



Université Ibn zouhr  
Ecole Supérieure de Technologie  
Agadir



## **DUT Techniques de Communication et de commercialisation**

### **Module 7: Informatique Appliquée 2**

### **Algorithmique et bases de la programmation**

**Pr. TATANE KHALID**

## **Algorithmique**

- ✓ 8 heures de cours ;
- ✓ 10 heures de travaux dirigés ;
- ✓ 04 heures de cours Pratiques.

## **Initiation à la programmation VBA sous Excel**

- ✓ 2 heures de cours ;
- ✓ 04 heures de cours Pratiques.

## **Introduction**

- ✓ Algorithme et programme
- ✓ Représentation d'un algorithme
- ✓ Processus de la programmation

## **Structure d'un algorithme**

- ✓ Variables et Constantes
- ✓ Types des variables

## **Instructions de base**

- ✓ L'instruction d'affectation
- ✓ Opérateurs et Expressions
- ✓ Priorité des opérateurs
- ✓ Les instructions de lecture et écriture

## **Structures conditionnelles**

- ✓ Structure simple
- ✓ Structure alternative
- ✓ Structures alternatives imbriquées
- ✓ Structures à choix multiples

## **Les structures répétitives**

- ✓ La structure Pour
- ✓ La structure Tantque
- ✓ La structure Répéter Jusqu'à

# Introduction



**Algorithme est un terme d'origine arabe (Abu Ja'far Mohammed Ben Musa Al-Khwarismi) ;**



**Algorithmique : discipline qui étudie les algorithmes et leurs applications en Informatique.**

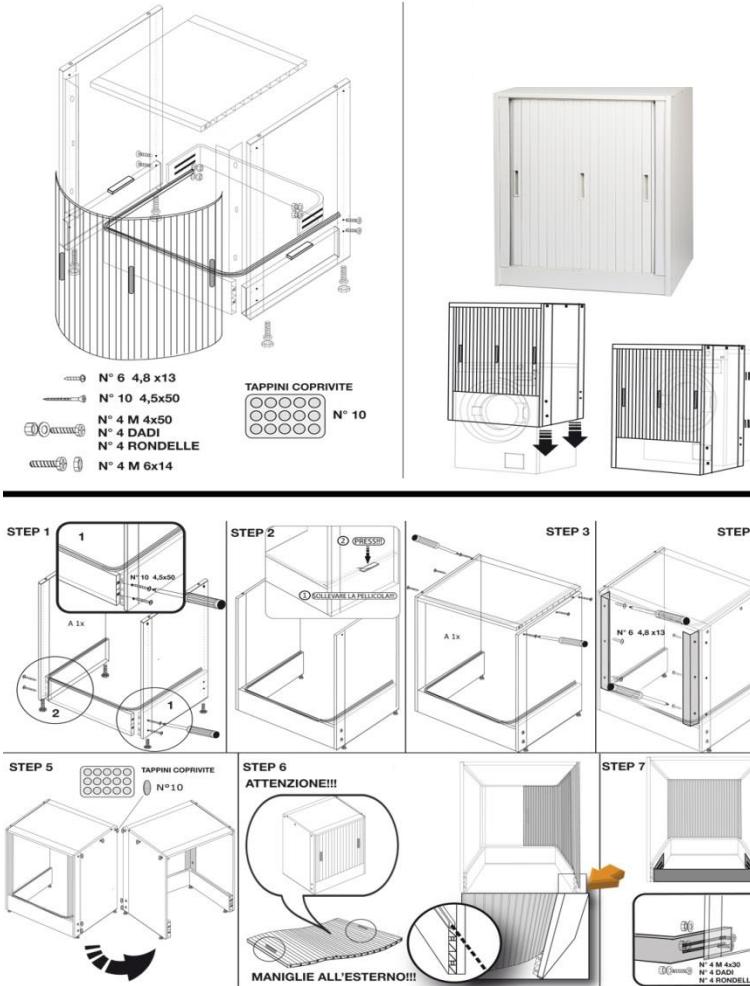
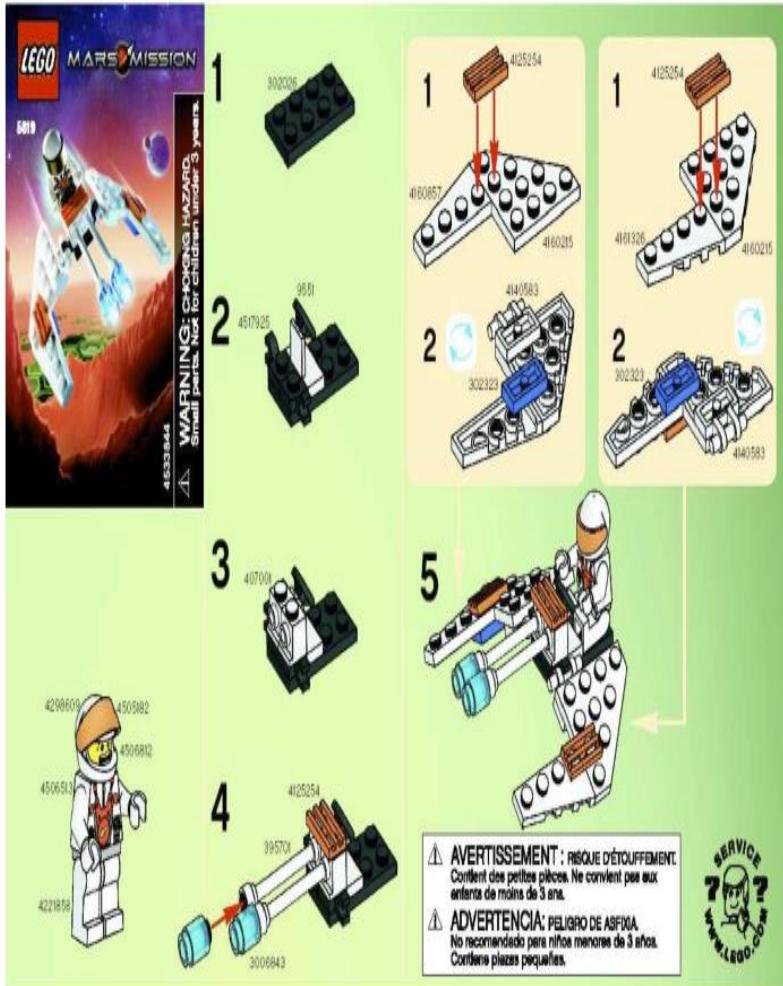


**Un algorithme, c'est une suite d'instructions, qui une fois exécutée correctement, conduit à un résultat donné;**

# Définition



**Un algorithme, c'est une suite ordonnée d'instructions (actions), qui indique la démarche à suivre pour résoudre un problème;**



## Comment cuire le riz ?

1. remplir une casserole d'eau ;
2. y ajouter une pincée de sel ;
3. la mettre sur le feu ;
4. attendre l'ébullition de l'eau ;
5. mettre le riz dans la casserole;
6. le laisser cuire 10 à 15 minutes ;
7. égoutter le riz.

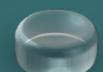
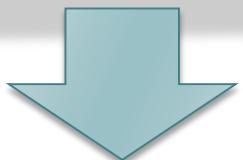
# Algorithme et programme



**Un algorithme n'est pas exécutable directement par aucune machine;**



**Mais c'est un travail de programmation à visée universelle qui ne dépend pas du langage de programmation;**



**Un algorithme exprime les instructions résolvant un problème donné indépendamment des particularités de tel ou tel langage.**

# Algorithme et programme



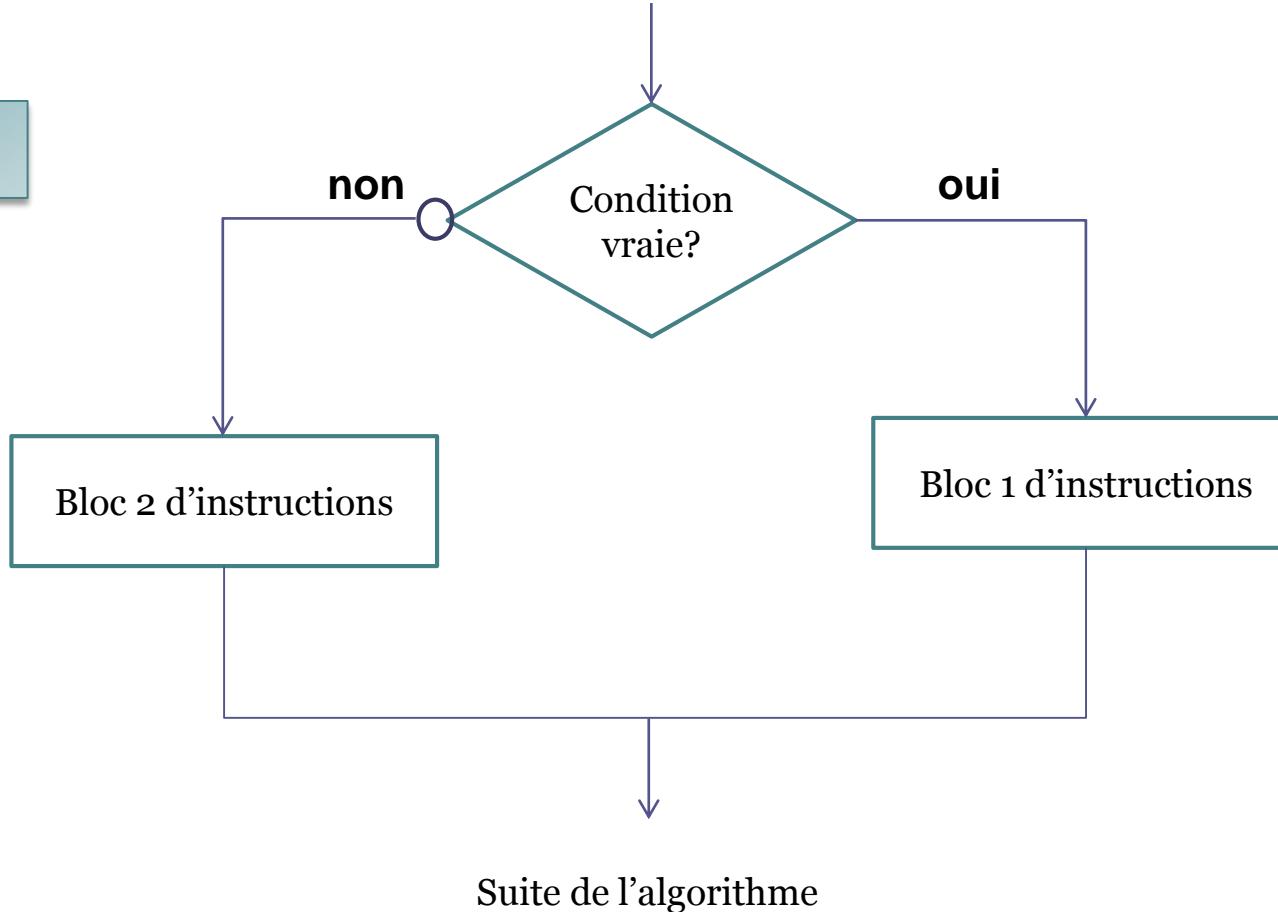
**L'élaboration d'un algorithme précède l'étape de programmation;**



**La rédaction d'un algorithme est un exercice de réflexion qui se fait sur papier.**

# Représentation d'un algorithme

## Organigrammes



Dès que l'algorithme commence à grossir, il devient illisible.

# Représentation d'un algorithme

## Pseudo-code

...

...

...

Si ( condition est vraie ) Alors  
    Bloc 1 d'instructions

Sinon

    Bloc 2 d'instructions

FinSi

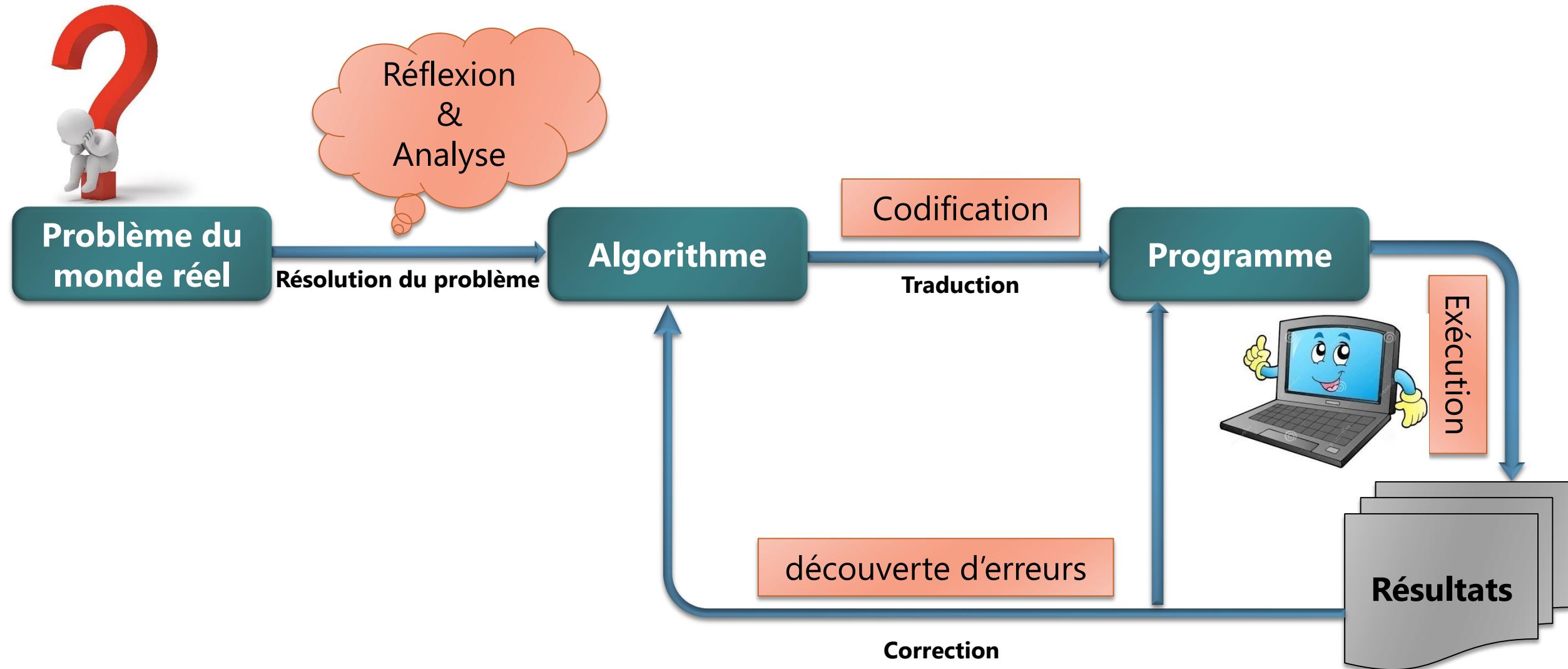
...

...

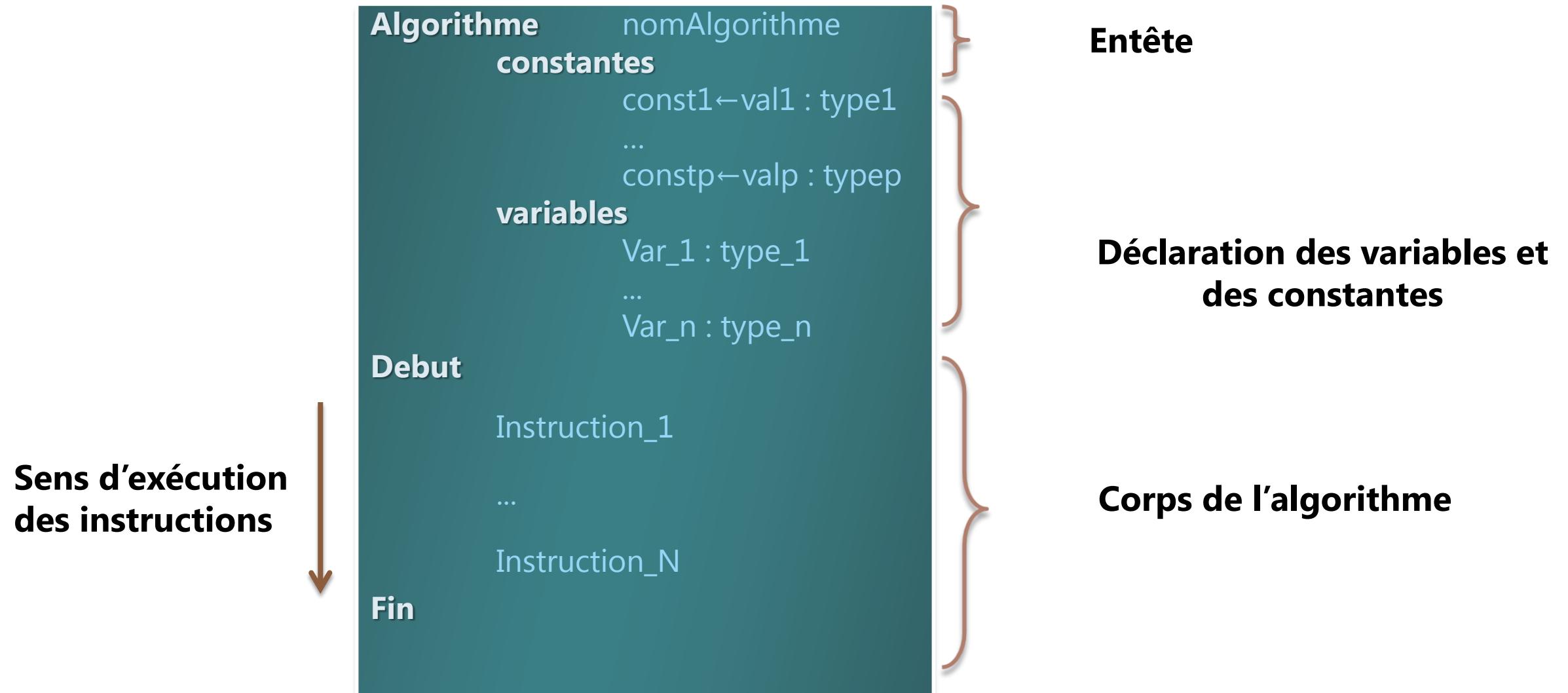
...

**Plus rapide à écrire et plus facile à traduire en un langage de programmation.**

# Processus de la programmation



# Structure d'un algorithme



# Variables et Constantes

La quasi-totalité des algorithmes utilisent des données élémentaires pour la résolution des problèmes. Ces données élémentaires sont sous forme de "variables" ou de "constantes".

# Variables et Constantes



**Une variable est une donnée dont la valeur peut changer au cours de l'exécution de l'algorithme . (d'où le nom de variable)**



**Une constante est une donnée fixe qui garde toujours la même valeur.**

Une variable/constante possède ces trois attributs:

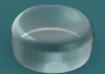
- ✓ **L'identificateur:** son nom;
- ✓ **La valeur:** son contenu;
- ✓ **Le type :** l'ensemble de ses valeurs.

- ▶ Un identifiant ne doit contenir que les caractères suivants: [a-z][A-Z][0-9]et { \_ };
- ▶ En plus un identifiant ne doit pas commencer par un chiffre.

# Types des variables



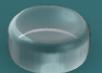
**Entier (-103, -80, -19, 0, 76, 2018 ...)**



**Réel (-345.45, -213, 0, 8.013 ...)**



**Chaine de caractères ('Hello', 'Ahmed', '2018\_@dut' ...)**



**Caractères ('X', 'y', '1', '?' ...)**



**Booléen: (VRAI ou FAUX)**

# Exemple de déclaration

**Attention!!!** toute variable/constante utilisée dans un algorithme doit faire l'objet d'une déclaration préalable.

## Constantes

PI←3.14 : réel

MAXI←32 : entier

## Variables

i, j, k : entier

x, y : réel

OK: booléen

Ch1, ch2 : chaîne de caractères

# L'instruction d'affectation

L'instruction d'affectation est l'opération qui consiste à attribuer une valeur à une variable. On la notera par le signe  $\leftarrow$ .

**VARIABLE  $\leftarrow$  valeur**

**Attention!!!** Si dans une instruction d'affectation, la variable à laquelle on affecte la valeur et la valeur affectée ont des types différents, cela provoquera une erreur.

On peut aussi attribuer à une variable la valeur d'une variable ou d'une expression de façon générale.

**VARIABLE  $\leftarrow$  EXPRESSION**

**Par exemple :**

**A  $\leftarrow$  5  
B  $\leftarrow$  A \* 6 + 2  
A  $\leftarrow$  A + 7**

# Affectation: Remarques

Lors d'une affectation, l'expression de droite est évaluée et la valeur trouvée est affectée à la variable de gauche.

Ainsi,  **$A \leftarrow B$  est différente de  $B \leftarrow A$**

l'affectation est différente d'une équation mathématique :

- **$A+1 \leftarrow 3$**  n'est pas possible en algorithmique et n'est pas équivalente à  **$A \leftarrow 2$** ;
- Les opérations  **$x \leftarrow x + 1$**  et  **$x \leftarrow x - 1$**  ont un sens en algorithmique et se nomment respectivement **incrémentation** et **décrémentation**.

# Exercices d'application

## Exercice 1:

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

### Algorithme Exercice1 Variables

A, B : Entier

Début

A ← 14

B ← A + 3

A ← 56

Fin

## **Exercice 2:**

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

### **Algorithme Exercice2**

#### **Variables**

**A, B, C : Entier**

**Début**

**A ← 5**

**B ← 3**

**C ← A + B**

**A ← 2**

**C ← B – A**

**Fin**

## Exercice 3:

Est-ce que cet algorithme permet d'échanger les valeurs de A et B (càd: mettre le contenu de A dans B et celui de B dans A) ?

### **Algorithme Exercice3**

#### **Variables**

**A, B : Entier**

**Début**

**A ← 11**

**B ← 6**

**A ← B**

**B ← A**

**Fin**

# Opérateurs et Expressions

**Un opérateur** est un signe qui relie deux variables/valeurs pour produire un résultat.

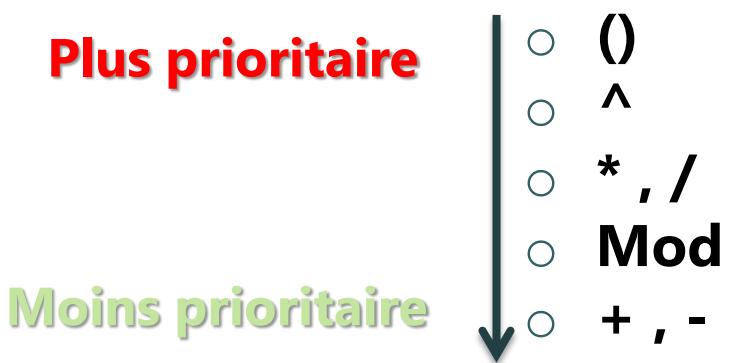
Les opérateurs dépendent du type de l'opération, ils peuvent être :

- **Opérateurs arithmétiques:** +, -, \*, /, Div (division entière),  
Mod (reste de la division entière), ^(Puissance);
- **Opérateurs de comparaison:** >, <, >= , <=, =, <> (différent);
- **Opérateurs logique:** and (et logique), Or (ou logique);
- **Opérateurs sur les chaînes:** & (concaténation).

**Une expression** est un ensemble de variables/valeurs reliées par des opérateurs et dont la valeur du résultat de cette combinaison est unique.

# Priorité des opérateurs

Pour les opérateurs arithmétiques donnés ci-dessus, l'ordre de priorité est le suivant:



Par exemple,  $10 * 6 + 2$  donne comme résultat 62.

En cas de besoin, on utilise les parenthèses pour indiquer les opérations à effectuer en priorité. Par exemple,  $10 * (6 + 2)$  donne comme résultat 80.

# Les instructions de lecture et écriture

Considérons l'algorithme suivant :

## Algorithme Calcul\_Carre

### VARIABLES

A : ENTIER

Début

A  $\leftarrow$  11 \* 11

Fin

Le problème de cet algorithme, c'est que, si l'on veut calculer le carré d'un autre nombre que 11, il faut réécrire à chaque fois l'algorithme;

# Les instructions de lecture et écriture

Les instructions de **lecture et d'écriture** permettent à la machine de communiquer avec l'utilisateur. On les appelle aussi les **entrées/sorties standards**.

**La lecture** (instruction d'entrée) permet d'entrer des données à partir du clavier.

**Syntaxe :**

**Lire (NomVariable)**

Lorsque le programme rencontre une instruction **Lire**, l'exécution du programme s'interrompt, attendant la saisie d'une valeur au clavier.

**Dès que l'on frappe sur la touche ENTER, l'exécution reprend.**

**Exemples :**

**lire(x)**

**lire(angle)**

**lire(age)**

# Les instructions de lecture et écriture

**L'écriture** (instruction de sortie) permet d'afficher des résultats ou la valeur d'une variable à l'écran.

**Syntaxe :**

**Ecrire (NomVariable)**

**ou de façon générale:**

**Ecrire (Expression)**

## Exemples :

Ecrire(x)

/\* Cette instruction affiche la valeur de la variable x\*/

Ecrire("Hello")

/\* Cette instruction affiche le message « Hello » \*/

Ecrire("Le carré de 11est:", 11\* 11)

/\* Cette instruction affiche « Le carré de 11est:121 » \*/

Avant de lire une variable, il est fortement conseillé d'écrire des libellés à l'écran, afin de prévenir l'utilisateur de ce qu'il doit frapper.

# Exercices d'application

## Exercice 1:

Écrire un algorithme qui affiche "Bonjour à tous les étudiants de l'ESTA".

## Exercice 2:

Écrire un algorithme qui demande à l'utilisateur de saisir un nom et un prénom, puis affiche le nom complet.

## Exercice 3:

Écrire un algorithme qui demande à l'utilisateur de saisir deux nombres entiers, ensuite calcule et affiche leur somme.

## Exercice 4:

Écrire un algorithme qui demande à l'utilisateur de saisir un nombre entier, ensuite calcule et affiche le carré de ce nombre.

# Introduction

## □ Structure séquentielle ou linéaire

- Dans une structure séquentielle toutes les instructions sont exécutées une seule fois les unes après les autres, dans l'ordre où elles ont été écrites.

## □ Structure conditionnelle

- Contrairement à une structure séquentielle, Une structure conditionnelle permet de faire exécuter une instruction ou une séquence d'instructions que si une condition est vérifiée.

## □ Structure itératives ou répétitive

- Permet de répéter l'exécution d'un groupe d'instructions un certain nombre de fois.

**Algorithme nomAlgo**

**Debut**

```

graph TD
    Debut[Debut] --> Instruction1[Instruction 1]
    Instruction1 --- Ellipsis[...]
    Ellipsis --- InstructionN[Instruction n]
    InstructionN --> Fin[Fin]
  
```

**Fin**

**Exemple :**

- Si feu vert alors je passe
- Si le feu est rouge alors je m'arrête

**Exemple :**

- Afficher les nombres de 1 à 100.

# La structure simple

La structure simple permet d'exécuter un groupe d'instructions en fonction de réponses à des conditions.

## Syntaxe :

```
SI (condition) ALORS  
    bloc d'instructions  
FIN SI
```

} Si **condition** est vraie, on exécute le **bloc d'instructions**,

**Application:** Écrire un algorithme qui permet de calculer le résultat de la division de deux entiers A et B .

# La structure alternative

La structure alternative

Une structure alternative est une situation dans laquelle on ne peut choisir que deux solutions possibles.

Syntaxe :

```
SI (condition) ALORS  
    bloc 1 d'instructions  
SINON  
    bloc 2 d'instructions  
FIN SI
```

Si **condition** est vraie, on exécute le **bloc1 d'instructions**,  
Si la **condition** est fausse, on exécute le **bloc2 d'instructions**.

**Application:** Écrire un algorithme qui lit un entier non nul puis affiche s'il est positif ou négatif.

# Les structures Alternatives Imbriquées

Les structures alternatives imbriquées sont des structures utilisées lorsqu'on a plus de deux cas possibles.

**Format général:**

```
SI (condition1) ALORS
    SI (condition2) ALORS
        SI (condition3) ALORS
            ...
            SINON
            ...
            FIN SI
        FIN SI
    SINON
    SI (condition_n) ALORS
        ...
        FIN SI
    FIN SI
```

**Application:** Écrire un algorithme qui lit trois entiers A, B et C, et affiche le plus grand.

# Structures à choix multiples

Lorsqu'on doit comparer une **même** variable avec plusieurs valeurs, comme par exemple:

```
SI (a=1) ALORS
    instruction1
SINON SI (a=2) ALORS
    instruction2
SINON SI (a=4) ALORS
    instruction4
SINON ...
    FINSI
    FINSI
FINSI
```

On peut remplacer cette suite de **si** par l'instruction **cas**.

**Syntaxe :**

```
Cas (variable) Vaut:
    valeur_1 : instruction1
    valeur_2 : instruction2
    ...
    valeur_n : instruction n
Sinon      : instruction n+1
FinCas
```

# Exercice d'application

Les mois de l'année sont codés de 1 à 12.

## **Exemple:**

- ✓ 1: Janvier;
- ✓ 2:Février;
- ✓ ...
- ✓ 12:Décembre.

## **Application:**

Écrire un algorithme qui affiche le mois correspondant à un code entré par l'utilisateur.