( " 8 * 7 = " , M )
        M ⟵ 9 * 7
        Ecrire ( " 9 * 7 = " , M )
        M ⟵ 10 * 7
        Ecrire ( " 10 * 7 = " , M )
Fin
```

Solution sans utiliser les boucles

Structures répétitives: ***La Boucle Pour***

Ecrire un algorithme qui permet de calculer la somme des 20 premiers entiers positifs.

La somme des 20 premiers entiers positifs est: 210



Structures répétitives: ***La Boucle Pour***

Ecrire un algorithme qui permet de calculer la somme des 20 premiers entiers positifs.

```
Algorithme Somme
Variables
        S , I : Entier
Début
        S ← 0
        Pour I ← 1 à 20 pas 1 Faire
                S ← S + I
        Fin Pour
        Ecrire ( " La somme des 20 premiers entiers positifs est : " , S )
Fin
```

Structures répétitives: La boucle ***Tant que*** .... ***faire*** ....

0 * X = ?
1 * X = ?
2 * X = ?
3 * X = ?

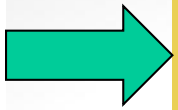


Table de multiplication

Veuillez saisir un nombre : -3

Veuillez saisir un nombre : 13

Veuillez saisir un nombre : 3

0 x 3 = 0	6 x 3 = 18
1 x 3 = 3	7 x 3 = 21
2 x 3 = 6	8 x 3 = 24
3 x 3 = 9	9 x 3 = 27
4 x 3 = 12	10 x 3 = 30
5 x 3 = 15	

Structures répétitives: La boucle ***Tant que*** .... ***faire*** ....

2 - La boucle Tant que

Cette boucle permet de répéter un bloc d'instructions tant qu'une **condition** est **vraie**.

Syntaxe :

```
Tant que Condition faire
        Instructions
Fin Tant que
```

Remarque :

La vérification de la condition s'effectue avant l'exécution des instructions. Celles-ci peuvent donc ne jamais être exécutées.

Structures répétitives: La boucle ***Tant que*** .... ***faire*** ....

```
Algorithme table_multiplication
Variables
        M, N , I : Entier
Début
        Ecrire ( " Entrez un nombre entre 1 et 10 : " )
        Lire ( N )
        Tant que N < 1 ou N > 10 faire
                Ecrire ( " Entrez un nombre entre 1 et 10 : " )
                Lire ( N )
        Fin Tant que
        Pour I ⟵ 0 à 10 pas 1 Faire
                M ⟵ I * N
                Ecrire ( I , " * " , N , " = " , M )
        Fin Pour
Fin
```

Structures répétitives: La boucle ***Tant que .... faire ....***

Ecrire un algorithme qui demande un nombre compris entre 10 et 20, jusqu'à ce que la réponse convienne. En cas de réponse supérieure à 20, on fera apparaître un message : « Plus grand ! », et inversement, « Plus petit ! » si le nombre est inférieur à 10.

Entrez un nombre entre 10 et 20 : 6
Plus petit !
Entrez un nombre entre 10 et 20 : -2
Plus petit !
Entrez un nombre entre 10 et 20 : 29
Plus grand !
Entrez un nombre entre 10 et 20 : 15
Bravo ! vous avez tapé un nombre compris entre 10 et 20

Structures répétitives: La boucle ***Tant que .... faire ....***

```
Algorithme nombre
Variables
        n : Entier
Début
        Ecrire ( " Entrez un nombre entre 10 et 20 : " )
        Lire ( n )
        Tant que n < 10 ou n > 20 faire
                si n < 10 alors
                        Ecrire ( " Plus petit ! " )
                Fin si
                si n > 20 alors
                        Ecrire ( " Plus grand ! " )
                Fin si
                Ecrire ( " Entrez un nombre entre 10 et 20 : " )
                Lire ( n )
        Fin Tant que
        Ecrire ( " Bravo ! vous avez un nombre compris entre 10 et 20 " )
Fin
```

3 - La boucle Répéter ... Jusqu'à ...

Cette boucle permet de répéter un bloc d'instructions jusqu'à ce qu'une **condition** soit **vérifiée**.

Syntaxe :

```
Répéter
        Instructions
Jusqu'à Condition
```

Remarque :

La vérification de la condition s'effectue après l'exécution des instructions. **Celles-ci sont donc exécutées au moins une fois**.

Structures répétitives: La boucle ***Répéter ... Jusqu'à ...***

```
Algorithme table_multiplication
Variables
        M , N , I : Entier
Début
        Répéter
                Ecrire ( " Entrez un nombre entre 1 et 10 : " )
                Lire ( N )
        Jusqu'à N > 0 et N <= 10
        Pour I ⟵ 0 à 10 pas 1 Faire
                M ⟵ I * N
                Ecrire ( I , " * " , N , " = " , M )
        Fin Pour
Fin
```

Structures répétitives: La boucle ***Répéter ... Jusqu'à ...***

La boucle répéter … jusqu'à …

```
Algorithme table_multiplication
Variables
        M , N , I : Entier
Début
        Répéter
                Ecrire ( " Entrez un nombre entre 1 et 10 : " )
                Lire ( N )
        Jusqu'à N > 0 et N <= 10
        Pour I ← 0 à 10 pas 1 Faire
                M ← I * N
                Ecrire ( I , " * " , N , " = " , M )
        Fin Pour
Fin
```

La boucle Tant que … faire …

```
Algorithme table_multiplication
Variables
        M , N , I : Entier
Début
        Ecrire ( " Entrez un nombre entre 1 et 10 : " )
        Lire ( N )
        Tant que N < 1 ou N > 10 faire
                Ecrire ( " Entrez un nombre entre 1 et 10 : " )
                Lire ( N )
        Fin Tant que
        Pour I ← 0 à 10 pas 1 Faire
                M ← I * N
                Ecrire ( I , " * " , N , " = " , M )
        Fin Pour
Fin
```

Structures répétitives: La boucle ***Répéter ... Jusqu'à ...***

Ecrire un algorithme qui demande a l'utilisateur de saisir un nombre entier strictement supérieur à 1, et qui calcule la somme des entiers jusqu'à ce nombre.

Par exemple, si l'on entre 5, le programme doit calculer : 1 + 2 + 3 + 4 + 5

```
Entrez un nombre : -13
Entrez un nombre : 1
Entrez un nombre : 0
Entrez un nombre : 7
La somme est : 28
```

Structures répétitives: La boucle ***Répéter ... Jusqu'à ...***

```
Algorithme somme
Variables
        I , N , S : Entier
Début
        Répéter
                Ecrire ( " Entrez un nombre : " )
                Lire ( N )
        Jusqu'à N > 1
        S ←— 0
        Pour I ←— 1 à N pas 1 Faire
                S ←— S + I
        Fin Pour
        Ecrire ( " La somme est : " , S )
Fin
```

Exercices

Exercice 1: Écrire un programme qui permet de calculer la valeur absolue d'un entier saisi par l'utilisateur.

Exercice 2: Écrire un algorithme qui permet d'échange le contenu de deux entiers A et B saisis par l'utilisateur. et afficher ces entiers après l'échange.

Exercice 3: Écrire un programme qui permet d'afficher si un nombre entier saisi au clavier est pair ou impair.

Exercices

Exercice 4: **Écrire un algorithme qui permet d'afficher le plus grand de trois entiers saisis au clavier.**

Exercice 5: **Écrire un programme qui demande l'âge d'un enfant et permet d'informer de sa catégorie sachant que les catégories sont les suivantes:**

- ➔**"poussin de 6 a 7 ans"**
- ➔**"pupille de 8 a 9 ans "**
- ➔**"minime de 10 a 11 ans "**
- ➔**" cadet après 12 ans ".**

Exercices

Exercice 6: **Écrire un programme permettant d'afficher le mois en lettre selon le numéro saisi au clavier. (Si l'utilisateur tape 1 le programme affiche janvier, si 2 affiche février , si 3 affiche mars...)**

Solutions➔ Exercices

Exercice 1

```
Algorithme La_valeur_absolue
Variables m :entier
Debut
    Ecrire("Entrer un nombre :")
    Lire(m)
    Si( m >= 0) alors
        Ecrire("La valeur absolue de",m,"est",m)
        SiNon
        Ecrire("La valeur absolue de",m,"est",-m)
        FinSi
Fin
```

Exercice 2

```
Algorithme  Echange
Variables  A,B,auxilaire :entiers
Debut
        Ecrire("Entrer un entier A:")
        Lire (A)
        Ecrire("Entrer un entier B:")
        Lire (B)

            auxilaire ← A
             A ← B
            B ← auxilaire
        Ecrire("le contenu de A est:",A)
        Ecrire("le contenu de B est:",B)
Fin
```

Exercice 3

```
Algorithme  Parité
Variables n :entier
Debut
  Ecrire("Entrer un entier:")
  Lire(n)
  Si(n mod 2 = 0) alors
     Ecrire(n,"est pair")
    SiNon
     Ecrire(n,"est impair")
  FinSi
Fin
```

Exercice 4

```
Algorithme  Maximum_Trois_nombres
Variables A,B,C,Max :entiers
Debut
  Ecrire(" Entrer A:")
  Lire(A)
  Ecrire(" Entrer B:")
  Lire(B)
  Ecrire(" Entrer C:")
  Lire(C)
    Max ← A
  Si (B >= Max) alors
                Max ← B
  FinSi
  Si(C >= Max) alors
     Max ← C
  FinSi
 Ecrire("Le Max est",Max)
Fin
```

Exercice 5

```
Algorithme
Variables age :réel
Debut
  Ecrire("Entrer votre age :")
  Lire(age)
   Si( age >= 5  et  age <= 7) alors
    Ecrire(" Vous etes poussin ")
            FinSi
   Si( age >= 8  et  age <= 9 ) alors
     Ecrire(" Vous etes pupille ")
   FinSi
   Si( age >= 10  et  age <= 11) alors
     Ecrire(" Vous etes minime ")
            FinSi
   Si ( age >= 12 ) alors
     Ecrire(" Vous etes Cadet ")
   FinSi
Fin
```

Exercice 6

```
Algorithme
Variables mois :entier
Debut
  Ecrire(" Entrer le numéro du mois:")
Lire (mois)
selon mois faire
            1: Ecrire (" janvier ")
                        …
                        …
            12: Ecrire ("décembre ")
 sinon
    Ecrire ("le numéro ne correspondant à aucun
mois")
FinSelon
Fin
```

Les Tableaux

Étudiants	Note
Ali	*18*
Ahmed	*15.5*
mouna	*08*
mustapha	*17*

```
Donner la note de l'étudiant num 1 : 10.5
Donner la note de l'étudiant num 2 : 8
Donner la note de l'étudiant num 3 : 13
Donner la note de l'étudiant num 4 : 14.5
```

```
Algorithme note
Variables
        N1 , N2 , N3 , ... , N317 , N318 : Réel
Début
        Ecrire ( " Donner la note de l'étudiant num 1  : " )
        Lire ( N1 )
        Ecrire ( " Donner la note de l'étudiant num 2 : " )
        Lire ( N2 )
        Ecrire ( " Donner la note de l'étudiant num 3 : " )
        Lire ( N3 )
        ... ... ... ...
        Ecrire ( " Donner la note de l'étudiant num 317 : " )
        Lire ( N317 )
        Ecrire ( " Donner la note de l'étudiant num 318 : " )
        Lire ( N318 )
Fin
```

Les Tableaux

Déclarer un Tableau | Initialisation d'un tableau | Accéder à un Élément | Afficher les Éléments | Remplir un Tableau

Syntaxe : `Tableau nom_tab ( Taille ) : Type`

Déclaration de la variable N de l'algorithme note : `Tableau N ( 318 ) : Réel`

Affectation

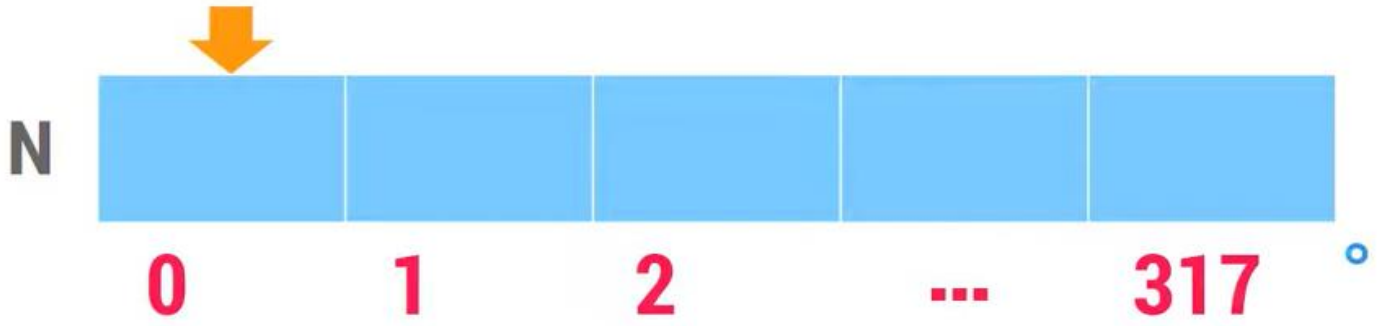
Syntaxe d'affectation: nom_tab (indice) ← Valeur

Syntaxe lecture: Lire (nom_tab (indice))

Syntaxe écriture: Ecrire (nom_tab (indice))

Affectation de la note 12 à l'étudiant num 1 :

N (0) ← 12



La lecture

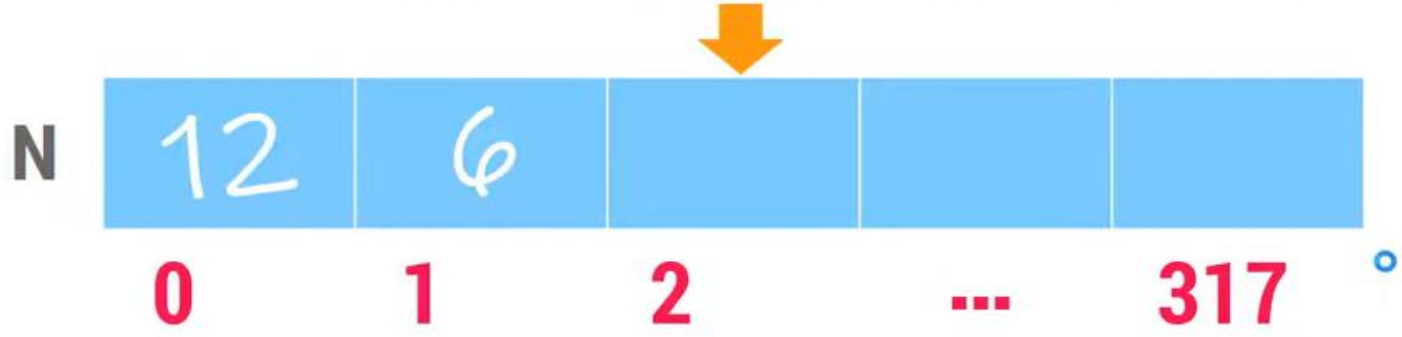
Syntaxe d'affectation: nom_tab (indice) ← Valeur

Syntaxe lecture: Lire (nom_tab (indice))

Syntaxe écriture: Ecrire (nom_tab (indice))

Utilisation de la lecteur pour saisir la note de l'étudiant num 3 :

Lire (N (2))



Écriture

Syntaxe d'affectation: nom_tab (indice) ← Valeur

Syntaxe lecture: Lire (nom_tab (indice))

Syntaxe écriture: Ecrire (nom_tab (indice))

Affichage de la note du dernier étudiant de la liste :

Ecrire (N (317))

Les Tableaux

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 indice.

Syntaxe générale

Syntaxe : `Tableau nom_tab ( Taille ) : Type`

Exemple : déclaration d'un tableau nommé T composé de cinq éléments entier :

`Tableau T ( 5 ) : Entier`

Le tableau est représenté schématiquement dans la mémoire comme suit :

Syntaxe d'affectation

Syntaxe d'affectation : `nom_tab ( indice ) ← Valeur`

Exemple : Affectation des valeurs aux éléments du tableau T:

```
T(0) ← 6
T(1) ← T(0) - 2
T(2) ← T(0) * T(1)
T(3) ← 20
T(4) ← T(0) + T(1) + T(2) + T(3)
```

Syntaxe de lecture

Syntaxe de lecteur : `Lire ( nom_tab ( indice ) )`

Exemple : Remplissage de tous les éléments du tableau T avec l'instruction Lire :

```
Pour i ← 0 à 4 pas 1 Faire
        Lire ( T ( i ) )
Fin Pour
```

Syntaxe d'écriture

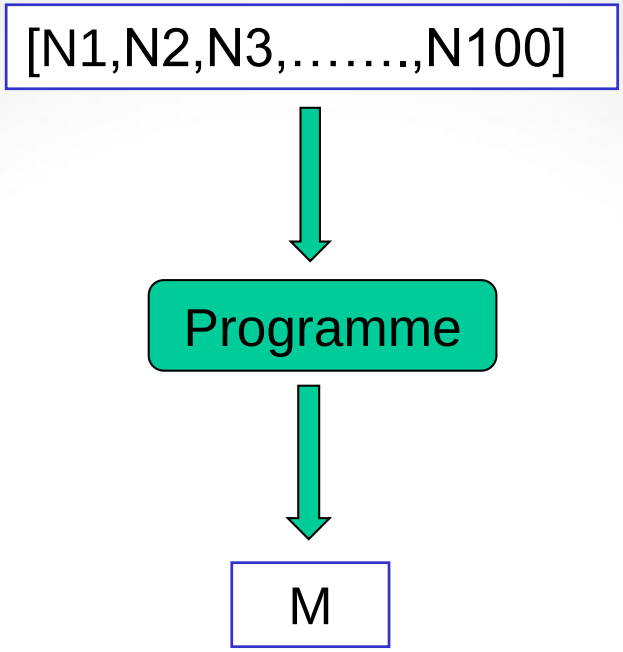
Syntaxe d'écriture: `Ecrire ( nom_tab ( indice ) )`

Exemple : Affichage des valeurs de tous les éléments du tableau T :

```
Pour i ← 0 à 4 pas 1

        Ecrire ( " L'élément " , i + 1 , " du tableau T est : " , T ( i ) )

Fin Pour
```

EXERCICE

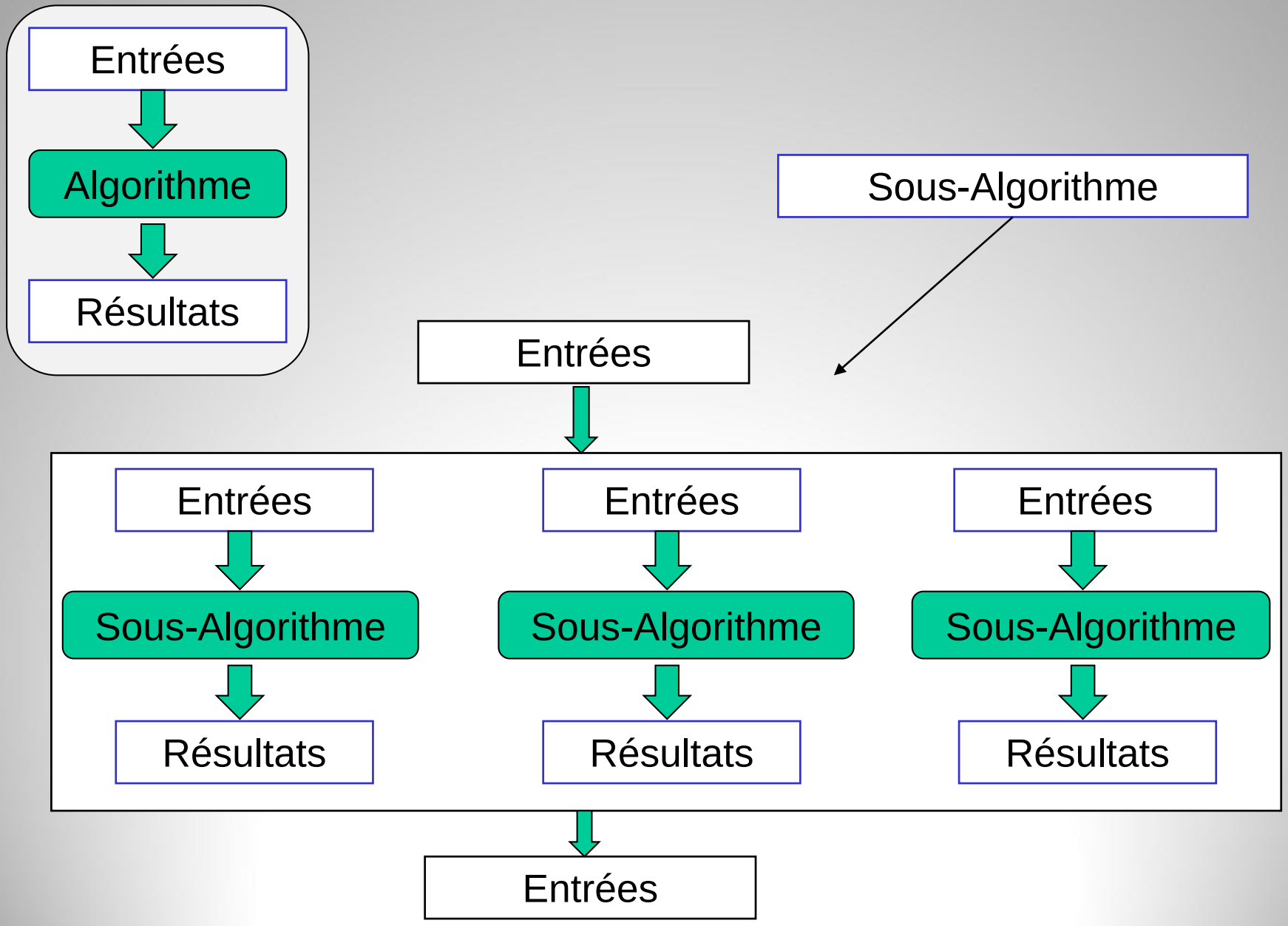
Écrire un programme qui permet de demander à l'utilisateur de saisir les notes des étudiants (318 étudiants), puis le programme calcule et affiche la moyenne des notes.



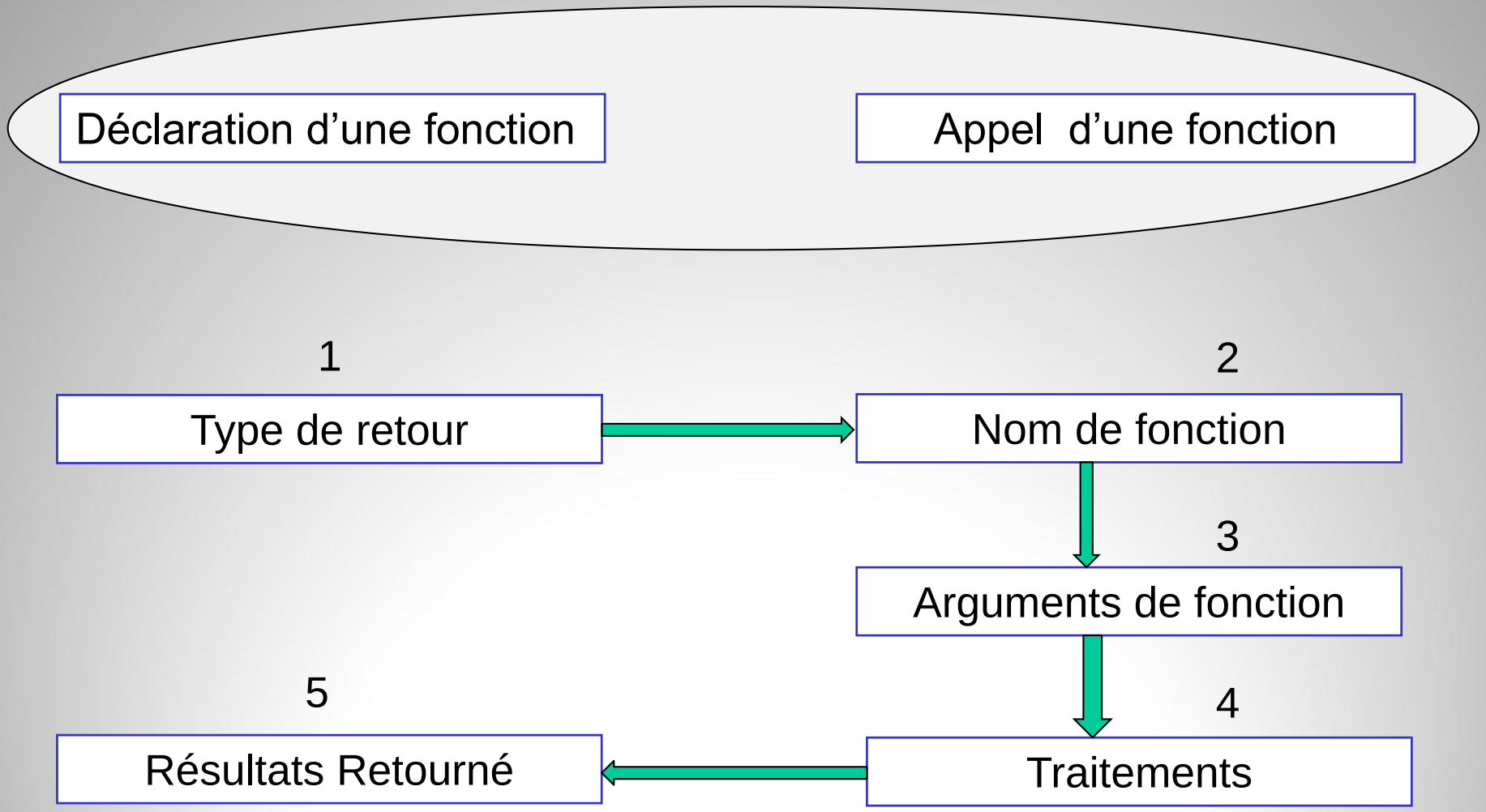
Correction ➔ Exercice

```
Algorithme moyenne_note
Variables
    Tableau N(318) : réel
    M , S : réel
    i : entier
Début
    Pour i <-- 0 à 317 pas 1 faire
        Ecrire (" Donner la note de l'étudiant num : ", i+1 , " : ")
        Lire( N( i ))
    fin Pour
    S <-- 0
    Pour i <-- 0 à 317 pas 1 faire
        S <-- S + N ( i )
    fin Pour
    M <-- S / 318
    Ecrire ( " La moyenne des notes est : " , M )
Fin
```

Les Fonctions



Les Fonctions

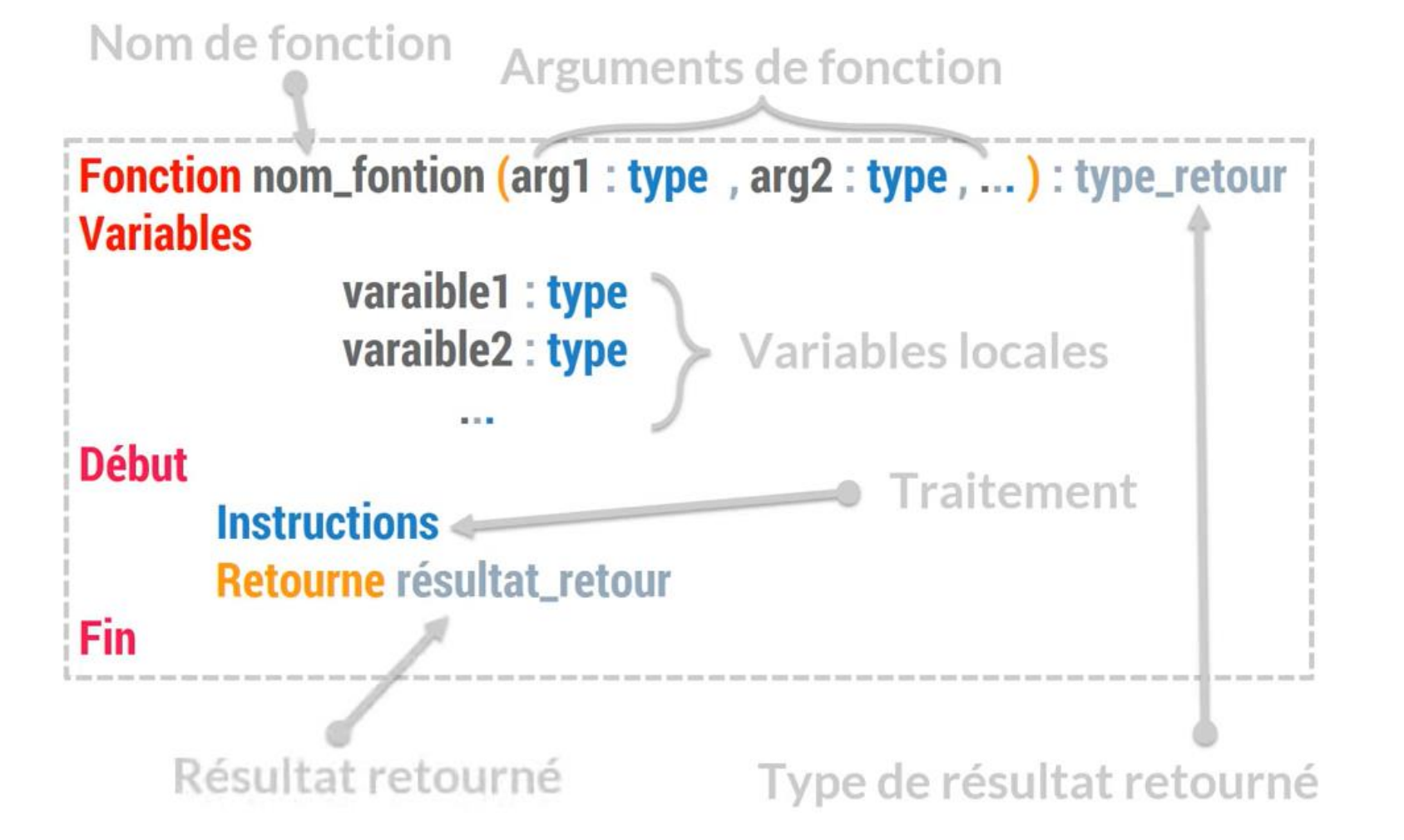


Définition

Une fonction est **une suite d'instructions** regroupées sous un nom; elle prend en entrée des **paramètres (arguments)** et retourne un **résultat**.

Une fonction est **écrite séparément** du corps de l'algorithme principal et sera **appelée** par celui-ci lorsque cela sera nécessaire.

Déclaration de fonction



Exemple

```
Fonction puissance ( N : Entier ) : Entier
Variables
                P : Entier
Début
        P ⟵ N ^ 2
        Retourne P
Fin
```

Appel de la fonction

Méthode 1 :

```
nom_var ← nom_fontion (arg1 , arg2 , ... )
```

Méthode 2 :

```
nom_var1 ← nom_var2 / nom_fontion (arg1 , arg2 , ... )
```

Méthode 3 :

```
Ecrire ( nom_fontion (arg1 , arg2 , ... ) )
```

Exemple

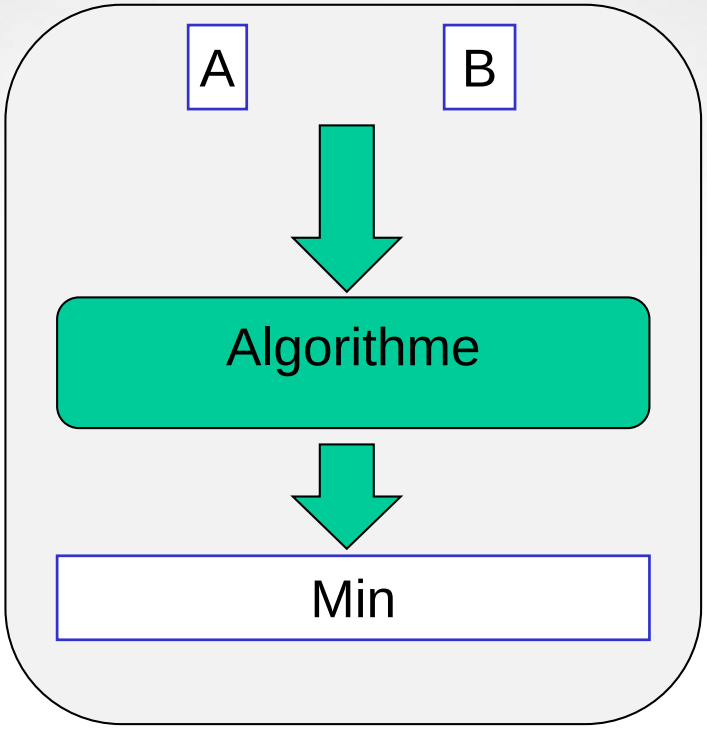
```
Algorithme exemple_fonction
        Fonction puissance ( N : Entier ) : Entier
        Variables
                P : Entier
        Début
                P ← N ^ 2
                Retourne P
        Fin
        Variables
                x : Entier
Début
        Ecrire ( " Veuillez saisir un nombre : " )
        Lire ( x )
        Ecrire ( " La puissance du nombre " , x , " est : " , puissance ( x ) )
Fin
```

Déclaration de la fonction

L'appel de la fonction

Exercice

Ecrire un algorithme qui permet de définir et d'appeler une fonction *Minimum* qui renvoie le plus petit de deux nombres différents.



Correction ➔ Exercice

```
Algorithme exercice_fonction
    Fonction minimum (x : entier, y : entier) : entier
        Variable
            min : entier
        Début
            Si x >= y alors
                min <-- y
            Sinon
                min <-- x
            fin Si
            Retourne min
        Fin
    Variables
        x , y : entier
Début
    Ecrire(" Veuillez entrer deux nombre : ")
    Lire(x,y)
    Ecrire("Le minimum est :" , minimum(x,y))
Fin
```

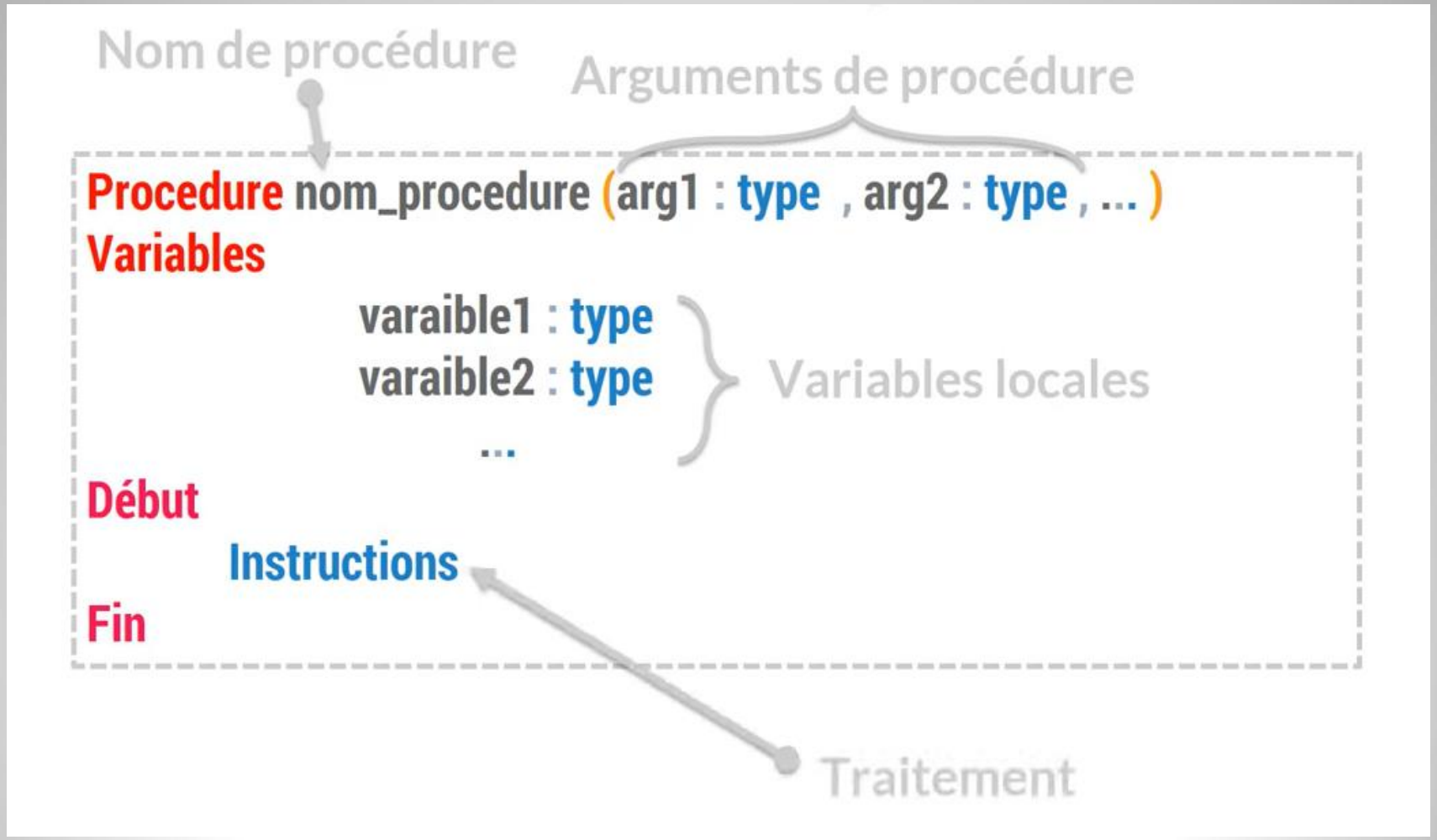
Les Procédures

Définition

Une procédure est **une suite d'instructions** regroupées sous un nom; elle prend en entrée des **paramètres** (**arguments**) mais qui **ne retourne rien**.

Une procédure est écrite séparément du corps de l'algorithme principal et sera **appelée** par celui-ci lorsque cela sera nécessaire.

Déclaration de la procédure



Exemple ➔ procédure puissance

```
Procdure puissance ( N : Entier )
Variables
            P : Entier
Début
        P ⟵ N ^ 2
        Ecrire ( " La puissance du nombre " , N , " est : " , P )
Fin
```

Exemple ➔ procédure puissance

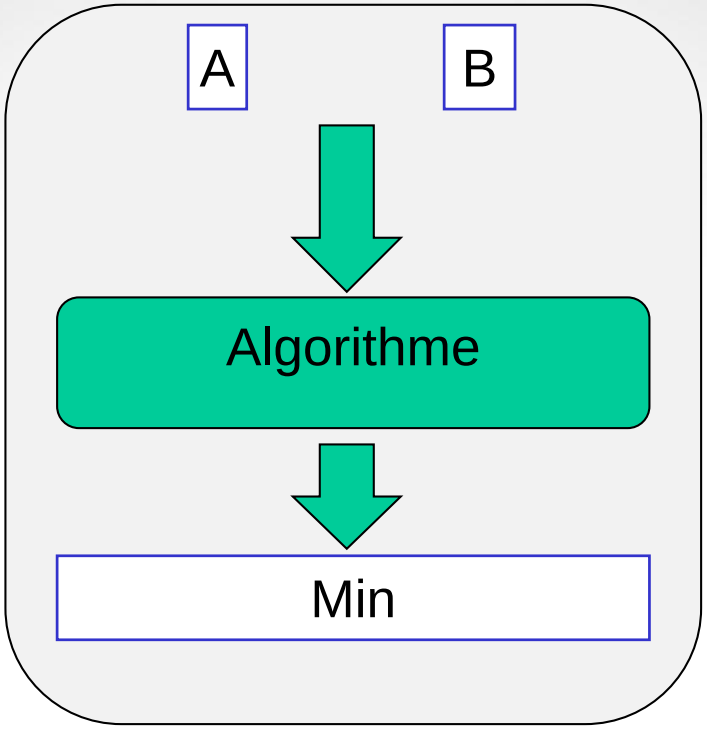
```
Algorithme exemple_procedure
        Procedure puissance ( N : Entier )
                Variables
                        P : Entier
                Début
                        P ⟵ N ^ 2
                        Ecrire ( " La puissance du nombre " , N , " est : " , P )
                Fin
        Variables
                x : Entier
Début
        Ecrire ( " Veuillez saisir un nombre : " )
        Lire ( x )
        puissance ( x )
Fin
```

Déclaration de la procédure

L'appel de la procédure

Exercice

Ecrire un algorithme qui permet de définir et d'appeler une procédure minimum qui renvoie le plus petit de deux nombres différents.



Correction ➔ Exercice

```
Algorithme exercice_procedure
    Procedure minimum (x : entier, y : entier)
        Variable
            min : entier
        Début
            Si x >= y alors
                min <-- y
            Sinon
                min <-- x
            fin Si
            Ecrire("Le minimum est :" , min)
        Fin
    Variables
        x , y : entier
Début
    Ecrire(" Veuillez entrer deux nombre : ")
    Lire(x,y)
    minimum(x,y)
Fin
```

Tableaux à deux dimensions - Matrices

Introduction

Tableau Analyse (5) : Réel

Tableau Algebre (5) : Réel

Tableau Statistiques (5) : Réel

Etudiant	Analyse	Algèbre	Statistiques	Moyenne
Rania	10.5	18	11.5	13.33
Adil	8	6	9	7.66
Manal	13	15	10	12.66
Walid	14.5	3	7	8.16
Amine	16	14	19	16.33
Max	16	18	19	
Min	8	3	7	

Déclarer un Tableau

Accéder à un Élément

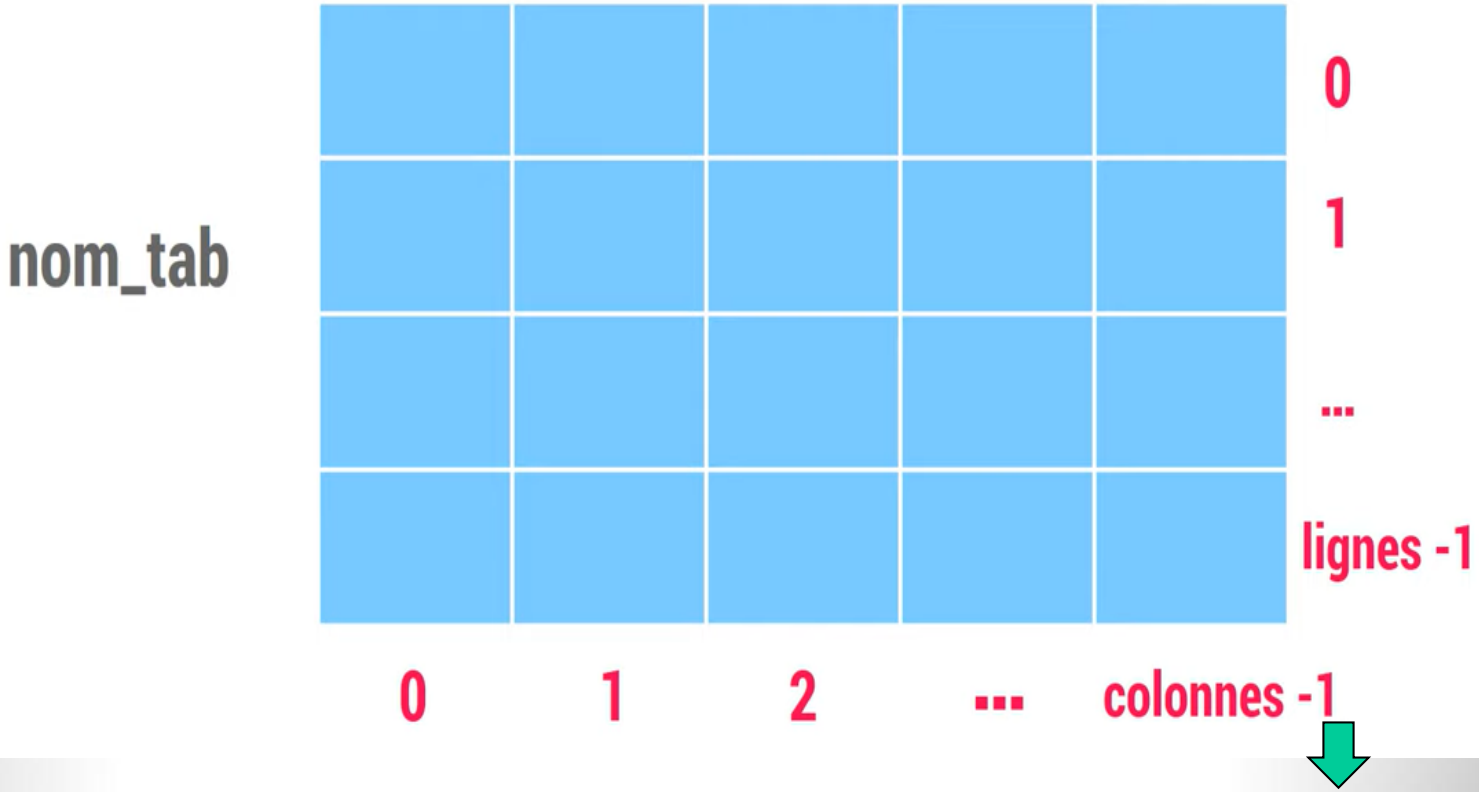
Afficher les Éléments

Remplir un Tableau

Tableaux à deux dimensions - Matrices

Déclaration d'un tableau à deux dimensions

Syntaxe : Tableau nom_tab (lignes , colonnes) : Type

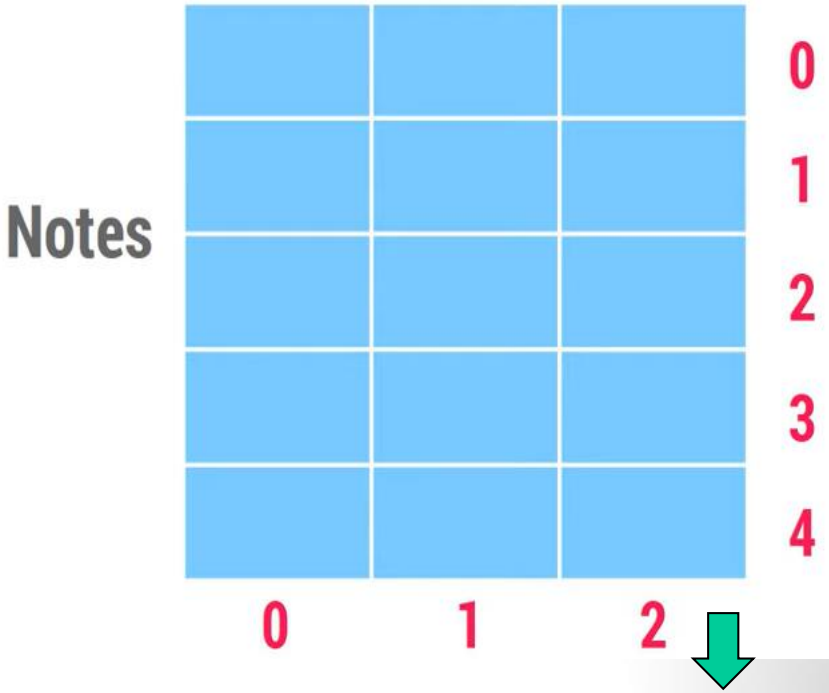


Tableaux à deux dimensions - Matrices

Déclaration d'un tableau à deux dimensions

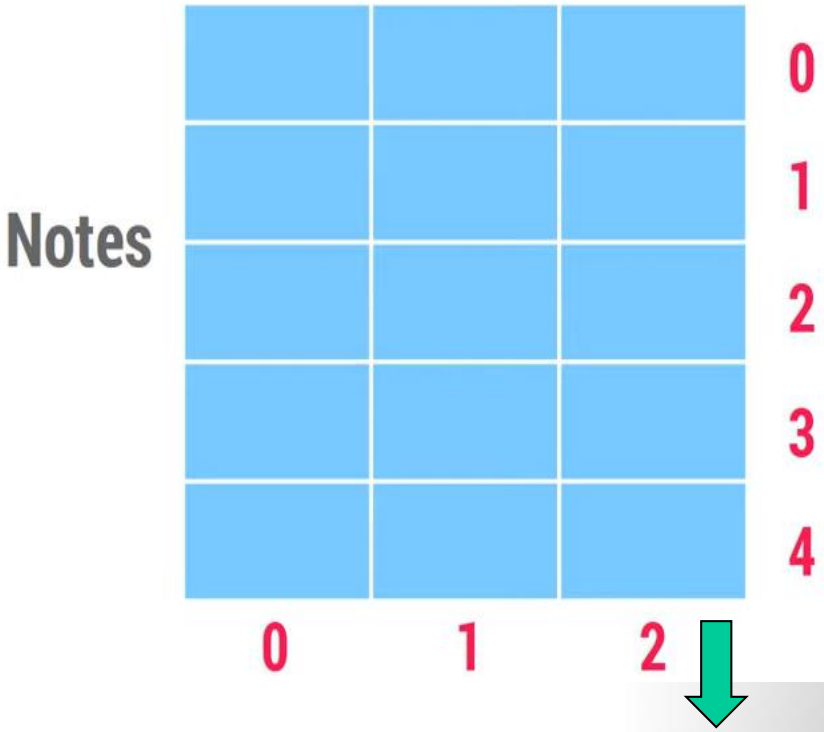
Exemple : déclaration du tableau notes

```
Tableau Notes ( 5 , 3 ) : Réel
```



Accéder à un élément

Syntaxe d'affectation: nom_tab (num_lig , num_col) ← Valeur

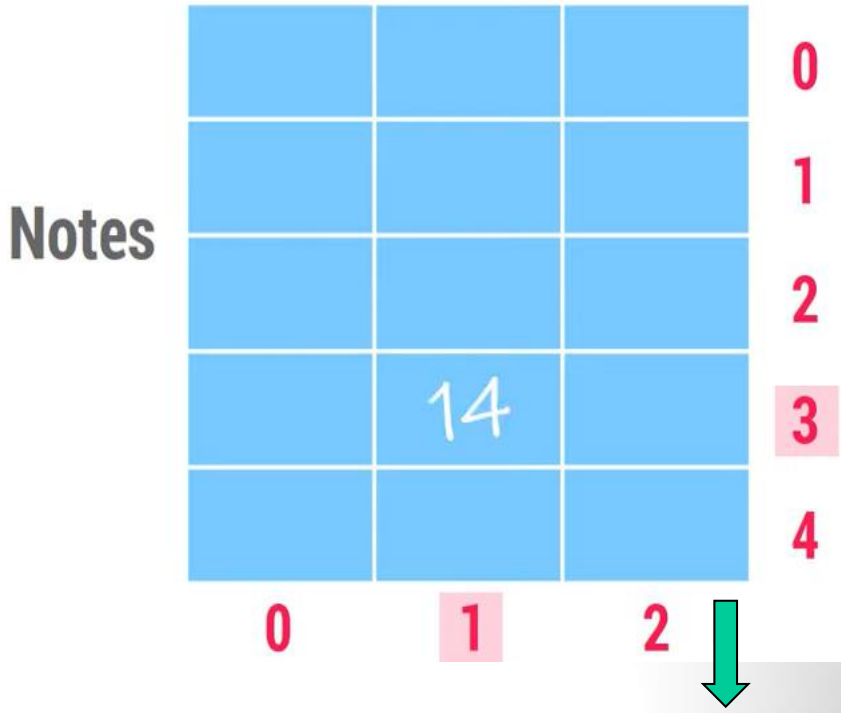


Accéder à un élément

Syntaxe d'affectation: `nom_tab ( num_lig , num_col ) ← Valeur`

Exemple : Affectation de la note 14 à l'étudiant num 4 dans la matière num 2 :

`Notes ( 3 , 1 ) ← 14`

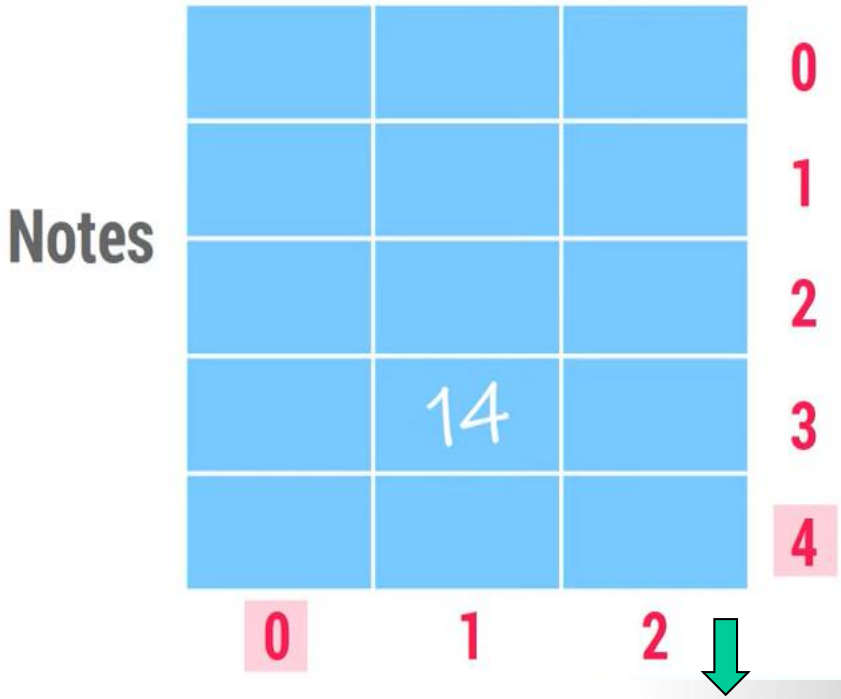


Accéder à un élément

Syntaxe lecture : `Lire ( nom_tab ( num_lig , num_col ) )`

Exemple : Utilisation de la lecteur pour saisir la note de l'étudiant num 5 dans la matière num 1 :

```
Lire ( Notes ( 4 , 0 ) )
```



Accéder à un élément

Syntaxe écriture : `Ecrire ( nom_tab ( num_lig , num_col ) )`

Exemple : Affichage de la note du premier étudiant de la liste obtenue en troisième matière

```
Ecrire ( Notes ( 0 , 2 ) )
```

Notes

10.5	18	11.5	0
8	6	9	1
13	15	10	2
14.5	14	7	3
16	14	19	4
0	1	2	

Tableaux à deux dimensions - Matrices

Remplir tous les éléments

```
Pour i ⟵ 0 à lignes-1 pas 1 Faire
    Pour j ⟵ 0 à colonnes-1 pas 1 Faire

        Lire ( T ( i , j ) )

    Fin Pour
Fin Pour
```

Afficher tous les éléments

```
Pour i ⟵ 0 à lignes-1 pas 1 Faire
  Pour j ⟵ 0 à colonnes-1 pas 1 Faire

    Ecrire (" L'élément " , i + 1 , j + 1 , " du tableau T est : " , T ( i , j ) )

  Fin Pour
Fin Pour
```

Exercice

Écrire un algorithme qui permet de demander à l'utilisateur de saisir les notes des étudiants (5 étudiants) dans chaque matière (3 matières), puis l'algorithme calcule et affiche la moyenne de chaque étudiant.

Etudiant	Analyse	Algèbre	Statistiques	Moyenne
Rania	10.5	18	11.5	13.33
Adil	8	6	9	7.66
Manal	13	15	10	12.66
Walid	14.5	3	7	8.16
Amine	16	14	19	16.33

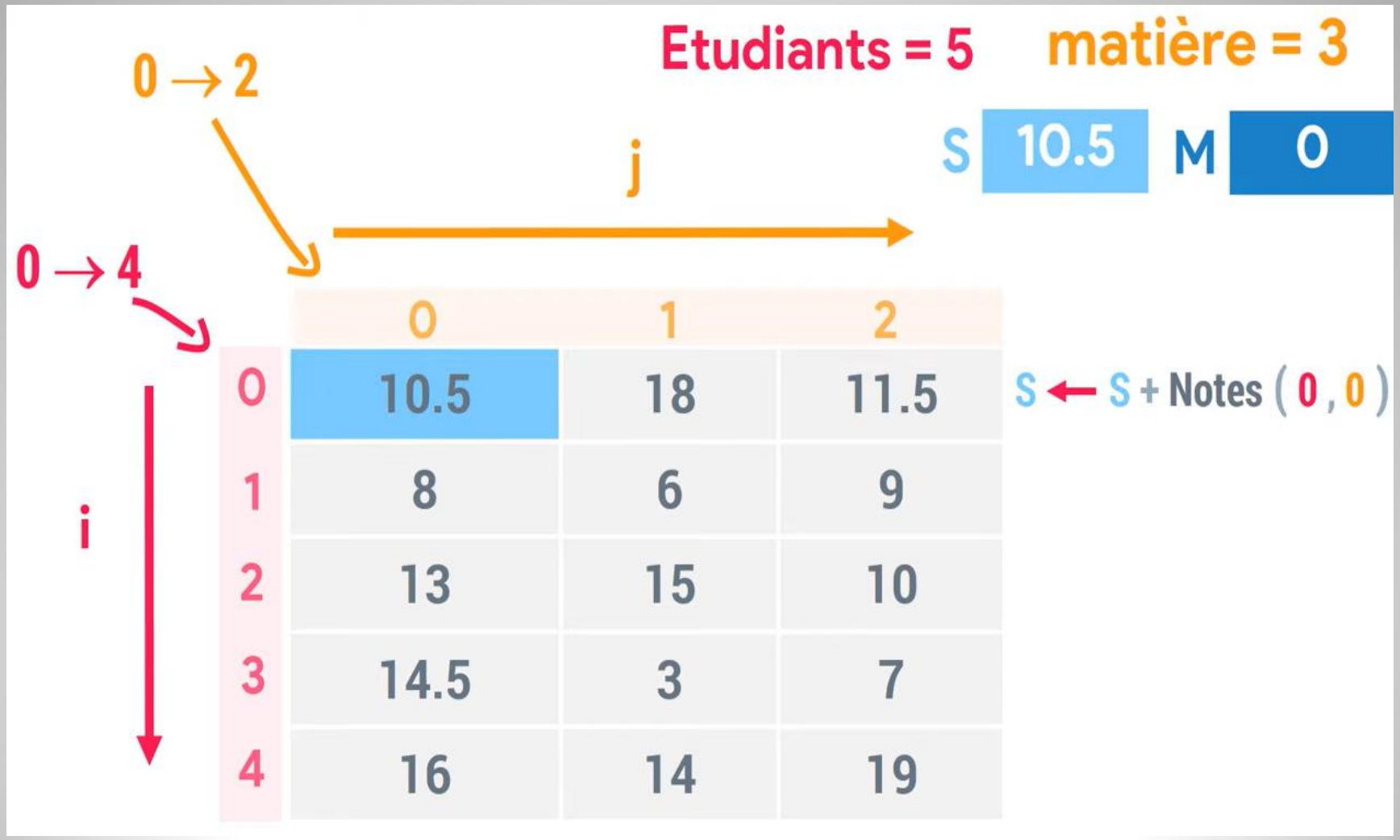
Exercice

Donner la note de l'étudiant num 1 dans la matière num 1 : 10.5
Donner la note de l'étudiant num 1 dans la matière num 2 : 18
Donner la note de l'étudiant num 1 dans la matière num 3 : 11.5
Donner la note de l'étudiant num 2 dans la matière num 1 : 8
...
Donner la note de l'étudiant num 5 dans la matière num 3 :

La moyenne de l'étudiant num 1 est : 13.33
La moyenne de l'étudiant num 2 est : 7.66
La moyenne de l'étudiant num 3 est : 12.66
La moyenne de l'étudiant num 4 est : 8.16
La moyenne de l'étudiant num 5 est : 16.33

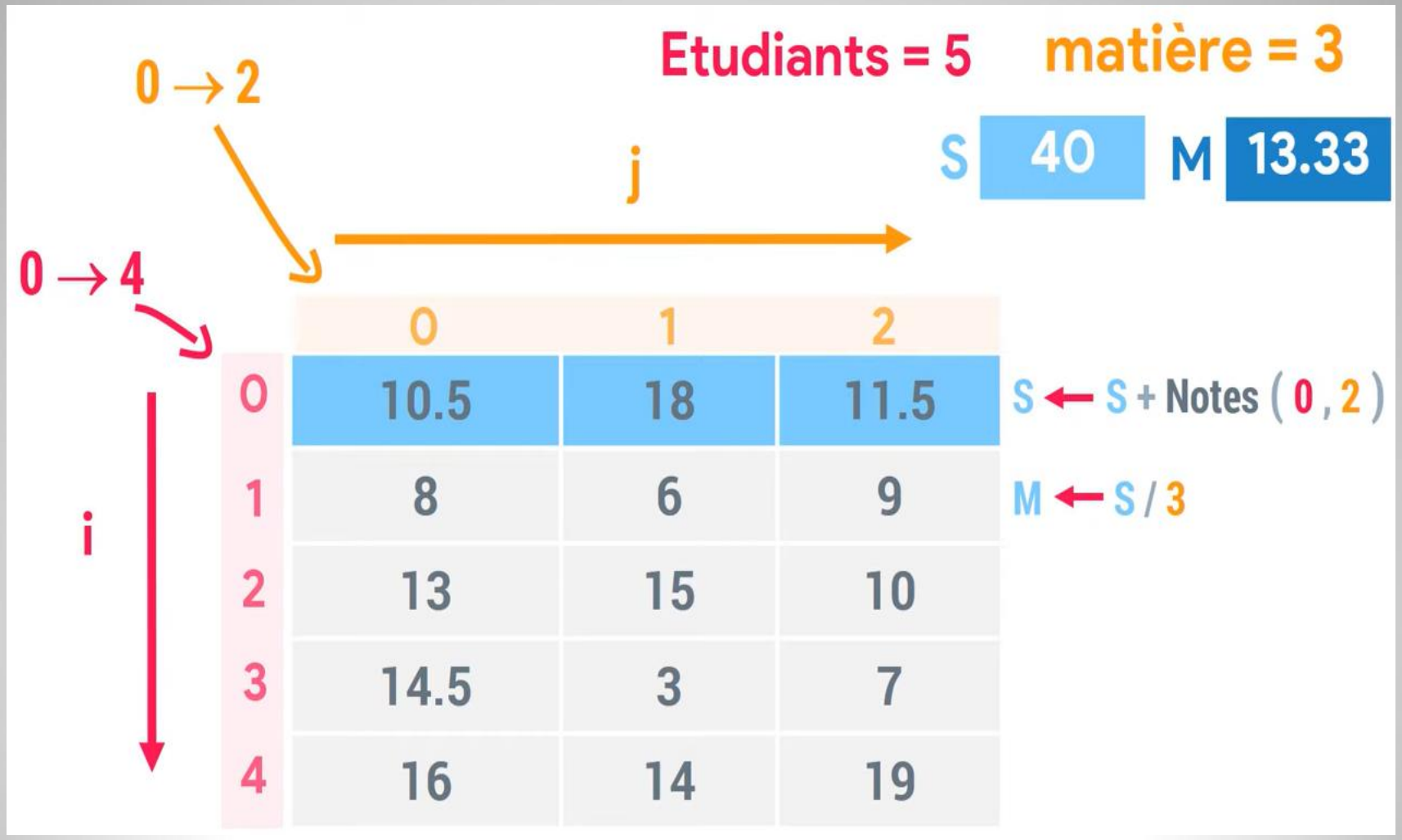
Tableaux à deux dimensions - Matrices

Explication ➔ Exercice



Tableaux à deux dimensions - Matrices

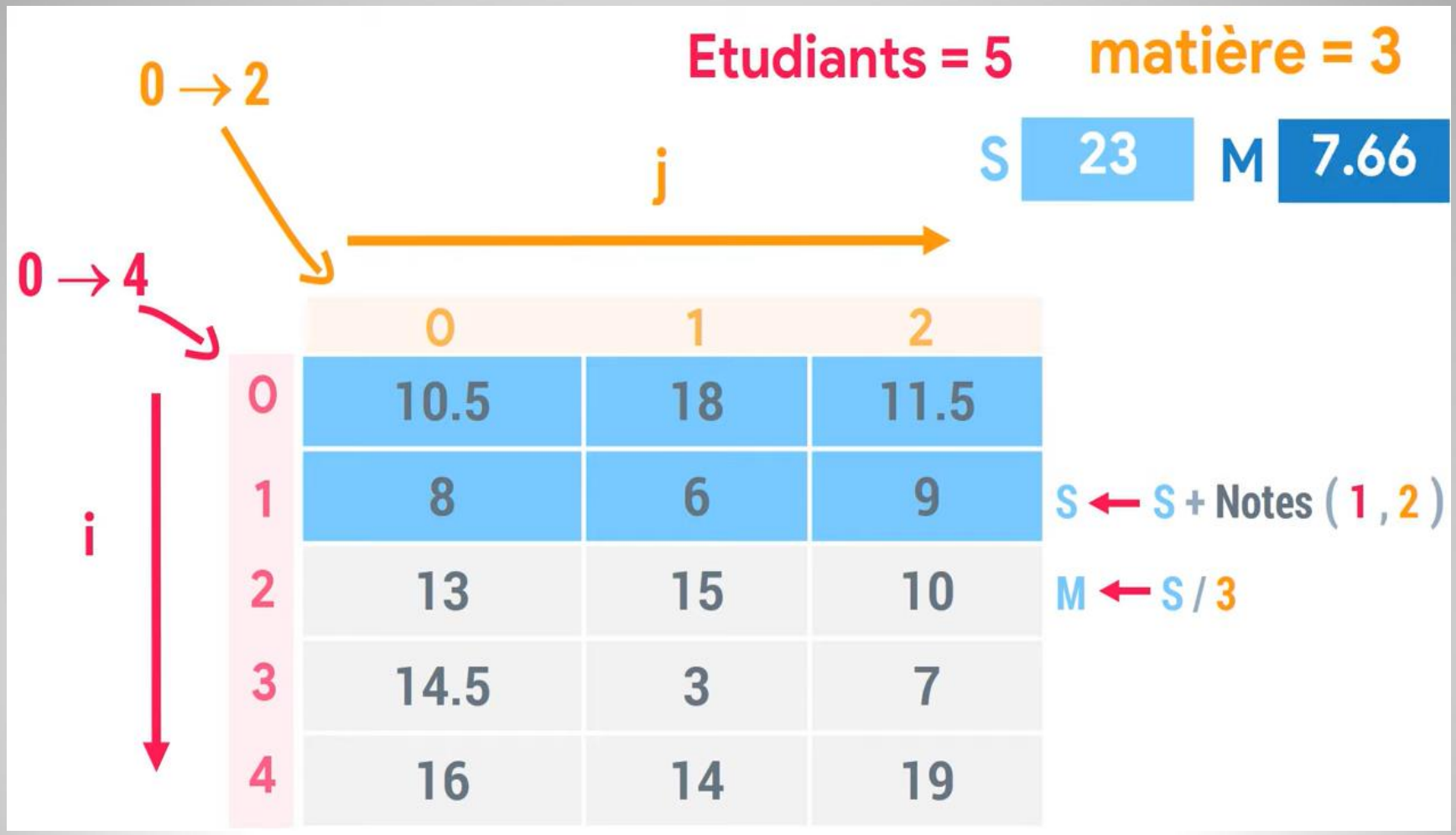
Explication ➔ Exercice



	0	1	2
0	10.5	18	11.5
1	8	6	9
2	13	15	10
3	14.5	3	7
4	16	14	19

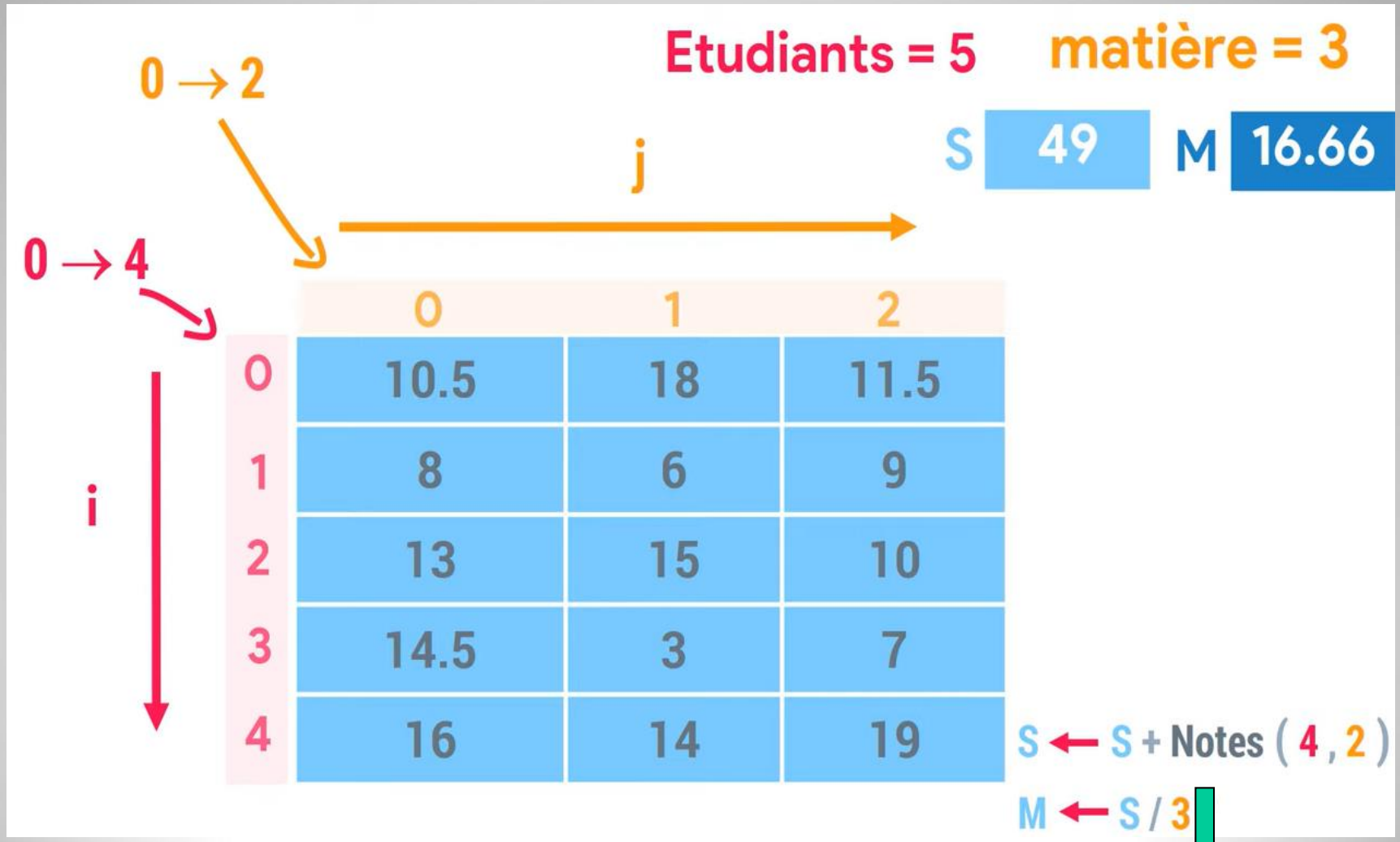
Tableaux à deux dimensions - Matrices

Explication ➔ Exercice



Tableaux à deux dimensions - Matrices

Explication ➔ Exercice



Tableaux à deux dimensions - Matrices

Correction ➔ Exercice

```
Algorithme moyenne_notes
Variables
    Tableau Notes ( 5 , 3 ) : réel
    M , S : réel
    i , j : entier
Début
    Pour i <-- 0 à 4 pas 1 Faire
        Pour j <-- 0 à 2 pas 1 Faire
            Ecrire ( " Donner la noe de l'étudiant num : " , i + 1 , " dans la matière
            Lire ( Notes ( i , j ) )
        fin Pour
    fin Pour
    M <-- 0
    S <-- 0
    Pour i <-- 0 à 4 pas 1 Faire
        Pour j <-- 0 à 2 pas 1 Faire
            S <-- S + Notes ( i , j )
        fin Pour
        M <-- S / 3
        Ecrire ( " La moyenne de l'étudiant num : " , i + 1 , " est : " , M )
        S <-- 0
    fin Pour
Fin
```