



Pr. Youness KHOURLIFI, PhD en Informatique  
Professeur à la Faculté Polydisciplinaire – Khouribga –  
Université Sultan Moulay Slimane – Béni Mellal –  
**Consultant IT** : SQL 2016 Database Administration, Core  
Infrastructure 2016, Azure Solutions Architect Expert,  
Data Analyst Associate, Ingénieur DevOps.  
[y.khourdifi@usms.ma](mailto:y.khourdifi@usms.ma)

# ALGORITHMIQUE ET LA PROGRAMMATION EN LANGAGE C



## Règle du cours

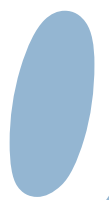
- ❑ Respect de l'horaire
- ❑ Respect des deadlines de remises des comptes rendus  
Respect des deadlines de remises des projets

# Systeme de notation

- ❑ Contrôles continus → 50%
- ❑ Travaux pratiques → 30%
- ❑ Mini-Projets → 20%

- Introduction générale
- Introduction à l'algorithmique :
  - Les objectifs ;
- Résolution des problèmes informatiques ;
- Les bases d'algorithmiques ;
- Types de base, variables, constantes ;
- Les Opérateurs et expressions ;
- Lecture/écriture et Les entrées sorties en C ;
- Les structures de contrôle ;
- Les tableaux ;
- Les pointeurs ;
- Conclusion et Evaluation.

**Chapitre**



**Introduction générale**

# Introduction générale :

6



Comment les ordinateurs font-ils pour résoudre des problèmes ?



Comment votre petit GPS parvient-il à déterminer, parmi la multitude d'itinéraires possibles, le chemin le plus rapide vers votre destination , et cela en quelques secondes ?



Lors d'un achat sur Internet, comment votre numéro de carte bancaire est-il protégé contre quiconque l'intercepterait ?

**La réponse à ces questions, comme à de nombreuses autres, tient dans les algorithmes. .**

# Introduction à l'algorithmique :

7

## Algorithme



```
graph LR; A([Algorithme]) --- B[Est une séquence d'opérations visant à la résolution d'un problème en un temps fini ;]; A --- C[Est écrit avec le langage humain, et ensuite traduit dans un langage de programmation ;]; A --- D[Va nous permettent d'automatiser certaines tâches qui reviennent sans cesse.];
```

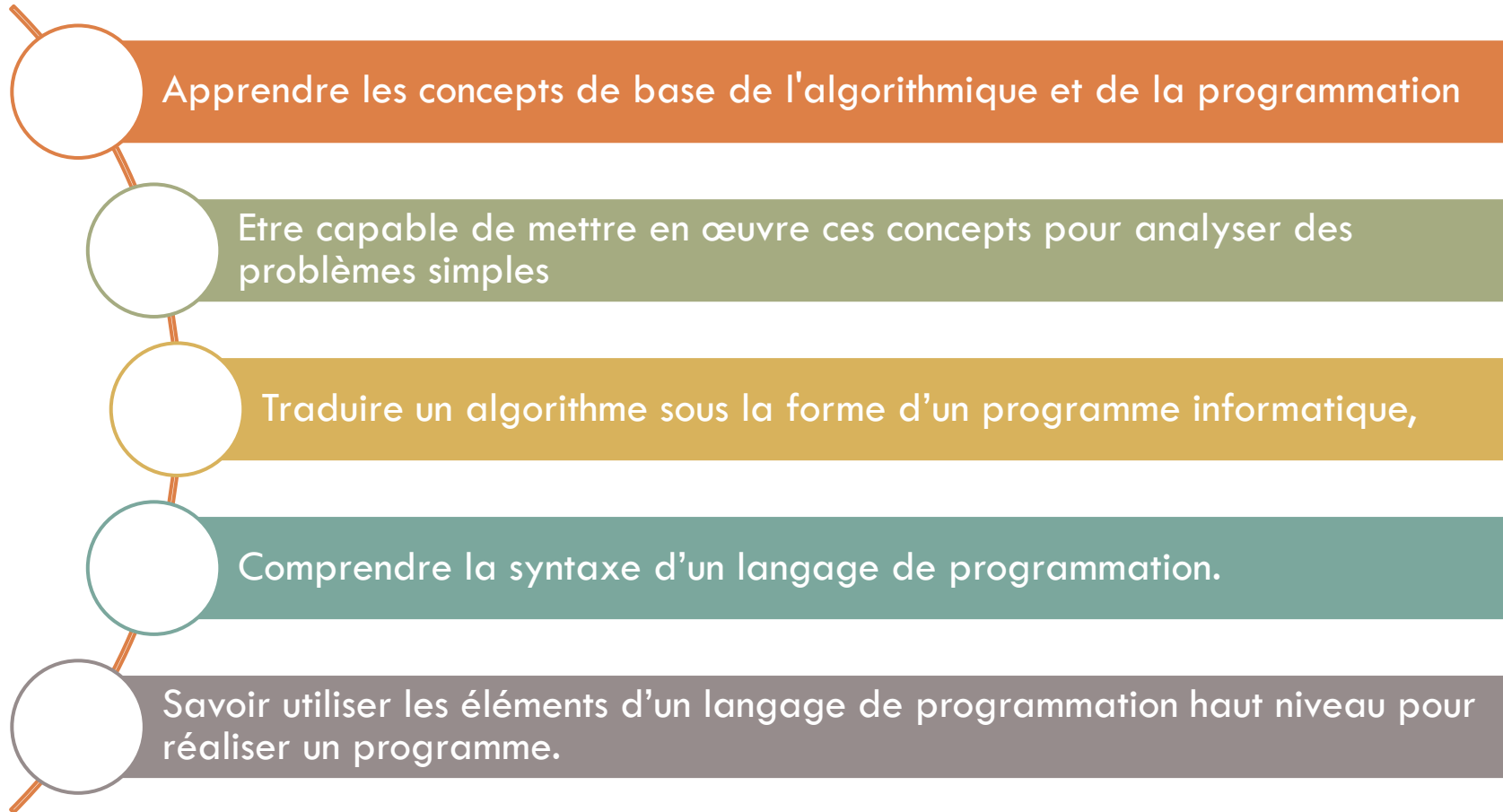
Est une séquence d'opérations visant à la résolution d'un problème en un temps fini ;

Est écrit avec le langage humain, et ensuite traduit dans un langage de programmation ;

Va nous permettent d'automatiser certaines tâches qui reviennent sans cesse.

# Les objectifs :

8

- 
- Apprendre les concepts de base de l'algorithmique et de la programmation
  - Etre capable de mettre en œuvre ces concepts pour analyser des problèmes simples
  - Traduire un algorithme sous la forme d'un programme informatique,
  - Comprendre la syntaxe d'un langage de programmation.
  - Savoir utiliser les éléments d'un langage de programmation haut niveau pour réaliser un programme.



## Chapitre

## Généralités

# I. 1. Résolution des problèmes informatiques :

10

- ❑ Un ordinateur **n'est qu'une machine** capable d'exécuter automatiquement une série d'opérations simples qu'on lui a demandé de faire.
  - ❑ Alors, son intérêt réside dans sa capacité de manipuler rapidement et sans erreur un grand nombre d'informations.
- ❑ Certains voient, à tort, dans l'ordinateur une machine pensante et intelligente, capable de résoudre bien des problèmes.
  - ❑ Celui-ci serait capable de rien si quelqu'un (le programmeur) ne lui avait fourni la liste des actions à exécuter.
- ❑ Les opérations élémentaires que peut exécuter un ordinateur sont en nombre restreint et doivent être communiquées de façon précise dans un langage qu'il comprendra.
- ❑ Donc, le rôle principal de l'utilisateur est de décrire la suite des actions élémentaires permettant d'obtenir, à partir des données fournies, les résultats escomptés.

## I. 2. Comment résoudre un problème informatique ?

11

- ❑ **Analyser le problème :** Définir avec précision les résultats à obtenir, les informations d'entrée dont on dispose.
- ❑ **Déterminer les méthodes de résolution:** Trouver une solution efficace du problème en déterminant la suite des opérations à effectuer pour atteindre le résultat voulu.



**Cette suite d'opérations constitue un Algorithme**

- ❑ **Formuler un algorithme définitif:** Faciliter la résolution sur ordinateur par l'expression de l'algorithme dans un formalisme adéquat
- ❑ **Traduire l'algorithme:** Réécrire l'algorithme dans un langage de programmation

## I. 3. Du problème au programme :

12

Décomposer un problème complexe en plusieurs sous-problèmes plus simples

Décrire à la machine la suite des actions élémentaires qu'il faut exécuter afin d'obtenir les résultats attendus

Envisager le moindre détail prévoir les diverses possibilités de données

Cette démarche de trois étapes porte le nom **d'Algorithme**

## I . 4. Méthodes de résolution d'un problème :

13

**Analyse:** c'est la phase d'étude, elle consiste à répondre aux 3 questions; (1) qu'est ce que j'ai? (2) qu'est ce que je veux? et (3) comment faire?

**Reconstruction:** c'est la phase de spécifications des données et du description du traitement.

**Codage:** c'est la phase de traduction de l'algorithme en un programme informatique via un langage de programmation.

ICI

null

## I . 5. Les bases d'algorithmiques :

14

- ❑ L'algorithmique est un terme d'origine arabe, comme algèbre.
- ❑ Le mot "algorithme" viendrait du nom AL- KHOWÂRIZMI .
- ❑ Un algorithme, est un Processus décrivant étape par étape comment résoudre un problème ;
  - ❑ Indiqué un chemin à quelqu'un ?
  - ❑ Faire chercher un objet à quelqu'un par téléphone ?
- ❑ Si l'algorithme est juste, le résultat est le résultat voulu,
- ❑ Si l'algorithme est faux, le résultat est aléatoire.



~780-850

## I . 5. Les bases d'algorithmiques :

15

- ❑ **L'algorithmique** est la discipline qui s'intéresse aux algorithmes.
- ❑ **Un algorithme** est une suite logique d'opérations dites instructions, qui, une fois exécuté correctement, donne le résultat attendu.
- ❑ **Exemple:** Préparer les spaghettis
  - ❑ Prendre une casserole;
  - ❑ La remplir de l'eau;
  - ❑ La mettre sur le feu;
  - ❑ Quand l'eau devient bouillante, mettre les spaghettis;
  - ❑ Attendre 5 minutes puis les retirer,



Il s'agit d'un algorithme

## I . 5. Les bases d'algorithmiques :

16

### Définitions :

- ❑ « Un Algorithme est **une spécification d'un schéma de calcul**, sous forme d'une suite finie d'opérations élémentaires obéissant à un enchaînement déterminé ».
- ❑ « Un Algorithme est **un ensemble de règles opératoires** dont l'application permet de résoudre un problème donné au moyen d'un nombre fini d'opérations ».
- ❑ « Un Algorithme est **une suite finie d'instructions "non ambiguë" qui décrit un traitement** sur un nombre fini de données structurées, et qui se termine après un nombre fini d'opérations pour donner un résultat »



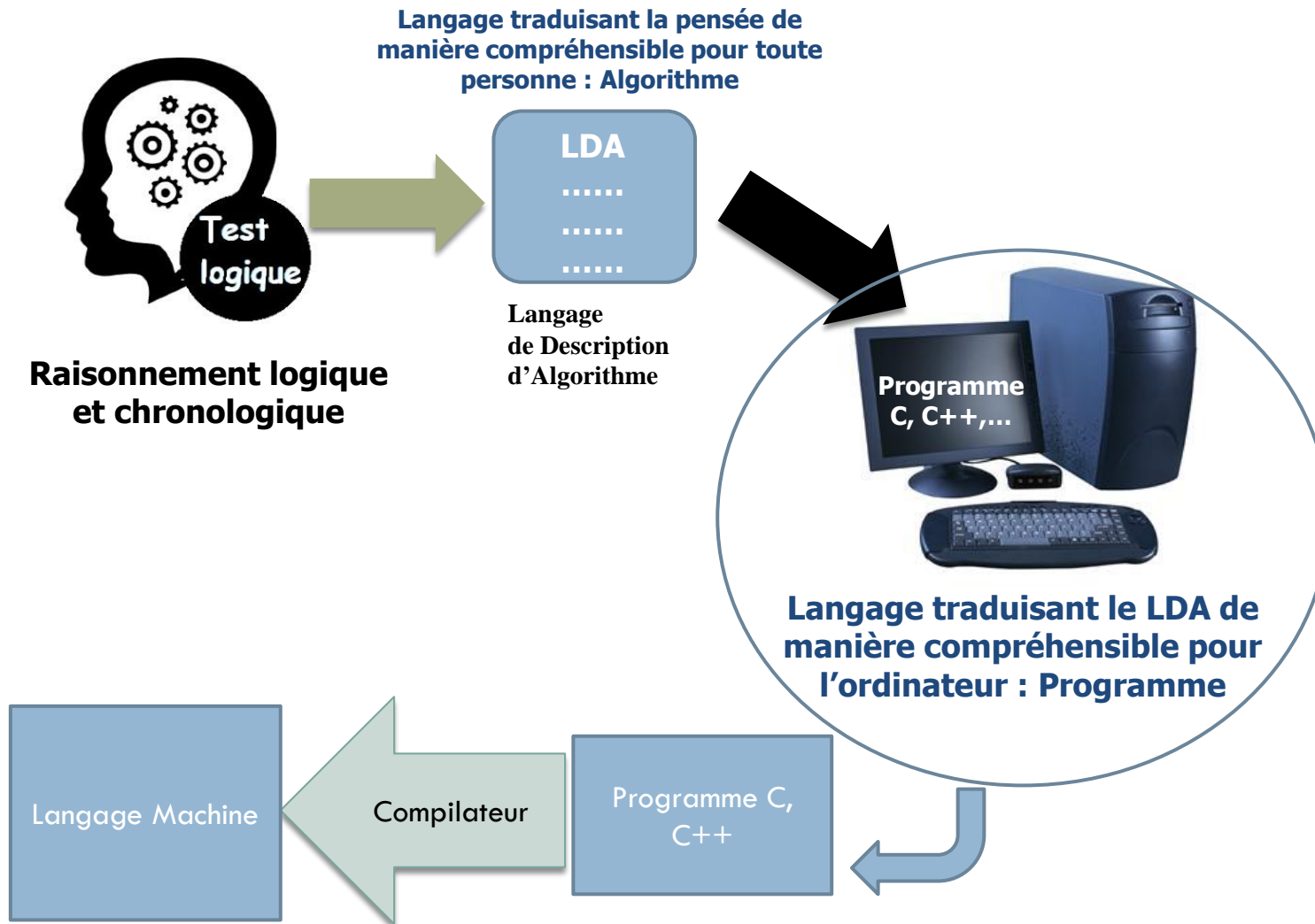
## I . 5. Les bases d'algorithmiques :

17

- ❑ Un algorithme est un ensemble de **règles logiques** et **chronologiques** qu'on doit suivre pour aboutir à la résolution d'un problème particulier.
- ❑ Ces règles sont constituées d'un nombre fini **d'opérations élémentaires (Arithmétique & Logique)**.
- ❑ Ces opérations seront **exécutées dans un ordre** bien déterminé.
- ❑ **Un algorithme** peut être assimilé à un raisonnement que l'on peut traduire avec un **langage** que toute personne peut comprendre :
  - ❑ **LDA : Langage de Description d'Algorithme**
- ❑ Le programme informatique est à la traduction du LDA à un autre langage compréhensible pour la machine (Pascal, Visual Basic, C, C++, C#, Java...)
- ❑ **Le LDA n'est pas un langage informatique.**

## I . 5. Les bases d'algorithmiques :

18



## I . 5. Les bases d'algorithmiques :

19

### Choix de langage :

Le choix de langage dépend souvent au domaine auquel appartient le problème à traiter comme le domaine du web, du mobile, du réseau ou encore de la sécurité informatique, les critères de choix se résume comme suit:

- ✓ La maniabilité et portabilité du langage de programmation;
- ✓ La popularité de langage (support disponible);
- ✓ La fiabilité;
- ✓ La syntaxe facile et intuitive,

**Le point commun entre tous les langages de programmation est la logique de la programmation elle-même ➔ ce qu'on appelle l'algorithme**

## I . 5. Les bases d'algorithmiques :

20

- ❑ L'algorithme est indépendant de tout langage de programmation;
- ❑ Il ne s'agit pas d'un code à soumettre à l'ordinateur pour l'exécuter;
- ❑ IL s'agit d'un esquisses ou d'un schéma détaillé des opérations qu'elles faut entamer pour mener à bien la solution du problème posé;
- ❑ Une fois l'algorithme est terminé et jugé correct et adéquat, on le traduit à un vrai programme à l'aide d'un langage de programmation pour qu'il soit exécuté par la machine.
- ❑ Un algo ne s'écrit pas forcément sur un ordinateur, on peut l'écrire sur un papier comme l'ébauche d'un future programme,

## I . 6. Types de notation :

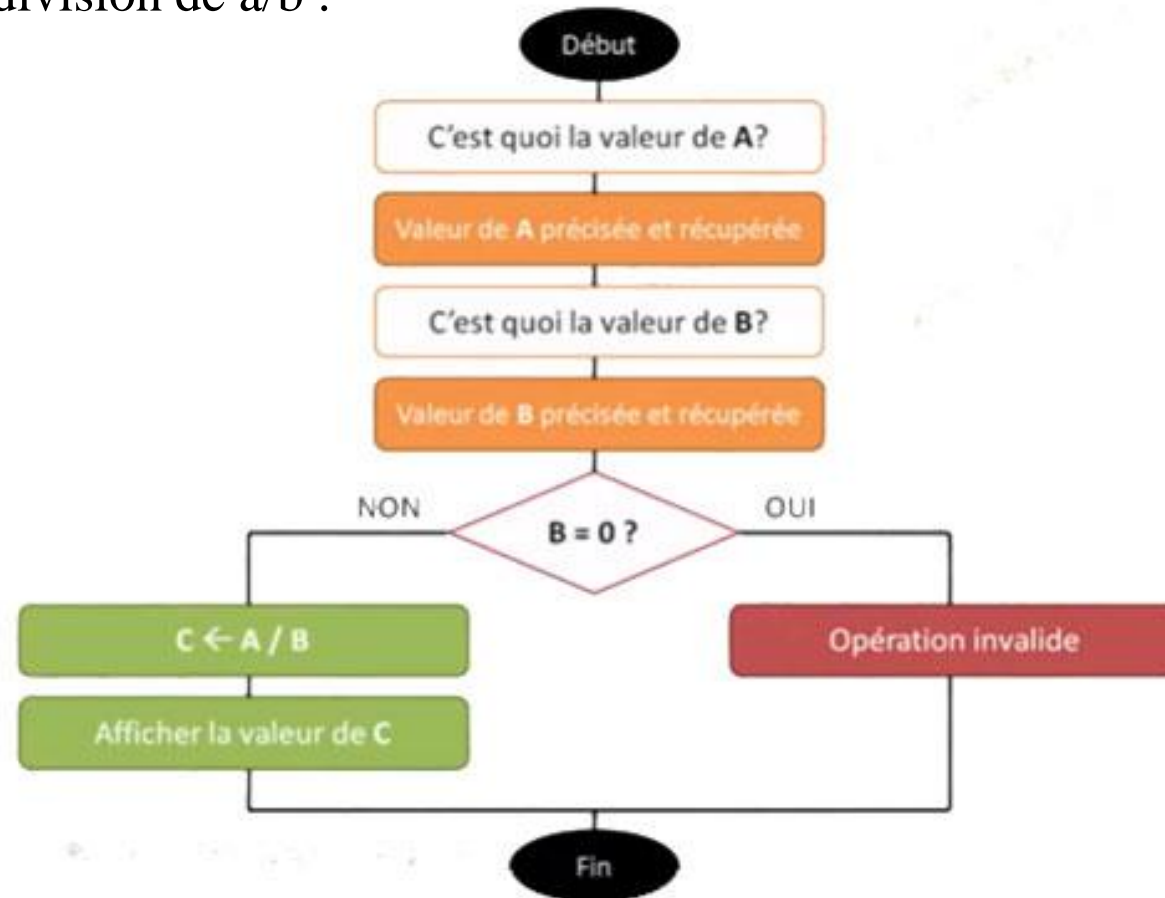
- ❑ Historiquement, plusieurs types de notations ont représenté des algorithmes :
- ❑ Il y a une représentation graphique, avec des carrés, des losanges, etc... qu'on appelle des **organigrammes**.
- ❑ On utilise généralement une série de conventions appelée «**pseudo-code**», qui ressemble à un langage de programmation authentique.
- ❑ Donc, chaque cuisinier peut faire sa sauce à sa guise, avec ses petites épices bien à lui, sans que cela prête à conséquence.

## I . 6. Types de notation :

22

### Organigramme :

- ❑ Algorithme de la division de  $a/b$  :



## I . 6. Types de notation :

23

### Pseudo-code:

- ❑ Algorithme de la division de  $a/b$  :

```
1  Algorithme Division
2
3  Variables
4      A,B : Entier
5      C   : Réel
6
7  Début
8      Ecrire('Entrer la valeur de A: ')
9      Lire(A);
10     Ecrire('Entrer la valeur de B: ')
11     Lire(B);
12     Si B=0 Alors
13         Ecrire('Opération invalide!')
14     Sinon
15         C <- A/B;
16         Ecrire('Le résultat est: ',C)
17     FinSi
18 Fin
```



Le pseudo-code est facilement traduisible en un langage de programmation

## I . 7. Structure de l'algorithme :

24

**Algorithme**

➔ *// nom\_algo*

**Variables**

➔ *// les différents variables : type*

**Début**

*// Instructions*

**Fin**



## I . 7. Structure de l'algorithme :

25

**Algorithme**    `nom_de_l_algo`



Le nom de l'algorithme

**Variables**

`variable1, variable2, ... : type1`  
`variable3, variable4, ... : type2`

La déclaration de variables

**Début**

`Liste des instructions`

Le corps de  
l'algorithme

**Fin**

## I . 7. Structure de l'algorithme :

26

**Exemple :** d'Algorithme qui permet de calculer le prix TTC d'un ensemble d'articles :

Algorithme **PrixTTC**      *// L'algorithme calculer le prix TTC*

Variables

tva, quantite : Entier  
prixHT, PrixTTC\_ : Réel

Début

**Les instructions**      *// Voici un exemple de commentaire*

Fin

## I . 7. Structure de l'algorithme :

27

- ❑ Dans chaque bloc, un ordre fourni se traduit par un ensemble d'instructions et ces instructions manipulent des objets.
- ❑ Chaque objet est défini grâce à un ensemble de trois qualificatifs qui sont :
  - ❑ **L'identificateur** : c'est le nom de l'objet.
  - ❑ **Le type** : c'est l'ensemble dans lequel l'objet prend sa valeur.
  - ❑ **La valeur** : c'est une valeur précise dans l'ensemble défini par son type.

On note cela de la manière suivante :

objet  $\rightarrow$  ( identificateur, type, valeur)

**Exemple** : masse  $\rightarrow$  ( m, réel, 20 Kg)

Noter qu'un objet doit être déclaré avant d'être manipulé.

## I . 8. Généralités sur les langages de programmation :

28

- ❑ Chaque ordinateur n'est capable de comprendre que son **langage machine**.
- ❑ Quelle que soit la nature de l'information traitée par un ordinateur, elle l'est toujours sous la forme d'un ensemble de nombres écrits en **base 2**, par exemple **01001011**.
- ❑ Le terme bit signifie « **binary digit** », c'est-à-dire **0** ou **1** en numérotation binaire. Il s'agit de la plus petite unité d'information manipulable par une machine numérique.
- ❑ Il est possible de représenter physiquement cette information binaire par un signal électrique.
- ❑ La structure binaire des mémoires impose aux ordinateurs une logique binaire, et une arithmétique binaire, c'est-à-dire portant sur des nombres représentés en **base 2** d'où le nom de **langage binaire**.

## I . 8. Généralités sur les langages de programmation :

29

### Le langage binaire

- ❑ C'est un langage dans lequel l'information est exprimée et manipulée sous forme d'une suite de bits
- ❑ **Un bit (binary digit) = 0 ou 1** (deux états électriques)
- ❑ L'octet est une unité d'information composée de 8 bits. Il permet par exemple de stocker un caractère comme une lettre ou un chiffre.

### Le format, le codage d'une information

- ❑ Les chiffres décimaux sont au nombre de 10, soit 0,1,2, ...,9.
- ❑ Dans le langage binaires ces 10 chiffres doivent être représentés en utilisant **"0"** et **"1"**.
- ❑ On doit donc trouver le format qui permet de représenter ces 10 états avec deux caractères.

## I . 8. Généralités sur les langages de programmation :

30

Donc le format 4 permet de représenter 16 symboles différents, donc il nous faut 4 bit.

- ❑ Ou encore  $2^8=256$  représente le nombre de possibilités qui permettent de coder tous les caractères alphabétiques, numériques et symboles tels que .,\*,+,8,.....
- ❑ Chacun des 256 caractères trouve sa représentation binaire via le **code ASCII**

## I . 8. Généralités sur les langages de programmation :

31

### Par exemple :

- De 97 à 122 on a l'alphabet minuscule (a...z)
- De 48 à 57 on les différents chiffres (0...9)
- Etc ...

En informatique, le codage de l'information s'effectue principalement en trois étapes :

- ☐ Elle est exprimée par une suite de nombres (numérisation)
- ☐ Chaque nombre est converti sous forme binaire
- ☐ Chaque élément binaire est représenté par un état physique

## I . 8. Généralités sur les langages de programmation :

32

### Code ASCII :

- ❑ ASCII (American Standard Code for Information Interchange) est un code dans lequel chaque chiffre de 0 à 9, chaque lettre de l'alphabet minuscule, chaque lettre de l'alphabet majuscule, chaque symbole syntaxique est identifié par un code.
- ❑ Il existe d'autres codage, tel que l'unicode (65536 caractères), etc....

Table des codes ASCII

Décimale	Binaire	Valeur	Explication
033	00100001	!	exclamation mark
034	00100010	"	Double quote
035	00100011	#	Number sign / hash sign
036	00100100	\$	Dollar sign
037	00100101	%	Pourcent
038	00100110	&	Ampersand
039	00100111	'	Simple quote
040	00101000	(	Left parenthesis / Opening parenthesis
041	00101001	)	Right parenthesis / Closing parenthesis
042	00101010	*	Asterisk
043	00101011	+	Plus
044	00101100	,	Comma
045	00101101	-	Minus / Dash
046	00101110	.	Dot
047	00101111	/	Forward slash
048	00110000	0	
049	00110001	1	
050	00110010	2	
051	00110011	3	
052	00110100	4	





## I . 8. Généralités sur les langages de programmation :

34

### La conversion :

#### Conversion binaire à décimal

Si on effectue maintenant la conversion dans le sens inverse, c'est à dire passer de la base 2 à la base 10, il faut :

1. Multiplier chaque bit par la puissance de 2 correspondants au rang du bit
2. Ajouter les résultats

#### Exemple 1 :

$$\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 2^3 & 2^2 & 2^1 & 2^0 \\ \hline 8 & + & 4 & + & 0 & + & 0 & = & (12)_{10} \end{array}$$

#### Exemple 2 :

$$(101101)_2 = 1*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 32 + 0 + 8 + 4 + 0 + 1 = (45)_{10}$$

## I . 8. Généralités sur les langages de programmation :

35

### Exercice :

1. Convertir en binaire les nombres 397, 133, 740
2. Convertir en décimal les nombres 1111 1011, 01011, 1101 1101



## I . 8. Généralités sur les langages de programmation :

36

### La conversion :

#### Conversion binaire à hexadécimal :

Elle comporte quatre étapes :

1. On décompose le nombre binaire en des groupes de 4 bits en allant de la droite vers la gauche.
2. Si le dernier groupe est incomplet, on le complète par des zéros
3. Représenter chaque groupe en base 10
4. Établir la correspondance de chaque groupe dans l'étape (3) avec la base 16

#### Exemple :

$$\begin{aligned}(101100)_2 &= \quad 0010 & 1100 \\ &= 0*2^3+0*2^2+1*2^1+0*2^0 & 1*2^3+1*2^2+0*2^1+0*2^0 \\ &= \quad 2 & 12 \\ &= \quad 2 & C \\ &= (2C)_{16}\end{aligned}$$

## I . 8. Généralités sur les langages de programmation :

37

### La conversion :

#### Conversion hexadécimal à binaire :

1. On isole chaque symbole du nombre
2. On donne la représentation en binaire de chaque symbole
3. La représentation de chaque symbole est complétée sur quatre bits

On arrive enfin à la représentation binaire du nombre

#### Exemple :

$2C_{16}$

2 en base 16 est 2 en base 10, et C en base 16 est 12 en base 10

Donc :

$(2C)_{16} =$	2	12
	10	1100
	0010	1100
$(2C)_{16} =$	$(00101100)_2$	
$=$	$(101100)_2$	

## I . 8. Généralités sur les langages de programmation :

38

### Généralisation de la conversion :

La formule se généralise pour une base quelconque, on divise de la même façon le nombre par la base.

**Exemple :** 18 en base 4

$$(18)_{10} = (102)_4$$

- Dans Le système octale à base 8, les chiffres vont de 0 à 7.
- Dans le système hexadécimal à base 16, les chiffres vont de 0 à 9, puis de A à F.

**Exemple :**

$$(5D)_{16} = (93)_{10} = (1011101)_2$$

$$(73)_{10} = (111)_8 = (49)_{16}$$

## I . 8. Généralités sur les langages de programmation :

39

### Exercice :

1. Convertir en hexadécimal

a) 316710

b) 21910

c) 656010

2. Convertir en décimal

a) 3AE16

b) FFF16

c) 6AF16

3. Convertir en base 2

a) F0A16

b) C0116



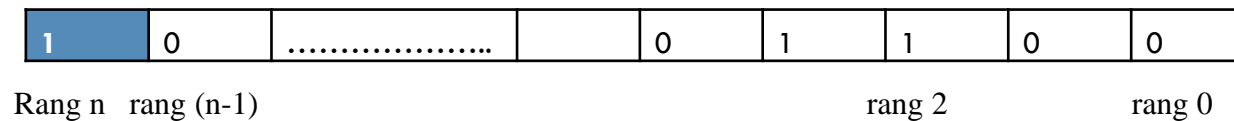
## I . 8. Généralités sur les langages de programmation :

40

### La représentation des entiers sur une mémoire :

- ❑ Il existe aussi une représentation qui tient compte du signe de l'entier lorsqu'on manipule des entiers relatifs (+5, -2,...).
- ❑ La case de rang n est réservée pour le signe (0 pour un entier  $>0$  et 1 pour un entier  $<0$ ) et sur les autres cases on représente la valeur absolue du nombre en binaire pur :

**Exemple :** Le nombre -12 s'écrit





## I . 8. Généralités sur les langages de programmation :

41

### La représentation des entiers sur une mémoire :

- ❑ Cette représentation tenant compte du signe n'est pas celle réellement utilisée car elle coute très chère.
- ❑ On lui préfère celle du complément à deux.
- ❑ Dans cette dernière représentation, si le nombre est positif, on le représente comme précédemment.
- ❑ Si le nombre est négatif, la représentation est obtenue en trois étapes :
  1. On code la valeur absolue du nombre
  2. On applique le non logique au résultat obtenu pour chaque bit
  3. On ajoute « 1 » au résultat obtenu en (2)

Ainsi pour représenter -12 les étapes sont les suivantes :

2) 

1	1	.....		1	0	0	1	1
---	---	-------	--	---	---	---	---	---

  
Rang n   rang (n-1)                      rang 2    ...    rang 0

1	1	.....		1	0	1	0	0
Rang n	rang (n-1)					rang 2	....	rang 0

## I . 8. Généralités sur les langages de programmation :

43

### Exemple 2 :

Addition en binaire :

$$1010 + 1010 = 10100$$

### Table d'addition

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 10 \text{ (on pose 0 et on retient 1)}$$

$$1+1+1 = 11 \text{ (on pose 1 et on retient 1)}$$

### Exemple 3 :

Multiplication en binaire

$$1010 * 1010 = 1100100$$

### Table de soustraction

$$0 - 0 = 0$$

$$0 - 1 = 1 \text{ et on retient 1}$$

$$1 - 1 = 0$$



Pour la multiplication, elle se fait exactement comme en décimale.

## I . 8. Généralités sur les langages de programmation :

44

- ❑ Le langage binaire n'est pas très pratique à cause des erreurs qui peuvent arriver, on peut dire que c'est un langage de première génération.
- ❑ Selon les besoins, beaucoup d'autres langages ont vu le jour dont on peut citer, l'assembleur, le basic, le fortran, le java, le turbo pascal,.....etc.
- ❑ La plus part de ces langages appelés langages évolués sont universels, c'est à dire qu'on peut lire les programme d'une machine par une autre.
- ❑ Ils sont proches du langage parlé et permettent une écriture simple des programmes.

## I . 8. Généralités sur les langages de programmation :

45

- ❑ Sur chaque machine est installé un programme écrit en langage machine (**compilateur**) qui permet la traduction du programme écrit en **langage évolué** en **langage machine** et c'est **le programme** écrit en langage machine qui est exécuté par la machine et donc il suffit à chaque fois d'avoir un bon compilateur pour travailler avec un langage de programmation.

## Chapitre

# 2 Les éléments de base d'un algorithme et leur représentation en langage C

## II . Les éléments de base d'un algorithme et leur représentation en langage C :

47

### Les commentaires :

- ❑ Un algorithme doit être aussi lisible que possible, d'où l'intérêt des commentaires qui sont de deux type :
- ❑ Les commentaires qui n'apparaissent pas à l'étape d'exécution:
  - ❑ Pour ces commentaires, on utilise le signe `//` s'il s'agit d'un commentaire sur une seule ligne.
  - ❑ Pour le cas d'un commentaire qui porte sur plusieurs lignes, on le commence par le signe `/*` et on le ferme par le signe `*/`.

## II .1. Les éléments de base d'un algorithme :

48

### Les commentaires :

#### ❑ Exemples :

// Ceci est un commentaire sur une seule ligne

/\* Ceci est un autre

Commentaire

Sur plusieurs lignes \*/

#### ❑ Les commentaires qui apparaissent à l'étape d'exécution:

- ❑ On les met entre deux guillemets, on les verra après les ordres d'entrée sortie.



## II .1. Les éléments de base d'un algorithme :

49

### Types de bases :

□ L'algorithme se content de définir les 4 types de bases qui sont:

- Les entiers ;
- Les réels ;
- Les booléens ;
- Les chaines de caractères.



- Chaque variable et constante doit être déclarée avant d'être utilisé : **nom + type**.
- Cette déclaration consiste en une réservation en mémoire de la place nécessaire et suffisante à l'hébergement de la valeur.

## II .1. Les éléments de base d'un algorithme :

50

### Types de bases :

#### ❑ Les entiers :

- ❑ Sont des nombres qui s'expriment sans virgule et peuvent être positif ou négatif ;
- ❑ Les entiers s'expriment sur 16 bits (entier court) ou 32 bits (entier long).

#### ❑ Les réels :

- ❑ Sont des nombres qui s'expriment avec une virgule (dite virgule flottante) ;
- ❑ Les réels s'expriment sur 32 bits (float) ou 64 bits (double précision) ou 80 bits (long double).

## II .1. Les éléments de base d'un algorithme :

51

### Types de bases :

#### ❑ Les booléens :

- ❑ Représentent un type qui n'accepte que l'une des deux valeurs « vrai » ou « faux »  
(représenté aussi par 0 ou 1) ;
- ❑ Il s'agit d'un type très léger de stockage dans la mémoire.

#### ❑ Les caractères :

- ❑ Représentent un type qui accepte des caractères alphabétiques, numériques ou symboles ;
- ❑ Il s'agit d'un texte (nom, code, commentaire).

## II .1. Les éléments de base d'un algorithme :

52

### Types de bases :

#### □ Exemples:

a : entier

a, b :entiers courts

a : réel

a, b :réels longs

a, b, c : caractère

## II .1. Les éléments de base d'un algorithme :

53

### Types de bases :

#### □ Résumé :

Type	Taille	Signé	Non signé
char	1 octet	-128 à 127	0 à $2^8-1$
short int	2 octets	$-2^{15}$ à $2^{15}-1$	0 à $2^{16}-1$
int	4 octets	$-2^{31}$ à $2^{31}-1$	0 à $2^{32}-1$
long int	4/8 octets (processeur 32/64 bits)	$-2^{63}$ à $2^{63}-1$	0 à $2^{64}-1$
int *	4/8 octets (processeur 32/64 bits)	$-2^{63}$ à $2^{63}-1$	0 à $2^{64}-1$
float	4 octets	$-2^{31}$ à $2^{31}-1$	0 à $2^{32}-1$
double	8 octets	$-2^{63}$ à $2^{63}-1$	0 à $2^{64}-1$
long double	12/16 octets (processeur 32/64 bits)	$-2^{95}$ à $2^{95}-1$ $-2^{127}$ à $2^{127}-1$	0 à $2^{96}-1$ 0 à $2^{128}-1$

## II .1. Les éléments de base d'un algorithme :

54

### Constantes, variables :

- ❑ Dans un programme, on a toujours besoin de manipuler des données, comme des nombres ou des textes. Ces données sont enregistrées dans des variables. Avant l'utilisation d'une variable au sein d'un programme (ou dans un algorithme) on doit définir son type qui peut être entier, réel, booléen ou chaîne de caractères.
- ❑ Une variable est une entité dont la valeur peut changer, d'où le nom variable, à l'inverse d'une constante qui est une entité dont la valeur reste inchangée durant tout le temps d'exécution du programme

## II .1. Les éléments de base d'un algorithme :

55

### Constantes, variables :

#### ❑ Exemple 1 :

Const entier  $N = 10$

C'est à dire qu'on déclare que  $N$  est une constante entière qui a pour valeur 10.

#### ❑ Exemple 2 :

Var  $m$ : réel

Si notre variable est la masse alors pour l'identificateur on peut choisir  $m$ , et pour le type c'est un réel.

## II .1. Les éléments de base d'un algorithme :

56

### Les variables: règles de nommage :

- ❑ Le nom de la variable (appelé aussi identifiant):
- ❑ Doit contenir seulement des caractères alphabétiques, numériques et le symbole souligné,
- ❑ Il doit commencer par un caractère alphabétique.

#### ❑ Exemples:

- ❑ Var123;
- ❑ Ma\_Variable (souligné),
- ❑ Prenom (sans accent);
- ❑ Numer\_de\_telephone (souligné et sans accent);
- ❑ NumDeTelephone (notation chameau).



## II .1. Les éléments de base d'un algorithme :

57

### Expressions :

Une expression peut être une instruction ou une suite d'instructions séparées par l'un des opérateurs suivants :

#### □ Les opérateurs d'assignation (L'affectation)

L'affectation consiste à attribuer une valeur à une variable

(ça consiste en fait à remplir ou à modifier le contenu d'une zone mémoire)

En pseudo-code, l'affectation se note avec le signe  $\leftarrow$  ou  $:=$

$\text{Var} \leftarrow e$  : attribue la valeur de  $e$  à la variable  $\text{Var}$

- $e$  peut être une valeur, une autre variable ou une expression
- $\text{Var}$  et  $e$  doivent être de même type ou de types compatibles
- l'affectation ne modifie que ce qui est à gauche de la flèche

## II .1. Les éléments de base d'un algorithme :

58

### Expressions :

□ **L'affectation** : (avec  $i, j, k$  : entier;  $x$  : réel;  $ok$  : booléen;  $ch1, ch2$  : chaîne de caractères)

### Exemples valides:

$i \leftarrow 1$      $j \leftarrow i$      $k \leftarrow i + j$      $x \leftarrow 10.3$      $OK \leftarrow \text{FAUX}$      $ch1 \leftarrow \text{"SMA"}$

$ch2 \leftarrow ch1$      $x \leftarrow 4$      $x \leftarrow j$

### Exemples non valides:

non valides:  $i \leftarrow 10.3$      $OK \leftarrow \text{"SMA"}$      $j \leftarrow x$

## II .1. Les éléments de base d'un algorithme :

59



### Quelques remarques :

- ❑ Beaucoup de langages de programmation (C/C++, Java, ...) utilisent le signe égal  $=$  pour l'affectation  $\leftarrow$ .
  - ❑ Attention aux confusions:
    - ❑ l'affectation n'est pas commutative :
      - ❑  $A=B$  est différente de  $B=A$
    - ❑ l'affectation est différente d'une équation mathématique :
      - ❑  $A=A+1$  a un sens en langages de programmation
      - ❑  $A+1=2$  n'est pas possible en langages de programmation et n'est pas équivalente à  $A=1$
- ❑ Certains langages donnent des valeurs par défaut aux variables déclarées. Pour éviter tout problème il est préférable d'initialiser les variables déclarées.

## II .1. Les éléments de base d'un algorithme :

60

### Exercices simples sur l'affectation :

Donnez les valeurs des variables A, B et C après exécution des instructions suivantes ?

Variables A, B, C: Entier

Début

$A \leftarrow 3$

$B \leftarrow 7$

$A \leftarrow B$

$B \leftarrow A + 5$

$C \leftarrow A + B$

$C \leftarrow B - A$

Fin

## II .1. Les éléments de base d'un algorithme :

61

### Exercices :

❑ **Exercice 1:** Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

**Variables A, B :Entier**

**Début**

**A  $\leftarrow$  1**

**B  $\leftarrow$  A + 3**

**A  $\leftarrow$  3**

**Fin**

❑ **Exercice 2:** Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

**Variables A, B, C :Entier**

**Début**

**A  $\leftarrow$  5**

**B  $\leftarrow$  3**

**C  $\leftarrow$  A + B**

**A  $\leftarrow$  2**

**C  $\leftarrow$  B - A**

**Fin**

## II .1. Les éléments de base d'un algorithme :

62

### Exercices :

❑ **Exercice 3:** Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

**Variables A, B : Entier**

**Début**

**$A \leftarrow 5$**

**$B \leftarrow A + 4$**

**$A \leftarrow A + 1$**

**$B \leftarrow A - 4$**

**Fin**

❑ **Exercice 4:** Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

**Variables A, B, C : Entier**

**Début**

**$A \leftarrow 3$**

**$B \leftarrow 10$**

**$C \leftarrow A + B$**

**$B \leftarrow A + B$**

**$A \leftarrow C$**

**Fin**

## II .1. Les éléments de base d'un algorithme :

63

### Exercices :

❑ **Exercice 5:** Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

**Variables A, B :Entier**

**Début**

**A  $\leftarrow$  5**

**B  $\leftarrow$  2**

**A  $\leftarrow$  B**

**B  $\leftarrow$  A**

**Fin**

❑ **Exercice 6:** Écrire un algorithme permettant d'échanger les valeurs de deux variables A et B, et ce quel que soit leur contenu préalable.

## II .1. Les éléments de base d'un algorithme :

64

### Expressions :

#### □ Les opérateurs de calcul (ou arithmétiques) :

+ : Addition

- : Soustraction

\* : Multiplication

/ : Division

% Modulo (reste de la division)

^ Puissance (2^3 vaut 8)

**A ← 10**

**B ← 2**

**C ← A+B**

**C vaut 12**

**C ← A/B**

**C vaut 5**

**C ← A%B**

**C vaut 0**

**A ← 10**

**A ← A+1**

**A vaut 11**



Le fait d'assigner une nouvelle valeur à une variable écrase automatiquement son ancienne valeur.



## II .1. Les éléments de base d'un algorithme :

65

### Expressions :

#### □ les opérateurs de comparaison:

Strictement plus grand que...

>

Plus petit ou égal à...

<=

Plus grand ou égal à...

>=

Égal à...

=

Différent de...

<>

Strictement plus petit que...

<

**A ← 10**

**B ← 2**

**A = B (Faux)**

**A > B (Vrai)**

## II .1. Les éléments de base d'un algorithme :

66

### Expressions :

#### □ Les opérateurs logique :

- Une condition composée est une condition formée de plusieurs conditions simples reliées par des opérateurs logiques:

ET, OU, OU eXclusif (XOR) et NON

#### Exemples :

- x compris entre 2 et 6 :  $(x > 2) \text{ ET } (x < 6)$
- n divisible par 3 ou par 2 :  $(n \% 3 = 0) \text{ OU } (n \% 2 = 0)$
- deux valeurs et deux seulement sont identiques parmi a, b et c :

$(a=b) \text{ XOR } (a=c) \text{ XOR } (b=c)$



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

## II .1. Les éléments de base d'un algorithme :

67

### Tables de vérité :

C1	C2	C1 ET C2
VRAI	VRAI	<b>VRAI</b>
VRAI	FAUX	<b>FAUX</b>
FAUX	VRAI	<b>FAUX</b>
FAUX	FAUX	<b>FAUX</b>

C1	C2	C1 OU C2
VRAI	VRAI	<b>VRAI</b>
VRAI	FAUX	<b>VRAI</b>
FAUX	VRAI	<b>VRAI</b>
FAUX	FAUX	<b>FAUX</b>

C1	C2	C1 XOR C2
VRAI	VRAI	<b>FAUX</b>
VRAI	FAUX	<b>VRAI</b>
FAUX	VRAI	<b>VRAI</b>
FAUX	FAUX	<b>FAUX</b>

C1	NON C1
VRAI	<b>FAUX</b>
FAUX	<b>VRAI</b>

## II .1. Les éléments de base d'un algorithme :

68

### Les opérateurs :

❑ Que produit l'algorithme suivant ?

Variables A, B, C en Caractères

Début

A ← "423"

B ← "12"

C ← A + B

Fin

❑ Que produit l'algorithme suivant ?

Variables A, B, C en Caractères

Début

A ← "423"

B ← "12"

C ← A & B

Fin

## II .1. Les éléments de base d'un algorithme :

69

- ❑ Fondamentalement, les ordinateurs, quels qu'ils soient, ne comprennent que **quatre catégories** d'ordres (d'instructions). Ces quatre familles d'instructions sont :
  1. **l'affectation de variables;**
  2. **la lecture / écriture;**
  3. **les tests;**
  4. **les boucles,**
- ❑ Un algorithme informatique se ramène donc toujours à la **combinaison de ces quatre** petites briques de base.
- ❑ Il peut y en avoir quelques unes, quelques dizaines, et jusqu'à plusieurs centaines de milliers.

## II .1. Les éléments de base d'un algorithme :

70

### La lecture : Instruction d'Entrées (1/3) :

- ❑ L'instruction de **lecture** permet d'introduire une valeur à l'algorithme.
- ❑ Cette valeur sera enregistrée dans une variable.
- ❑ Donc une lecture concerne une variable, puisque la valeur introduite sera mise dans cette variable.
- ❑ La syntaxe de l'instruction de lecture est comme suit :
  - ❑ **Lire** (**id\_var**) où **<id\_var>** est un identificateur d'une variable.
- ❑ On peut lire plusieurs variables à la fois, en suivant la syntaxe suivante :
  - ❑ **Lire** (**id1, id2, ..., idn**) où **<id1>, ... <idn>** représentent des identificateurs de variables qui sont séparés par des virgules.

## II .1. Les éléments de base d'un algorithme :

71

### La lecture : Instruction d'Entrées (2/3) :

- ❑ Soit l'exemple suivant (Algorithmique )

`Lire(x)` // *Ça veut dire quoi cette instruction ?*

- ❑ Le sens de cette instruction est comme suit : Lors du déroulement de l'algorithme, cette instruction indique au processeur qu'il doit attendre une valeur qui sera saisie par l'utilisateur. Une fois validée, cette valeur sera Enregistrée dans la variable x (espace mémoire).

#### ❑ Remarques:

- ❑ Une instruction de lecture concerne uniquement une variable. Cette dernière va recevoir la valeur saisie par l'utilisateur.
- ❑ On peut jamais écrire ces instructions :

`Lire(5); Lire (x+y); Lire('a');` Pourquoi ?

## II .1. Les éléments de base d'un algorithme :

72

### La lecture : Instruction d'Entrées (3/3) :

#### □ Quant-est-ce que nous utiliserons l'instruction de lecture ?

Elaborer une solutions informatique (Algorithme) pour résoudre un problème passe par plusieurs étapes. Parmi celles-ci, il y a l'étape de modélisation (établir un modèle). En général, ce modèle est constitué d'un ensemble de variables et des formules mathématiques. Pour avoir des résultats, on doit tout d'abord introduire les valeurs pour les variables de base (Non calculées). Donc, nous utiliserons l'instruction de lecture pour introduire ces valeurs.

#### ■ Prenons comme exemple le calcul de la surface d'un cercle :

On sait que la surface  $S$  d'un cercle est calculée comme suit :  $S = \pi R^2$

La valeur de  $\pi$  est connue (3.14), mais celle de  $R$  (le rayon) n'est pas connue. Donc il faut donner la valeur de  $R$  pour pouvoir calculer la valeur de  $S$ .

On aura donc, dans l'algorithme correspondant, l'instruction Lire ( $R$ ).



## II .1. Les éléments de base d'un algorithme :

73

### L'écriture : Instruction de Sortie (1/3) :

- ❑ L'instruction de **l'écriture** permet à l'algorithme d'afficher une information (une valeur fixe, valeur d'une variable ou une valeur calculée à travers une expression).
- ❑ La syntaxe de l'instruction de l'écriture est :
  - ❑ **Écrire**(**<valeur\_fixe>**|**<id\_var>**|**<expr>**) où : **<valeur\_fixe>** : est une valeur fixe (exmple : 5, 12.5, 'a', 'Erreur', vrai, ...).
  - ❑ **<id\_var>** : Identificateur d'une variable
  - ❑ **<expr>** : Expression mathématique (exemple :  $x + y/2 - 10$ )
- ❑ On peut écrire plusieurs objets (valeur, variable, expression) à la fois, en suivant la syntaxe suivante :
  - ❑ **Ecrire**(**<obj1>**, **<obj2>**, ..., **<objn>**) où **<obj1>**, ... **<objn>** représentent soit des valeurs fixes, des variables ou des expressions.

## II .1. Les éléments de base d'un algorithme :

74

### L'écriture : Instruction de Sortie (2/3) :

□ Prenons l'exemple suivant :

- Écrire (" Test ");
  - Le sens de cette instruction est comment suit : Lors de l'exécution de cette instruction, le processeur affichera sur l'écran la chaîne de caractère Test.
- Écrire (a, b)
  - Dans ce cas, le processeur affichera les valeurs enregistrée dans les variables a et b respectivement.
- Écrire ('a = ', a, ' et b = ', b)
  - Si a contient la valeur 8 et b contient la valeur 10, donc aura à l'affichage : a = 8 et b = 10

## II .1. Les éléments de base d'un algorithme :

75

### L'écriture : Instruction de Sortie (3/3) :

#### □ Quant-est-ce que l'instruction de l'écriture sera utilisée ?

La résolution d'un problème donné vise à avoir des résultats (la solution du problème). Ces résultats sont en générale des valeurs calculées et stockées au niveau des variables. On doit indiquer, à travers l'algorithme qu'il faut montrer ces résultat. Ceci sera en utilisant l'instruction de l'écriture.

#### ■ Prenons l'exemple précédent de calcul de la surface d'un cercle :

On a vu que la surface  $S = 2 \pi R$ . La valeur recherchée sera enregistrée dans la variable S. Donc, pour l'afficher on utilise l'instruction suivante :

Écrire (S);

## II .1. Les éléments de base d'un algorithme :

76

### Lecture et écriture: résumé :

- ❑ Les fonctions `lire()` et `écrire()` permettent de récupérer une valeur venant de l'extérieur (`lecture`) ou de transmettre une valeur à l'extérieur (`écriture`).
- **Lecture** : saisie (entrée) par clavier;
- **Ecriture** : Affichage sur écran (Moniteur);

## II .1. Les éléments de base d'un algorithme :

77

### Lecture et écriture: exercices :

❑ **Exercice 1:** Quel résultat produit le programme suivant ?

```
Variables val, double: réels
Début
    Val ← 231
    Double ← Val * 2
    Ecrire (Val)
    Ecrire (Double)
Fin
```

❑ **Exercice 2:** Ecrire un programme qui demande un nombre à l'utilisateur, puis qui calcule et affiche le carré de ce nombre.

## II .1. Les éléments de base d'un algorithme :

78

### Lecture et écriture: exercices :

#### ❑ Exercice 3:

Ecrire un algorithme qui demande son prénom à l'utilisateur, et qui lui réponde par un charmant « Bonjour » suivi du prénom. On aura ainsi le dialogue suivant :

```
machine : Quel est votre prénom ?  
utilisateur : Ahmed  
machine : Bonjour, Ahmed! ».
```

## II .1. Les éléments de base d'un algorithme :

79

### Lecture et écriture: résumé :

- ❑ L'instruction de lecture permet de réaliser les opérations des entrées de l'algorithme. Elle concerne les variables d'entrée.
- ❑ L'instruction d'écriture permet de montrer les résultats d'un algorithme. On affiche souvent les variables de sorties.
- ❑ L'affectation c'est une instruction qui sert à modifier la valeur d'une variable. Cette valeur est soit fixe (initialisation) ou calculée à travers une expression. Cette expression permet d'avoir soit le résultat final ou des résultats intermédiaires.