Département d'Informatique
Filière: SMIA
A.U: 2022-2023

جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

كلية العلوم
FACULTÉ DES SCIENCES

Module INFO II : Algorithmique I

Pr. Badraddine AGHOUTANE

1

2

Introduction à l'informatique

Qu'est ce que l'informatique ?

Mot inventé par **Philipe Dreyfus** en **1962**

INFORMATIQUE ?

Computer Science *ou* ***Informatics*** *en* ***Anglais***

INFORMATION
Science de l'information

AUTOMATIQUE
Art d'exécuter automatiquement des actions

Traitement automatique de l'information

Automatique : à l'aide d'une machine → **ORDINATEUR**

➔ **L'informatique** est la science de **traitement automatique de l'information** à l'aide d'un **ordinateur.**

3

Introduction à l'informatique

Que signifie le terme «information» ?

L'information est l'**élément de connaissance** susceptible d'être **codé** pour être **stocké** (mémorisé) et **traité**.

Différents types d'informations : numérique, texte, image, sons, vidéo…

Codage de l'information:

L'ordinateur **traite** des informations **codées** en **bits.**

1. Le **bit** correspond à un état : **0** ou **1**
2. L'association de **8 bits (octet)** , permet de coder **256 informations (code ASCII étendu)**

Par exemple: **01000001** → Codification de la lettre **A**

Unités de mesure de la quantité de mémoire:

- 8 bits → 1 octet
- 1024 octets →**1 Ko** (Kilo octets).
- 1024 Ko →**1 Mo** (Mega octets)
- 1024 Mo →**1 Go** (Géga octets).
- 1024 Go →**1 To** ((Téra octets)
- 1024 To → **1 Po** (Péta octets).
- 1024 Po →**1 Eo** ((Exa octets)
- 1024 Eo → **1 Zo** (Zetta octets).
- 1024 Zo →**Yo** (Yotta octets).

4

Introduction à l'informatique

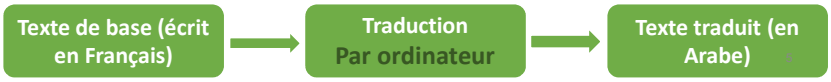
Que signifie le terme « traitement » ?

Un traitement est l'ensemble de toutes les opérations que l'on peut effectuer sur des informations (***saisie, mémorisation, calcul, modification, transmission***) afin de les rendre **utiles et utilisables.**

Un **traitement automatique** signifie qu'un **ordinateur** va exécuter les opérations, sans intervention humain :

Données à l'état brut

➢Exemple d'un **traitement**: **traduction d'un texte**

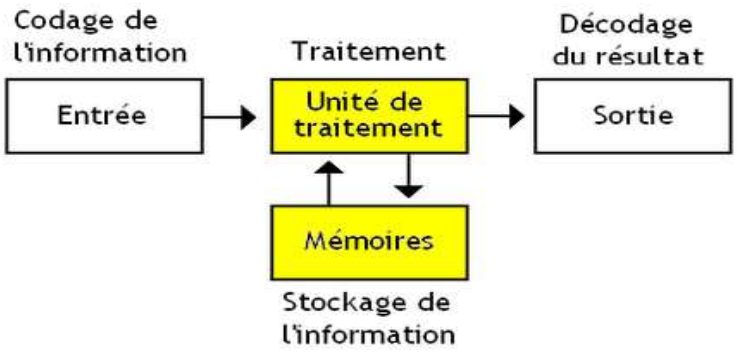


Introduction à l'informatique

Qu'est ce qu'un ordinateur?

Un **ordinateur** est une **machine** (*ensemble de circuits électronique*) qui permet le **traitement automatique de l'information** :

1. **Acquisition, stockage :** acquérir et conserver de l'information
2. **Traitement** : effectuer des **calculs** et exécuter des actions,
3. **Restitution** : restituer les ***résultats*** obtenus.



Introduction à l'informatique

Qu'est ce qu'un ordinateur?

Types d'ordinateurs

Toute machine capable de **manipuler des informations binaires** peut être qualifiée **d'ordinateur**. Le type d'ordinateur le plus présent sur le marché est le PC (Ordinateur personnel), toutefois il existe d'autres types d'ordinateurs :

- *PC Portable*
- *Stations (Alpha, SUN,…)*
- *Tablette*
- …

Un **ordinateur** se compose de **deux parties** essentielles :

1. **Matériel (Hardware):** ensemble d'éléments physiques utilisés pour le traitement de l'information.
2. **Logiciel (Software):** ensemble de programmes (algorithmes) servant à un traitement déterminée.

7

Introduction à la programmation

Généralités

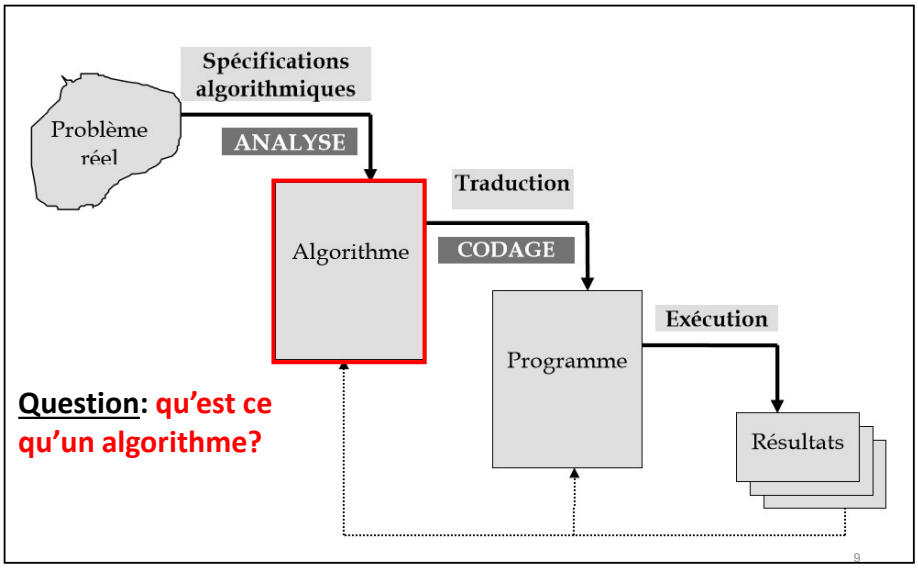
- En **informatique,** pour **résoudre un problème**, l'utilisateur doit écrire un **programme** qui sera exécuté par un **ordinateur**,
- Le **programme** est une succession logique et ordonnée d'actions,
- La **programmation** est l'ensemble des **étapes** et **tâches** qui permettent d'élaborer un **programme**.

Pour écrire un programme, il faut:

1. **Analyser** et bien **définir le problème** à traiter;
2. **Choisir la bonne solution** et savoir la **découper** logiquement en un ensemble **d'opérations élémentaires** (**actions**),
3. **Ecrire** un **algorithme** qui décrit toutes ces **actions** et leurs **séquencement** pour obtenir un **résultat correct**,
4. **Traduire l'algorithme** en un **programme** en utilisant un **langage de programmation compréhensible** par la machine (ex. *langage C*)

Introduction à la programmation

Les différentes étapes du processus de programmation:



Question: qu'est ce qu'un algorithme?

9

Algorithmique: Outils de base

Exemple d'un algorithme?

Ci-dessous, un exemple d'algorithme rédigé en langage courant:

Nom : **E1**

Choisir un nombre.
Lui ajouter 1.
Multiplier le résultat par 2.
Soustraire 3 au résultat.
Afficher le résultat.

Si on applique cet algorithme au nombre 3, on a :

$$3 \xrightarrow{+1} 4 \xrightarrow{\times 2} 8 \xrightarrow{-3} 5$$

On peut identifier cet algorithme à une fonction affine :

$$f(x) = 2(x+1) - 3 = 2x + 2 - 3 = 2x - 1$$

10

Algorithmique: Outils de base

Qu'est ce qu'un algorithme?

Le terme ***algorithme*** vient du nom du célèbre mathématicien arabo-musulman **Muḥammad ibn Mūsā *Al-Khawarizmi*** (780-850 après J.C.)

Un **algorithme** est une liste complète et détaillée d'actions et de leur séquencement, afin de résoudre un problème donné :

- **Intérêt: séparation** des étapes **d'analyse/de codage** (*ne pas se préoccuper de la syntaxe des langages de programmation*),
- **Qualités: exact** (*fournit le résultat souhaité*), **efficace** (*temps d'exécution, mémoire occupée*), **clair** (*compréhensible*), **général** (*traite le plus grand nombre de cas possibles*), …

L'algorithmique désigne la discipline qui étudie les algorithmes et leurs applications en Informatique.

11

Algorithmique: Outils de base

Qu'est ce qu'un algorithme?

Objectifs de l'algorithmique

L'**objectif de l'algorithmique** est de permettre *une bonne analyse du problème* afin de proposer une *meilleure solution*, celle-ci doit être correcte, précise et moins couteuse en utilisant un ensemble de **techniques de programmation.**

L'analyse d'un problème en informatique peut prendre **jusqu'à 90% du temps** alloué à la résolution du problème.

Cette **analyse** permet de :

1. **Transformer** le texte du **problème** en un ensemble **d'étapes élémentaires**
2. **Revoir les actions élémentaires** utilisés dans la solution pour essayer **d'optimiser,** si possible, ce schéma avant de procéder à sa **traduction** dans un langage de programmation

12

Algorithmique: Outils de base

Qu'est ce qu'un bon algorithme?

On peut noter qu'un **bon algorithme** est un schéma de résolution possédant les caractéristiques suivantes:

- ✓ **Correct**: répond au problème posé.
- ✓ **Précis**: fournit exactement les résultats attendus.
- ✓ **Rapide**: utilise un temps d'exécution minimal.
- ✓ **Efficace**: utilise le moins d'espace mémoire possible.
- ✓ **Clair et lisible**: ne présente pas de difficulté de compréhension pour un autre programmeur désirant le maintenir ou le développer.
- ✓ **Résistant**: est capable de détecter les cas de mauvaise utilisations.

13

Algorithmique: Outils de base

Qu'est ce qu'un bon algorithme?

Exemple de résistance d'un algorithme

Dans la **résolution d'une équation de premier degré:**

$$a.x+b=0$$

Pour laquelle:

$$x=-b/a$$

Un **algorithme de résolution** de cette équation doit réagir si on lui introduit une valeur **a=0** (Mauvaise utilisation).

- **Si (a=0),** il faut refuser de calculer la solution et afficher par conséquent un **Message d'erreur**,
- **Sinon,** la machine va se bloquer car elle ne sait pas faire des **divisions par zéro.**

14

Algorithmique: Outils de base

Définition d'un algorithme?

Algorithme:

Etape 1

Opération 1
Opération 2
....
Opération L

… Etape n

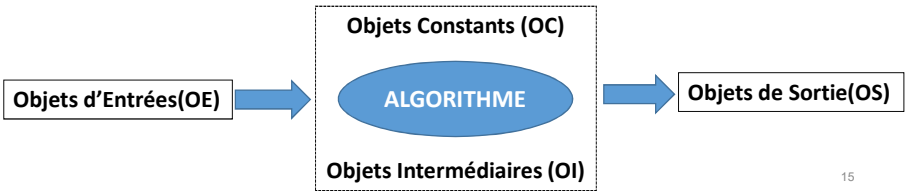
Opération 1
Opération 2
....
Opération N

Etape 2

Opération 1
Opération 2
....
Opération M

Un **algorithme** utilise un certain **nombre d'objets** pour fonctionner.

Cette ensemble s'appelle: **Environnement de l'algorithme**.



15

Algorithmique: Outils de base

Définition d'un algorithme?

Cet **environnement** est constitué d'un **ensemble d'objets :**

1. **Objets d'entrée (OE):** représentent l'ensemble des données que l'utilisateur doit introduire à l'algorithme,
2. **Objets de sortie (OS):** représentent l'ensemble des résultats produits par l'algorithme.
3. **Objets constants (OC):** représentent l'ensemble des objets dont la valeur ne change pas au cours de l'exécution de l'algorithme
4. **Objets intermédiaires (OI):** ensemble d'objets de traitements internes, qui ne sont ni d'entrée ni de sortie, mais peuvent être :
   - **des compteurs,**
   - **Ou des objets calculés** à partir des objets d'entrée et des objets constants et serviront pour produire des objets de sortie.

16

Algorithmique: Outils de base

Structure simplifiée d'un algorithme

Rappelons qu'un algorithme :

- est une **séquence** bien définie d'**étapes et d'opérations** (*calcul, manipulation de données,…*), permettant d'accomplir une **tâche**.
- utilise un certain nombre **d'objets** (objets d'entrées, constants, …) pour fonctionner appelé **environnement de l'algorithme.**

On distinguera **trois parties** dans un algorithme:

1. **Le titre,** tout algorithme porte un titre. Choisissez un titre qui permet de comprendre ce que fait l'algorithme.
2. **Une partie déclaration,** de tous les objets utilisés par l'algorithme (CONST, VAR)
3. **Une partie exécutable,** contenant toutes les actions permettant d'agir sur ces objets délimitée par les mots DEBUT et FIN

17

Algorithmique: Outils de base

Présentation d'un algorithme

Historiquement, deux façons pour représenter un algorithme:

1. **L'Organigramme:** représentation graphique avec des symboles (carrés, losanges, etc.)
   - ✓ offre une vue d'ensemble de l'algorithme
   - ✓ représentation quasiment abandonnée aujourd'hui
2. **Le Pseudo-code:** représentation textuelle avec une série de conventions ressemblant à un langage de programmation (sans les problèmes de syntaxe)
   - ✓ plus pratique pour écrire un algorithme
   - ✓ représentation largement utilisée

18

Algorithmique: Outils de base

Présentation d'un algorithme

Un algorithme pourra se présenter par un langage algorithmique (**pseudo code**) de la manière suivante:

```
Algorithme  nom_algorithme
        (Déclaration)
Const
        NOM_DE_CONST=valeur_de_const
Var
        nom_var: nom_type_var
Début
        (Corps de l'algorithme)
        (Actions)
Fin
```

19

Algorithmique: Outils de base

Exemple des variables et de constantes:

Const **ANNEE_COURS** = 2023
TAILLE_MIN = 1.65
PI = 3.14

Var **Nom**, **Prenom, Adresse** : chaine de caractère
Age : entier
Etat_civil : booléen

Remarque: Si plusieurs **objets variables** sont de **même type**, on peut les déclarer tous ensembles, séparés par des virgules.

Exemple d'algorithmes:

1. Algorithme qui calcul et affiche la **somme**, le **produit** et la **moyenne** de trois nombres entiers **a, b et c**.
2. Algorithme qui calcul et affiche **la surface et le périmètre** d'un **cercle de rayon R**.

20

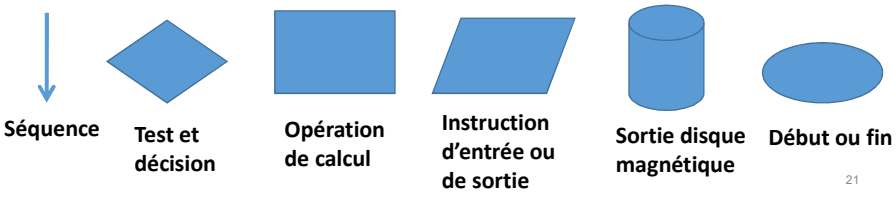
Algorithmique: Outils de base

Organigramme:

Un **organigramme** est une représentation schématique ou graphique d'un algorithme mettant en valeur sa structure.

Un **organigramme** permet de mieux présenter les différents modules de traitement et d'exploiter la succession des opérations d'un travail

Liste des symboles les plus utilisé dans un organigramme

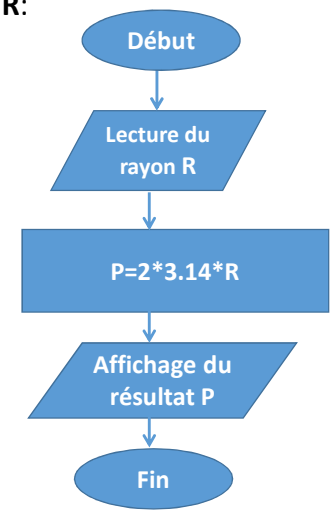


21

Algorithmique: Outils de base

Organigramme : Exemple

Traduction schématique de l'**algorithme de calcul du périmètre d'un cercle de rayon R**:



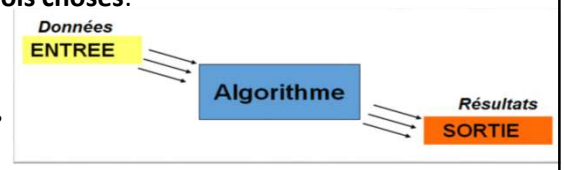
22

Les outils de base de l'algorithmique

Un **algorithme** doit expliquer à l'ordinateur quelles sont les **données** et quelles sont les **opérations** à appliquer **pour traiter ces données.**

Un **algorithme** informatique, **trois choses**:

1. ***Il lit des données en entrée***
2. ***Il effectue des calculs***
3. ***Il écrit des résultats en sortie***



Notion d'objet

Un **algorithme** est constitué :

- de **données** (**objets**) appelé **environnement** de l'algorithme,
- **d'actions** et **opérations** pour **traiter ces données** (**objets**).

Un **objet** peut se présenter sous **deux formes différentes**:

- **Objet variable :** s'il **peut être modifié** par les **actions** de l'algorithme.
- **Objet constant :** dans le cas contraire.

23

Les outils de base de l'algorithmique

Notion de variable :

Une **variable** permet de donner un **nom** à un emplacement **mémoire** ou une **valeur** est stockée, en vue de sa **manipulation** par **l'algorithme**.

Pour définir une **variable**, il faut spécifier :

- **Son nom (identificateur),**
- **Sa valeur,**
- **Son type** (*nombre entier, réel, caractère, suite de caractères, …*)

La **syntaxe** de déclaration d'une **variable** est la suivante :

```
Var     nom_variable : Type
```

Remarque: si plusieurs **variables** sont de **même type**, on peut les **déclarer** tous ensembles, séparés par des **virgules**

```
Var     nom_variable1, nom_variable2, nom_variable3  : Type
```

24

Les outils de base de l'algorithmique

1. Identificateur :

Une **variable** doit avoir un **nom (identificateur)** qui commence par une **lettre** et ne comporte ***ni espace ni caractères spéciaux*** :

- La **suite des caractères** pour **nommer des variables** est:
  - ✓ *lettres non accentuées (**a,b,…z, A,B,…Z**),*
  - ✓ *chiffres (**0,1,2,….9**),*
  - ✓ *caractère soulignement (_).*

Exemples d'identificateurs:

temps, calcul_vectoriel, mois1, a54b2 ➔ **Identificateurs CORRECTs**

1temps, calcul vectoriel, a54/b2 ➔ **identificateurs INCORRECTs**

Remarque : de **préférence** le nom d'une variable doit être **significatif** et choisi par le **concepteur** en fonction du **rôle** que la variable joue dans cet algorithme.

Par exemple : **Nbre_employes** est une bonne appellation pour identifier le **nombre d'employés** dans une entreprise *(au lieu d'utiliser **N** ou **NB** pour nommer cette variable).*

25

Les outils de base de l'algorithmique

Type d'objet

A chaque **variable** utilisée dans un programme, il faut lui associer un **type** qui permet de définir :

- l'**ensemble des valeurs** que peut prendre cette variable,
- l'**ensemble des opérations** qu'on peut appliquer sur cette variable

La syntaxe de déclaration d'une variable est la suivante :

Var **Nom_Variable 1, Nom_Variable 2, … : Type**

Les **principaux types** utilisés en algorithmique sont le type :

- **entier**
- **réel**
- **caractère**
- **chaîne de caractères**
- **logique ou booléen**

26

Les outils de base de l'algorithmique

Le type Entier

Une **variable** est de type **entier** si elle prend des valeurs dans l'ensemble **Z** (*entiers relatifs*). Ainsi, elle peut supporter les opérations suivantes :

Opération	Notation
Addition	+
Soustraction	-
Multiplication	*
Division entière	div
Modulo (reste de la division)	mod

D'autres **opérateurs possibles:**

Exposant : (^) **Comparaisons : (<, = , >, <= , >= , <>)**

Exemples :

17 div 5 = 3 **17 mod 5 = 2**

Remarque: l'ensemble des **valeurs possibles** d'une **variable de type entier varie** selon le **langage de programmation** utilisé (*i.e.* ***nombre de bit*** *réservé*)

Les outils de base de l'algorithmique

Le type réel ou décimal

Il existe **plusieurs types de réels** représentant chacun un ensemble particulier de valeurs prises dans **IR** (*ensemble des nombres réels*).

Ici encore, cette distinction se justifie par le **mode de stockage** des informations dans le **langage de programmation**.

Il existe **deux formes** de représentation des **valeurs réelles** :

- ✓ la **forme usuelle** avec le **point** comme symbole **décimal.**

  Exemples :

  -3.2467 2 12.7 +36.49

- ✓ la **notation scientifique** selon le format **aEb**, où :

  a est la **mantisse**, qui s'écrit sous une forme **usuelle**

  b est l'**exposant** représentant un **entier relatif**.

  Exemples :

  325 = 3.25E2 = 0.325E+3 = 3250E-1 = ...

28

Les outils de base de l'algorithmique

Les opérations définies sur les réels sont :

Opération	Notation
Addition	+
Soustraction	-
Multiplication	*
Division (réelle)	/

D'autres **opérateurs possibles:**

Exposant : (^) **Comparaisons** : (**<**, **=** , **>**, **<=** , **>=** , **<>**)

Exemples :

17 **/** 5 = 3.4 0.5 **^** 2 = 0.25

29

Les outils de base de l'algorithmique

Le type caractère

- Un **caractère** peut appartenir au domaine des **chiffres** de **"0"** à **"9"**, des **lettres** (***minuscules*** *et majuscules*) et des **caractères spéciaux** (***** , **/** , **{** , **$** , **#** , **%** , …).
- Un **caractère** sera toujours noté entre des **guillemets**.

Exemple: Le caractère espace (**blanc**) sera noté **" "**

Les **opérateurs** définis sur les données de **type caractère** sont :

Opération	Notation
Égal	=
Différent	#
Inférieur	<
Inférieur ou égal	<=
Supérieur	>
Supérieur ou égal	>=

Remarque: La **comparaison** entre les **caractères** se fait selon leurs **codes ASCII**.

Exemples:

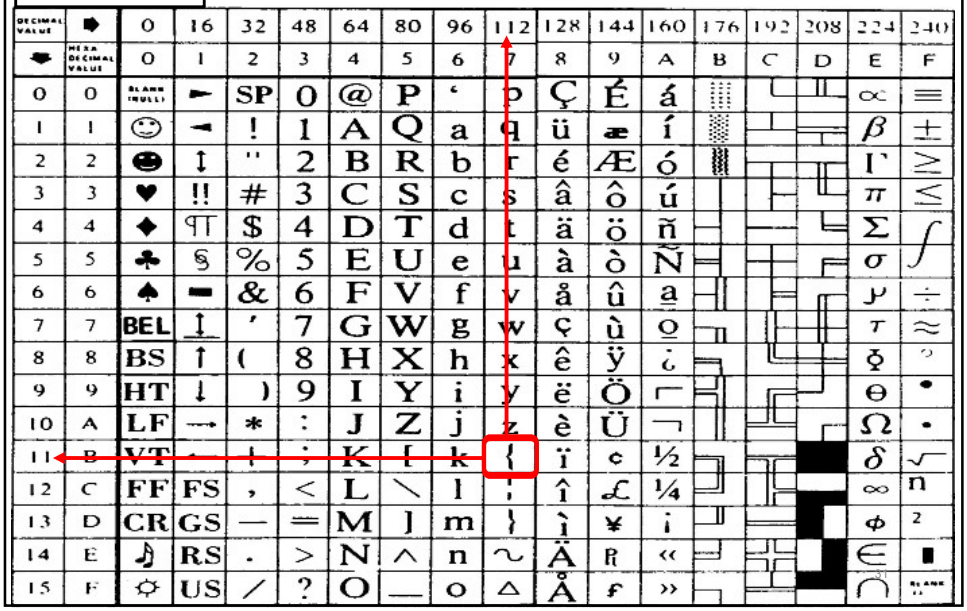
Caractère: " " < "0" < "1" < "A" < "B" < "a" < "b" < "{"

Code ASCII: **34 48 49 65 66 97 98 123**

30

Les outils de base de l'algorithmique

Code ASCII



Les outils de base de l'algorithmique

Le type logique ou booléen

Une **variable logique** (ou **booléen**) peut prendre l'une des deux états (valeurs) : **"Vrai" ou "Faux".**

Elle **intervient** dans l'**évaluation d'une condition (Vraie ou fausse).**

Les **principales opérations** définies sur les variables de type logique sont : **Négation (NON), Intersection (ET) et Union (OU).**

L'application de ces opérateurs se fait conformément à la **table de vérité** suivante :

A	B	NON (A)	A ET B	A OU B
Faux	Faux	Vrai	Faux	Faux
Faux	Vrai	Vrai	Faux	Vrai
Vrai	Faux	Faux	Faux	Vrai
Vrai	Vrai	Faux	Vrai	Vrai

Les outils de base de l'algorithmique

Remarque: les valeurs booléens **"VRAIE"** ou **"FAUSSE"** peuvent être aussi représenter par les valeurs binaires : **1** ou **0**

Les **tables de vérité** des opérateurs logiques peuvent alors s'écrire:

A	B	L=A ET B
0	0	0
0	1	0
1	0	0
1	1	1

L'opérateur **ET** peut être matérialisé physiquement par **deux interrupteurs en série** pour allumer une lampe **L**

A	B	L=A OU B
0	0	0
0	1	1
1	0	1
1	1	1

L'opérateur **OU** peut être matérialisé physiquement par **deux interrupteurs en parallèle** (**//**) pour allumer une lampe **L**

A	L=NON A
0	1
1	0

L'opérateur **NON** peut être matérialisé physiquement par un **interrupteur** normalement fermé au départ pour allumer une lampe **L**

33

Les outils de base de l'algorithmique

Type Chaîne de caractère : permet de manipuler une suite de caractères, ainsi de représenter des mots ou des phrases.

- **Exemple :** "bonjour, Monsieur".
- **Remarque:** Une **chaîne de caractères** est toujours notée entre guillemets **" "**, pour éviter toute **confusion** entre des **nombres** et des **suites de chiffres**.
- Par exemple, **423** peut représenter :
  - ✓ le **nombre entier 423** (*quatre cent vingt-trois*),
  - ✓ ou la suite de caractères 4, 2, et 3 notée : **"423"**

34

Les expressions

Les **expressions** peuvent être des expressions arithmétiques (**calcul**) ou des expressions logiques (**relation**). Ce sont des combinaisons entre des **variables** et des **constantes** reliées par des **opérateurs**.

1. *Les expressions arithmétiques*:

Exemple : **x * 53.4 / (2 + pi)**

- **L'ordre** selon lequel se déroule chaque opération est **important**.
- Afin d'**éviter les ambiguïtés** dans l'écriture, on se sert des **parenthèses** et la **priorité** entre les opérateurs arithmétiques :

Ordre de priorité des opérateurs arithmétiques

Priorité	Opérateurs	
1	-	signe négatif (oprérateur unaire)
2	( )	parenthèses
3	^	puissance
4	* et /	multiplication et division
5	+ et -	addition et soustraction

Remarque: **En cas de conflit** entre deux opérateurs de **même priorité**, on commence par celui situé à **gauche**.

35

Les expressions

2. *Les expressions logiques:*

Ce sont des combinaisons entre des **variables** et des **constantes** reliées par des **opérateurs** de **comparaisons** (**=, <, <=, >, >=, #**) **et/ou** des combinaisons entre des **variables** et des **constantes logiques** reliées par des **opérateurs logiques** (**NON , ET, OU,** …).

Exemple : **A+1<=2** **(A<B) ET (B=8)**

Ici encore, on utilise les **parenthèses** et l'**ordre de priorité** entre les différents opérateurs pour résoudre les **problèmes de conflits**.

Priorité	Opérateur
1	NON
2	ET
3	OU

opérateurs logiques

Priorité	Opérateur
1	>
2	>=
3	<
4	<=
5	=
6	#

opérateurs relationnels

36

Les expressions

3. Application 1 :

- L'utilisation la plus fréquente des **opérateurs de comparaison** est la **comparaison des variables et des expressions numérique**;
- La plus simple est la **comparaison** directe de **deux variables numériques**. Par exemple:

x	y	x=y	x≠y	x<y	x>=y	x>y	x<=y
1	1	Vrai	Faux	Faux	Vrai	Faux	Vrai
1	2	Faux	Vrai	Vrai	Faux	Faux	Vrai
2	1	Faux	Vrai	Faux	Vrai	Vrai	Faux

- Mais, on peut bien sur comparer des **expressions numériques** quelconques. Par exemple, pour **x=1** et **y=2** on aurait:

Expression logique	(x+1)=y	(x+1)*(y+1)=6	x+1<=2	x-1<-y
Valeur	Vrai	Vrai	Vrai	Faux

37

Les expressions

4. Application 2 :

1. Ecrire la formule suivante sous forme d'une expression arithmétique :

$$\frac{(3-xy)^2-4ac}{2x-z}$$

Priorité	Opérateurs	
1	-	signe négatif (oprérateur unaire)
2	( )	parenthèses
3	^	puissance
4	* et /	multiplication et division
5	+ et -	addition et soustraction

2. Quel est l'ordre d'exécution (priorité des différents opérateurs) de l'expression suivante :

((3 * a) – x ^ 2) – (((c – d) / (a / b)) / d)

Réponses:

1- ((3 – x * y) ^ 2 – 4 * a * c) / (2 * x – z)

2- ((3 ❶ * a) ❸ – x ❷ ^ 2) ❽ – (((c ❹ – d) ❻ / (a ❺ / b)) ❼ / d)

38

Instructions élémentaires en algorithmique

1. Introduction

Un **algorithme** informatique réalise, en général, **trois chose**:

- ✓ ***Il lit des données en entrée***
- ✓ ***Il effectue des calculs***
- ✓ ***Il écrit des données (résultats) en sortie***

Les **instructions élémentaires** sont celles qui figurent le plus souvent dans tous les algorithmes. Elles sont au nombre de trois:

1. **L'affectation,**
2. **Les instructions d'entrée de données,**
3. **Les instructions de sortie de données.**

39

Instructions élémentaires en algorithmique

2. l'affectation

- L'opération **affectation** permet d'assigner une **valeur** à une **variable**.
- Elle est **représentée**, en algorithmique, par une **flèche orientée de droite vers la gauche** :

Variable ← Valeur

L'opération **d'affectation** présente certaines **possibilités** et impose certaines **conditions**.

2.1 Possibilités

Le **membre** droite d'une affectation (**valeur**) peut être soit:

- Une **variable** de même type que la **variable : Variable1 ← Variable2**
- Une **constante** de même type que la **variable : Variable1 ← 4**
- Une **expression** dont l'**évaluation produit un résultat final** de même type que la **variable : Variable1 ← (3*A+2)*B-4**

40

Instructions élémentaires en algorithmique

2.2 Conditions

Une **affectation** est exécutable si et seulement si:

- Le **membre gauche** de l'affectation est une **variable déclarée.**

  *Par conséquent les **écritures suivantes n'ont pas de sens:***

  8 ← A ou A+B ← C

- La **partie droite** de l'affectation représente une **valeur bien définie.**

  *Par conséquent la **2ème affectation dans l'exemple suivant n'est pas exécutable** :*

  A ←0
  V ←1/A

41

Instructions élémentaires en algorithmique

Les **types** de deux parties de **l'affectation** sont les **mêmes**.

*Par conséquent la **2ème affectation dans l'exemple suivant n'est pas exécutable:***

```
Var   A: entier
      C: caractère
Début
      A ← 6
      C ←A
Fin
```

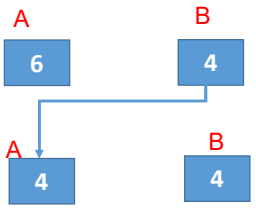
Exemples d'utilisation de l'affectation :

Si **A← 6** et **B← 4**,

En **mémoire** : **A=6** et **B=4**,

Après affectation **A ← B**

On aura : **A=4** et **B=4**

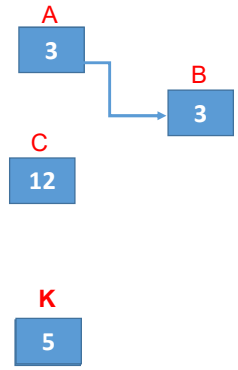


42

Instructions élémentaires en algorithmique

Exemples d'utilisation de l'affectation

- **A ← 3**
  donc A=3
  B ← A
  on aura donc B=3
  C ← (A+B)*2
  on aura donc C=12
- **K ← 4**
  donc K=4
  K ← K+1
  on aura donc K=5



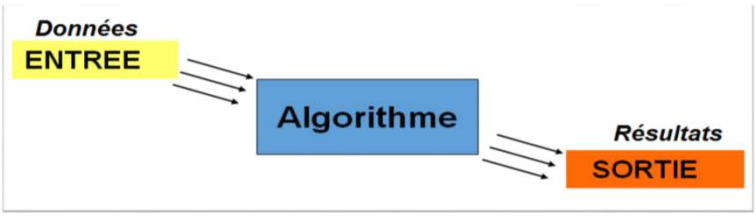
Remarque : l'affectation **détruit complètement et définitivement** le contenu précédent de la variable affectée si elle en avait un.

43

Instructions élémentaires en algorithmique

Instructions d'entrée/ sortie

Pour bien fonctionner, un **algorithme** doit en général **acquérir des données (entrées)** sur un périphérique (**clavier**) et fournir des **résultats (sorties)** sur un autre périphérique(**écran**).



Pour réaliser ces tâches fondamentales en programmation, on a donc besoin de **deux opérations distinctes:**

1. **Lire(vE): permet d'affecter** à la **variable** (**vE**) , la valeur **lue** sur le **périphérique d'entrée (clavier),**
2. **Ecrire(vS): permet de transformer** la **valeur** (**vS)** vers le **périphérique de sortie (écran)**

44

Instructions élémentaires en algorithmique

Instructions d'entrée/ sortie

Remarques

- Pour la lecture, l'identificateur (**vE**) doit être une variable déclarée. *Par conséquent les écritures suivantes n'ont pas de sens:*

  Lire(8); !!

  Lire(A+B); !!

- Pour l'écriture, la valeur de sortie (**vS**) peut être soit:

  Une variable déclarée: **Ecrire(A);**

  Une constante: **Ecrire(8);**

  Une expression : **Ecrire(3*B+4*D-2);**

- Un **algorithme** doit être **interactif**. Pour faciliter l'utilisation de l'algorithme on affiche des messages explicatifs qui **doivent être écrits entre guillemets (" ").**

  Ecrire("Entrez deux valeurs entiers x et y ");

45

Instructions élémentaires en algorithmique

Instructions d'entrée/ sortie

Exemple 1

Qu'obtient t-on à l'écran après l'exécution des instructions suivantes :

```
x ← 5;
Ville ← "Meknes";
Ecrire(x);
Ecrire(x*x);
Ecrire("Ville");
Ecrire("Ville = ", Ville);
```

→

```
5
25
Ville
Ville = Meknes
```

46

Instructions élémentaires en algorithmique

Exemples 2

1. Quelles sont les **valeurs successives** prises par les **variables X et Y** suite aux instructions suivantes:

```
X ←1;
Y ← -4;
X ← X+ 3;
X← Y-5;
Y← X+2;
Y← Y-6;
```

X	1	1	4	-9	-9	-9
Y	----	-4	-4	-4	-7	-13

2. Soit deux variables quelconque (nombre ou caractère) **X et Y** ayant respectivement comme **valeurs a et b**. Quelles sont les affectations qui donneront à **X la valeur b** et à **Y la valeur a**?

Analyse: la première idée peut être d'écrire :

```
X←Y;
Y←X;
```

Mais ça ne marche pas,

les deux variables **X et Y** se retrouvent avec la même valeur **b**!

➔ Il faut mettre la valeur de **X** de coté dans *une autre variable intermédiaire* pour ne pas la perdre.

47

Instructions élémentaires en algorithmique

Algorithme

Algorithme permutation;
Var
X, Y, Z: entier;
Début
Ecrire("Entrez deux valeurs entières X et Y ");
Lire(X, Y);
Z← X;
X←Y;
Y←Z;
Ecrire("la valeur de X et Y après permutation est:", X, Y);
Fin

Vérification!

```
X ←1;
Y ← -4;
Z ← X;
X← Y;
Y← Z;
```

X=1	Y=	Z=
X=1	Y=-4	Z=
X=1	Y=-4	Z=1
X=-4	Y=-4	Z=1
X=-4	Y=1	Z=1

48

Instructions élémentaires en algorithmique

Exemple 3 : Donner toutes les raisons pour lesquelles l'algorithme suivant est incorrect :

```
Algoritme Incorrect
 x, y : Entier;
     z : Réel;
Début
z ← x + 2;
y ← z;
x*2 ← 3 + z;
y ←5y + 3;
Fin
```

Cet algorithme est incorrect pour plusieurs raisons :

- **ligne 1** : Le mot **Algorithme** s'écrit avec un **« h »** au milieu
- **ligne 2** : La déclaration des variables commence par le mot **« Var »**
- **ligne 5** : La valeur de **x est indéterminée**
- **Ligne 6** : **incompatibilité de type** (un résultat réel affecté à une variable de type entier)
- **Ligne 7** : Le **membre gauche** d'une affectation doit être une **variable**
- **Ligne 8** : Il faut écrire **5*y** et non **5y**.

49

Instructions élémentaires en algorithmique

Exemple 4 : Ecrire un algorithme qui permet de calculer la somme de deux nombres réels A et B.

Solution

```
Algorithme Somme;
Var
   a, b, S : réel;
Début
        Ecrire("Entrez deux nombres");
        Lire(a, b);
        S ← a+ b;

        Ecrire("La somme de nombres a et b est:", S);
Fin
```

Instructions élémentaires en algorithmique

Remarque :
La dernière instruction de l'algorithme précédent peut être modifier comme suite:

Ecrire(”La somme de deux nombres ”, a, ” et ”, b, ” est :", S);

Exemple :

Si a=3 et b=8, l'instruction ci-dessus s'exécutera ainsi:

La somme de deux nombres 3 et 8 est :11

Exemple 5 : Ecrire un algorithme qui demande un nombre entier à l'utilisateur, puis calcule et affiche le double de ce nombre.

Instructions élémentaires en algorithmique

```
Algorithme doubleDevaleur;
Var
   val, dval: entier;
Début
   val← 12;
   dval←val*2
   Ecrire (val, dval);
Fin
```

Exécution:
12 24

N'est pas un bon algorithme (*Ni clair Ni général*)

```
Algorithme doubleDevaleur;
Var
   val, dval: entier;
Début
   Ecrire(”donner un entier: ”);
   Lire(val);
   dval←val*2
   Ecrire (”le double est ” , dval);
   Ecrire(”le double de :”,val, ”est ”, dval);
Fin
```

Exécution:
Donner un entier: **12**
Le double est **24**
Le double de **12** est **24**

Un bon algorithme (*Clair et Général*)

Instructions Conditionnelles et Alternatives