

# UNIVERSITE ASSANE SECK ZIGUINCHOR



## Algorithmique et Programmation en Langage Pascal

Dr Ousmane DIALLO  
odiallo@univ-zig.sn

# PLAN

- I. Introduction à l'informatique**
- II. Notions d'algorithme et de programme**
- III. Concepts de base de l'algorithmique**
- IV. Concepts de base du langage Pascal**
- V. Structures de contrôle**
- VI. Les tableaux et les chaînes de caractères**
- VII. Les sous programmes**
- VIII. Les enregistrements**

# PLAN

**IX. La récursivité**

**X. Les fichiers en Pascal**

**XI. Notion de complexité d'un algorithme**

**XII. Algorithmes de tri**

**XIII. Listes chaînées, piles et files**

**XIV. Arbres**

**XV. Tables de hachage**

# I. Introduction à l'informatique

## □ I.1. Généralités et bref historique

- Proposé par Philippe Dreyfus en 1962, le mot **informatique** est une contraction d'information et automatique. L'informatique désigne alors l'automatisation de l'information ou le traitement automatique de l'information.
  - L'information se présentant sous forme de textes, de nombres, d'images, de sons, de vidéos, etc.
- L'outil utilisé pour traiter l'information de manière automatique s'appelle un **ordinateur**. Ce nom a été proposé par Jacques Perret en 1954. Ce mot était à l'origine un adjectif qui signifiait "qui met de l'ordre", "qui arrange". L'anglais, plus restrictif, utilise le terme de **computer** qui peut se traduire par calculateur, machine à calculer.
- L'informatique désigne donc un **concept**, une **science** (c'est pourquoi en anglais on parle de **computer science**), tandis que l'ordinateur est un outil, une machine conçue pour réaliser des opérations informatiques.

# I. Introduction à l'informatique

## □ I.1. Généralités et bref historique

- L'ordinateur n'est capable de fonctionner que s'il y a **apport d'information** par l'utilisateur. Le traitement automatique implique un traitement qui suit des règles qui peuvent être identifiées et également programmées dans un ordinateur.
- L'utilisateur fournit des données à l'ordinateur (entrées ou **Input** en anglais), qui traite ces informations, puis renvoie les résultats ou réponses à l'utilisateur (sorties ou **Output** en anglais).



FIGURE 1.1 – traitement information.

# I. Introduction à l'informatique

## ❑ I.1. Généralités et bref historique

❑ Le but de l'informatique est donc de faire accomplir par l'ordinateur ce que fait l'homme, avec des gains de :

- Rapidité,
- Précision,
- Efficacité.

- ✧ **1500 av. J.C. (?), Le Boulier.** Il est toujours utilisé dans certains pays.
- ✧ **En 1641, La Pascaline:** machine à calculer mécanique de Blaise PASCAL.
- ✧ **En 1937, le Mark I d'IBM** permet de calculer 5 fois plus vite que l'homme. Il est constitué de 3300 engrenages, 1400 commutateurs et 800 km de fil. Les engrenages seront remplacés en 1947 par des composants électroniques.
- ✧ **En 1946, ENIAC:** premier grand ordinateur universel. 30 tonnes. 18 000 tubes électroniques.
- ✧ **En 1947:** invention du transistor qui va permettre de rendre les ordinateurs moins encombrants et moins coûteux.
- ✧ **En 1948, UNIVAC** (UNIVersal Automatic Computer).

# I. Introduction à l'informatique

## I.1. Généralités et bref historique

- ✧ En 1958: mise au point du **circuit intégré**, qui permet de réduire encore la taille et le coût des ordinateurs.
- ✧ En 1960, l'**IBM 7000**: premier ordinateur à base de transistors.
- ✧ En 1971, l'**Intel 4004**: le premier **microprocesseur** voit le jour. De la taille d'un ongle. Composé de 2 300 transistors. Puissance de calcul comparable à celle de l'**ENIAC** !
- ✧ En 1978, l'ordinateur familial (oric, sinclair, etc.)
- ✧ En 1980, **IBM-PC** (Personal Computer).
- ✧ En 1984, **Macintosh d'APPLE**.
- ✧ De nos jours...

# I. Introduction à l'informatique

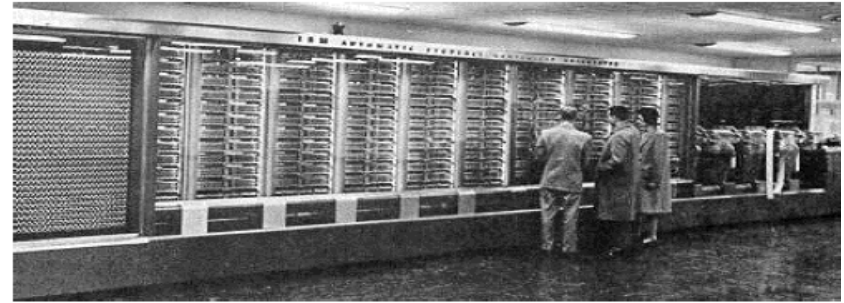
## I.1. Généralités et bref historique



Boulier 5/1



La Pascaline



Mark 1 (IBM)



L'ENIAC



UNIVAC



IBM 7000



IBM PC



Mackintosh



... loi de Moore



# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

- ❑ **Définition (Hachette):** Machine capable d'effectuer automatiquement des opérations arithmétiques et logiques (à des fins scientifiques, administratives, comptables, . . . ) à partir de programmes définissant la séquence de ces opérations.
- ❑ **Autre définition:** Un ordinateur est une machine de traitement de l'information capable:
  - d'acquérir et de stocker des informations
  - d'effectuer des traitements et de restituer des informations.
- ❑ Il est composé principalement de deux parties:
  - une partie matérielle (**Hardware** en anglais)
  - une partie logicielle (**Software** en anglais)

# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

- ❑ Le **matériel** est l'équipement physique, tels que les boîtiers, les lecteurs de disquettes, les claviers, les moniteurs, les haut-parleurs et les imprimantes.
- ❑ C'est la partie physique et palpable du système informatique et est divisée principalement en trois parties :
  - Une unité centrale appelée CPU (central processor unit), contenant le processeur et les registres
  - Des composants matériels autour (mémoires centrale et secondaire, cartes diverses ...)
  - Les organes périphériques (écran, clavier, imprimante...)

# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

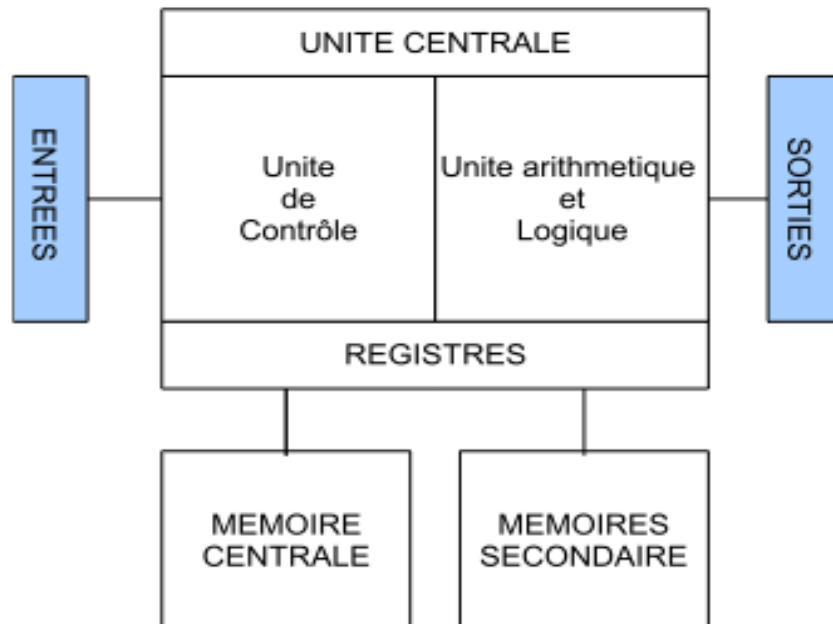


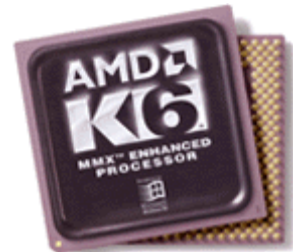
FIGURE 1.2 – Architecture.

# I. Introduction à l'informatique

## ❑ I.2. Architecture générale d'un ordinateur

❑ **Le CPU (Central Process Unit) ou Unité Centrale:** est le cœur de l'ordinateur.

- ❑ Il est composé de deux parties principales : l'unité arithmétique et logique (UAL) et l'unité de commande et de contrôle (UCC) encore appelée unité de contrôle (UC).
- L'UAL permet d'effectuer des opérations simples, comme l'addition de deux nombres, la permutation, etc. L'ensemble des opérations réalisables par l'unité de calcul est défini par le constructeur de l'unité centrale.
  - L'UCC a pour rôle de lire une instruction d'un programme en mémoire et de la faire réaliser par l'UAL en lui fournissant les opérandes et l'opération à réaliser puis de passer à l'instruction suivante du programme.



# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### ☐ Les mémoires centrale et secondaire

#### ☐ Il existe deux types de mémoire centrale :

- La mémoire vive, ou encore **RAM** (Random Access Memory), Mémoire à Accès Aléatoire : lecture ou écriture. Elle contient les données et les programmes dits "volatiles"; son contenu s'efface dès que l'alimentation en énergie est coupée.
- La mémoire morte, ou **ROM** (Read Only Memory), Mémoire à Lecture Seule. Elle contient des données et des programmes figés, son contenu est fixé à la fabrication de l'ordinateur et ne peut jamais être modifié.

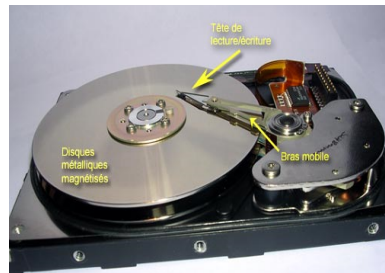


# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### ❑ Les mémoires centrale et secondaire

- ❑ Les mémoires secondaires contiennent des programmes et des données de façon permanente, puisqu'elles conservent l'information même si la machine est éteinte. Il peut s'agir de:
  - **disques durs:** dispositif de stockage de l'ordinateur, utilisant un empilement de plateaux recouverts d'une surface magnétisée pour enregistrer des données ou des programmes. Il existe des disques durs de différentes capacités.



- **disquettes:** disque en plastique souple de 3,5 pouces, recouvert de métal. La capacité de stockage d'une disquette standard est d'environ 1 Mo de données.

- **CD-ROMs ou DVD-ROMs**



# I. Introduction à l'informatique

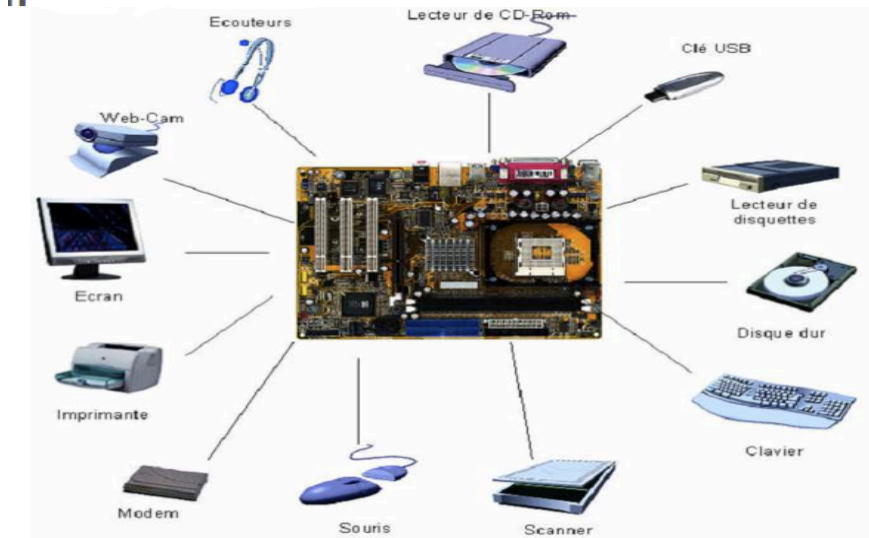
## I.2. Architecture générale d'un ordinateur

### ☐ Les organes périphériques

☐ Ils permettent surtout la communication avec l'utilisateur.

☐ On distingue:

- Les **périphériques d'entrée** qui permettent la collecte d'informations. Le clavier, la souris, le CD-ROM ou DVD-ROM sont des périphériques d'entrée.
- Les **périphériques de sortie** qui permettent la diffusion d'informations. L'écran, les imprimantes, ... sont des périphériques de sortie.



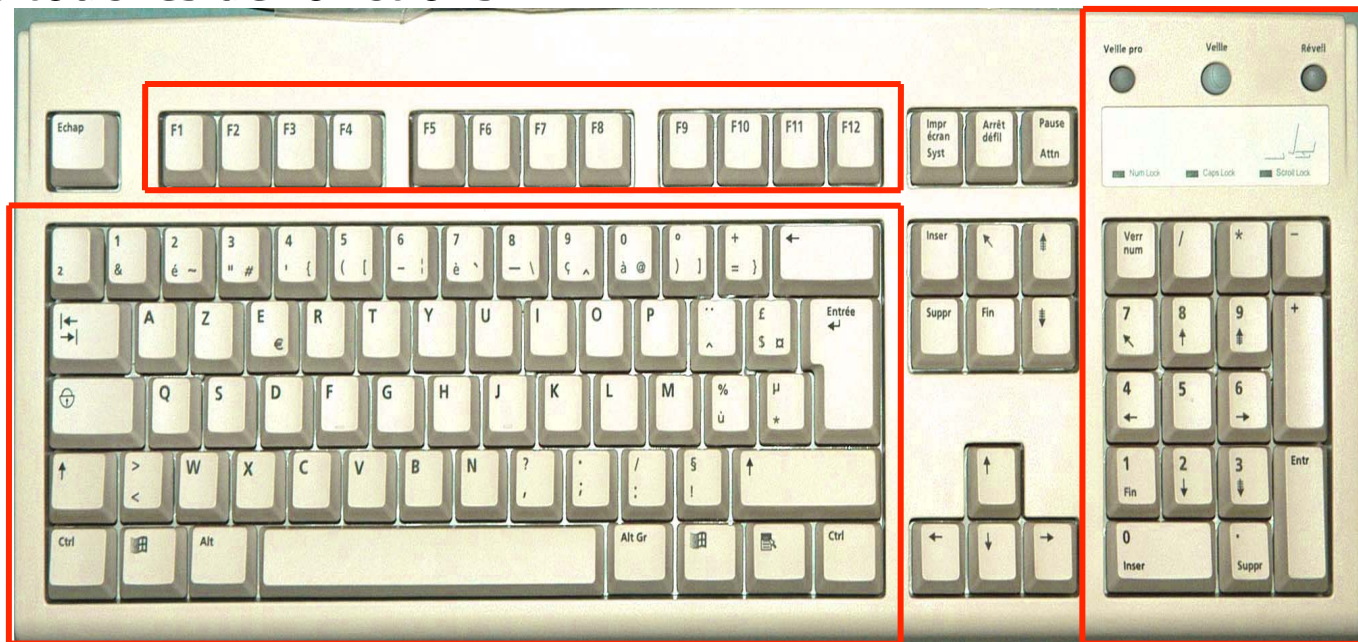


# I. Introduction à l'informatique

## □ I.2. Architecture générale d'un ordinateur

□ **Notion du clavier:** Le clavier est subdivisé en plusieurs compartiments que sont:

- Le pavé Alpha Numérique
- Le pavé Numérique
- Les touches de fonctions

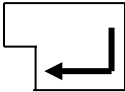





# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### Le pavé Alpha Numérique:

- Il contient en plus des alphabétiques, numériques et les caractères spéciaux, d'autres touches telles que:
  - **Shift** qui est la touche seconde fonction. Elle permet la mise momentanée du clavier en mode majuscule.
  - **Caps Lock**: elle permet de bloquer le clavier en mode majuscule
  - **Back Space**: elle permet d'effacer le caractère qui est à la gauche du curseur.
  - **entrée ou validation**. 
  - **Alt. Gr** : elle permet d'accéder au troisième caractère du clavier.
  - **Verr Num** : elle permet d'activer le pavé numérique. 

# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### Le pavé Alpha Numérique:

- ☐ le type de clavier se reconnaît par ses six premières touches alphabétiques du pavé alphabétique. Nous avons :
- Le clavier AZERTY qui est de type français ;
  - Le clavier QWERTY qui est de type anglais ;
  - Le clavier QWERTZ qui est de type allemand

# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### ☐ pavé Numérique

- Lorsque le voyant **[Num Lock]** du clavier est allumé, le pavé numérique donne aisément accès aux chiffres et aux opérateurs arithmétiques. Il est plus facile de frapper les chiffres sur le pavé numérique que sur le pavé alphanumérique du clavier. Pour allumer le voyant, frapper la touche **[Num Lock]**.

### ☐ Touches de fonctions

- Elles permettent souvent de sélectionner telle ou telle partie d'un programme que l'on utilise. Elles portent chacune la lettre **F** suivi d'un numéro allant de 1 à 12.

# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### Notion de souris:

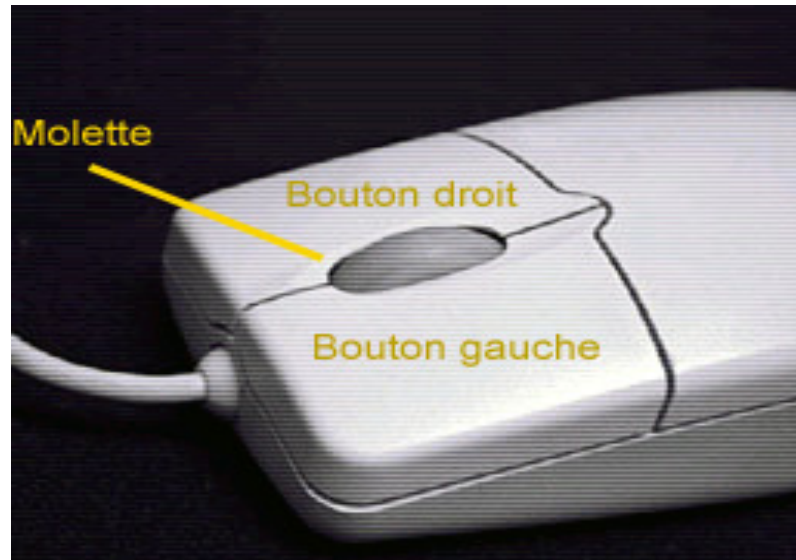
- ☐ La souris sous Windows est un élément essentiel ce qui fait qu'on doit bien maîtriser son utilisation. Selon sa position dans la fenêtre la souris change d'aspect, donc change de fonctionnalité.
- ☐ La souris dispose généralement de 2, voire 3 boutons.
- ☐ La plupart des manipulations s'effectuent avec le bouton de gauche.
- ☐ Le bouton de droite est réservé pour le menu contextuel.
- ☐ Le déplacement de la souris sur son tapis déplace la flèche (le pointeur) sur l'écran.

# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### Notion de souris:

- ❑ Le troisième bouton s'il est présent, est la **roulette** et il permet de faire défiler du texte par un nombre de ligne paramétrable grâce à la fenêtre de configuration de la souris.



# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### Notion de souris:

#### ☐ Un clic

- Le clic consiste à émettre une pression sur le bouton gauche comme vous appuieriez sur un interrupteur. Il est parfois appelé **cliqué** selon son contexte. Il permet d'entrer dans un menu, d'activer un bouton ou de positionner le curseur dans le document.

#### ☐ Un cliqué tiré

- Le cliqué tiré consiste à cliquer sur le bouton de la souris et à maintenir celui-ci appuyé, puis à déplacer le pointeur de la souris jusqu'à un endroit précis. Il est parfois appelé « **étendre la sélection** ». Permet de sélectionner une zone.

#### ☐ Un double clic

- Le double clic consiste à émettre une double pression rapide sur le bouton gauche de la souris. Permet l'activation de commande afin d'éviter certaines confirmations ou permet de sélectionner des zones.

#### ☐ Clic droit

- Permet l'activation de commande rapide afin d'éviter d'entrer dans les menus.

# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### La partie logicielle:

- ❑ Le terme **logiciel** désigne les programmes utilisés pour faire fonctionner le système.
- ❑ Les logiciels, également appelés programmes, précisent à l'ordinateur la manière dont il faut opérer. Ces opérations peuvent comprendre l'identification des informations, leur accès et leur traitement.
- ❑ Un programme est essentiellement une séquence d'instructions, qui décrit le mode de traitement des données.
- ❑ Nous distinguons deux **types de logiciels**:
  - les **systèmes d'exploitation**
  - les **logiciels d'application**.

# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### Le système d'exploitation SE (ou OS - operating system)

- ☐ Un **SE** est un programme sans lequel il n'est pas possible de communiquer avec l'ordinateur. C'est un logiciel constitué d'un ensemble de programmes destinés à faire fonctionner l'ordinateur et ses périphériques.
- ☐ Il fournit également l'environnement de fonctionnement des applications utilisées pour accéder aux ressources de l'ordinateur.
- ☐ Il effectue des tâches de base, telles que la reconnaissance des entrées au clavier ou à la souris, l'envoi des sorties sur l'écran vidéo ou sur l'imprimante, le suivi des fichiers sur les lecteurs et le contrôle des périphériques, tels que les imprimantes et les modems.



# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur système d'exploitation SE (ou OS - operating system)

□ Les OS les plus utilisés sont:

- **MS DOS** ( MicroSoft Disk Operating System) pour PC
- **Microsoft Windows 95 - 98 - 2000 – XP – 7 - 8** pour PC, nettement plus convivial que MS DOS
- **Mac OS (OSX)** pour Macintosh : **très convivial**
- **UNIX, LINUX et ses distributions** (Debian, Red Hat, SuSe, openSuSe, Fedora, Mandriva, Ubuntu, ...)

# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### Le système d'exploitation SE (ou OS - operating system)

- ☐ Les systèmes d'exploitation sont tributaires de la **plateforme**, c'est-à-dire qu'ils sont conçus pour un type spécifique d'ordinateurs.
  - le système d'exploitation Windows est conçu pour les ordinateurs individuels compatibles IBM (PC).
  - Mac OS, en revanche, ne fonctionne qu'avec des Macintosh.
- ☐ Le PC et le Macintosh représentent des plateformes.
- ☐ Une plateforme est un système informatique sur lequel différents programmes peuvent fonctionner

# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### Les logiciels d'application

- ❑ Le **logiciel d'application** accepte les entrées de l'utilisateur, puis les manipule pour obtenir un résultat. Ce résultat est appelé sortie.
- ❑ Les applications sont des programmes conçus pour effectuer une fonction spécifique pour l'utilisateur ou pour un autre programme d'application.
- ❑ Parmi les exemples d'applications figurent les traitements de texte, les bases de données, les tableurs, les navigateurs, les outils de développement Internet et les outils de conception graphique.

# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### Les logiciels d'application

Type de logiciel	Définition	Exemples
Edition de textes	Saisie de textes simples, sans mise en page sophistiquée.	Bloc-notes (PC) SimpleText (MAC)
Traitement de textes	Saisie de texte avec mise en page sophistiquée, insertion d'images et de tableaux, etc.	Word 97 et 2000 et XP sur PC version 98 et 2001 sur Mac
Logiciels graphiques	Dessins et images	Paint Shop Pro (PC) Adobe Photoshop (PC et Mac) Adobe Illustrator (PC et Mac)
Tableur	Réalisation de tableaux de calculs (factures, bulletins de salaire, etc.)	Lotus Excel
Logiciels de Bases de Données	Réalisation de listes structurées d'éléments et leur exploitation.	DBase (PC) 4 <sup>e</sup> Dimension (Mac et PC) Access (PC)
SGBD	Système de gestion de bases de données : logiciel puissant pour la gestion et l'interrogation des bases de données.	Oracle Sybase Ingres
Logiciels intégrés	Logiciels incluant à la fois des fonctionnalités de traitement de texte, dessin, tableur et base de données.	Microsoft Works (Mac et PC) Claris Works (Mac et PC)
Autres	Logiciels spécifiques à des domaines particuliers.	Architron

# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### Le langage machine

- ❑ Il est le seul directement compréhensible par la machine et est une succession de zéros et de uns définissant des opérations précises à effectuer :
  - **Toute information étant donc codée avec uniquement des 0 et des 1.**
- ❑ Ces chiffres binaires 0 et 1 à partir desquels on construit des nombres plus grands sont appelés bits (pour binary digit).
- ❑ Toutes les données manipulées sont ainsi représentées par des séquences de bits:
  - Un caractère : 8 bits (code entre 0 et 255),
  - Un entier : 32 bits,
  - Un réel en virgule flottante ( 32 ou 64 bits),
  - Les sons : décomposés en échantillon,
  - Les images décomposées en pixels.

# I. Introduction à l'informatique

## I.2. Architecture générale d'un ordinateur

### Le langage machine

#### ☐ Unités de mesure:

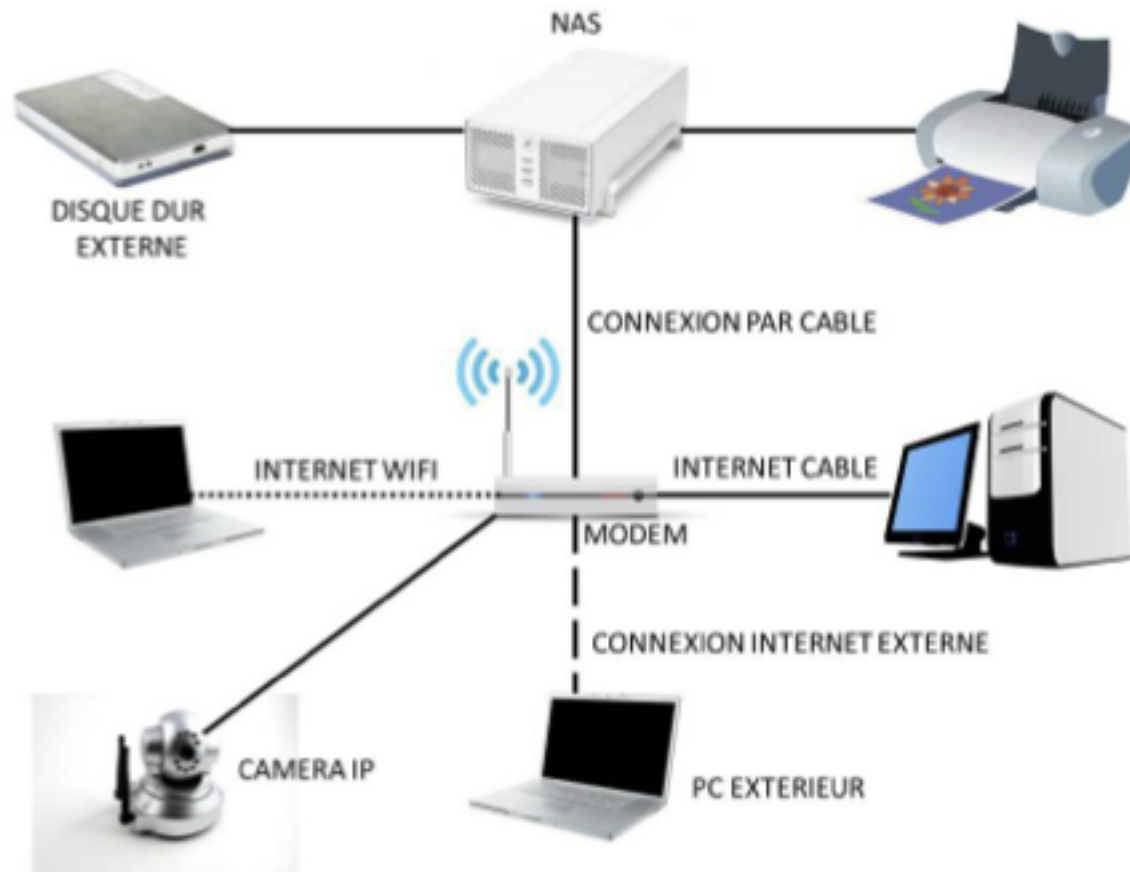
- Un Octet (ou Byte) est un ensemble de 8 bits.
- Les longueurs couramment utilisées sont des ensembles de 16, 32 ou 64 bits.
- Un Kilo (ou 1 K) correspond à 1024 bits, soit  $2^{10}$  bits.
- Un Méga (ou 1 M) correspond à 1000 K.
- Un Giga est un ensemble de 1000 Mégas.

# I. Introduction à l'informatique

## I.3. Réseau d'ordinateurs

- ☐ On parlera de réseau lorsque plusieurs ordinateurs sont connectés entre eux.
  - **Réseau local:** dans un même lieu, à l'aide d'un câble ou d'un wifi.
  - **Réseau distant:** ordinateurs distants, la liaison est réalisée à travers les lignes téléphoniques ou dédiées, les satellites etc.
- ☐ Un réseau permet le partage des ressources, la communication, le transfert d'informations. Il évite la duplication des logiciels et des informations sur tous les ordinateurs.
- ☐ **Internet** est le **réseau des réseaux**. Les ordinateurs du monde entier sont connectés entre eux à l'aide de câbles, de lignes téléphoniques et de satellites

# Réseau d'ordinateurs







**FIN  
CHAP1**

# UNIVERSITE ASSANE SECK ZIGUINCHOR



## Chapitre 2 Notion d'algorithme et de programmation

Dr Ousmane DIALLO  
odiallo@univ-zig.sn

# PLAN

- **Notion d'algorithme**
- **Propriétés d'un algorithme**
- **Organigramme d'un algorithme**
- **Programmation informatique**
- **Expression des algorithmes**
- **Langage de programmation**
- **Structures de données**
- **Structure générale d'un algorithme**

# II. Notions d'algo. Et progr.(1/16)

## II.1. Algorithmme

- ❑ L'algorithmique est une science très ancienne. Son nom vient d'un mathématicien arabe du IX<sup>ème</sup> siècle **Al-Khawarizmi**. Des mathématiciens grecs comme **Euclide** ou **Archimède** en ont été les précurseurs (calcul du PGCD de 2 nombres, calcul du nombre  $\pi$ ).
- ❑ L'**algorithmique** désigne actuellement la science qui étudie l'application des algorithmes à l'informatique.

## Mais qu'est ce qu'un algorithme?

### Définitions

- ❑ Suite finie, séquentielle de règles que l'on applique à un nombre fini de données, permettant de résoudre des classes de problèmes semblables. L'algorithme d'Euclide permet de trouver le P.G.C.D de deux nombre – Calcul, enchaînement des actions nécessaires à l'accomplissement d'une tâche. (**Petit Robert**)
- ❑ «Spécification d'un schéma de calcul, sous forme d'une suite finie d'opérations élémentaires obéissant à un enchaînement déterminé».
- ❑ «Etant donné un traitement à effectuer, un algorithme du traitement est l'énoncé d'une séquence d'actions primitives réalisant ce traitement »

# II. Notions d'algo. Et progr.(2/16)

## II.2. Propriétés d'un algorithme (1/2)

❑ Un algorithme doit être

❑ **PRECIS:** Il doit indiquer:

- l'ordre des étapes qui le constituent
- à quel moment il faut cesser une action
- à quel moment il faut en commencer une autre
- comment choisir entre différentes possibilités

❑ **DETERMINISTE**

- Une suite d'exécutions à partir des mêmes données doit produire des résultats identiques.

❑ **FINI DANS LE TEMPS ET PRODUCTIF**

- c'est-à-dire s'arrêter au bout d'un temps fini en fournissant le résultat escompté

# II. Notions d'algo. Et progr.(3/16)

## II.2. Propriétés d'un algorithme (2/2)

- Exemple : Algorithme de résolution de l'équation du premier degré  $AX + B = 0$

```
Lire les coefficients A et B
Si A est non nul
    alors
        affecter à X la valeur - B / A
        afficher à l'écran la valeur de X
sinon
    si B est nul
        alors
            écrire "tout réel est solution"
        sinon
            écrire "pas de solution"
```

### Précis

Suite linéaire d'opérations bien définies

### Déterministe

Avec les mêmes coefficients on obtiendra le même résultat

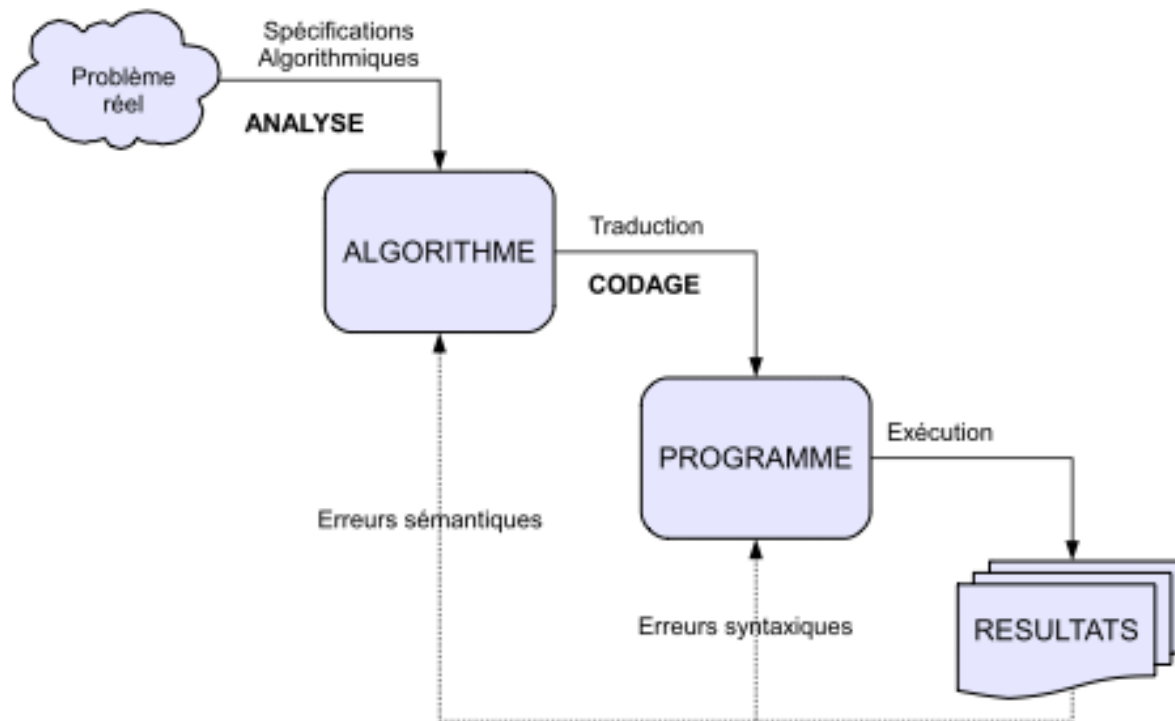
### Fini et productif

Se termine et donne pour chaque cas un résultat

# II. Notions d'algo. Et progr.(4/16)

## II.4. La programmation informatique (1/4)

- L'écriture d'un programme n'est qu'une étape dans le processus de programmation, comme le montre le schéma suivant :



**Les différentes étapes du processus de programmation**

# II. Notions d'algo. Et progr.(5/16)

## II.4. La programmation informatique (2/4)

❑ Les différentes phases de l'analyse et de la programmation informatique (implémentation sous forme de programme informatique d'un algorithme ou d'un procédé bien formalisé):

1. **Analyse du problème**
2. **Conception d'une solution** : algorithmique,  
choix de la représentation des  
données, choix de la méthode  
utilisée,
3. **Développement** : programmation,  
choix du langage de programmation,  
choix de la machine utilisée
4. **Tests et validation**
5. **Maintenance**
6. **Documentation du programme**



# II. Notions d'algo. Et progr.(6/16)

## II.4. La programmation informatique (3/4)

- ❑ **Analyse du problème:** L'analyse consiste à bien comprendre l'énoncé du problème et à définir les différentes étapes de sa résolution : Il est inutile et dangereux de passer à la phase suivante si vous n'avez pas bien discerné le problème. La question à se poser : **QU'EST-CE?**
- ❑ **Conception d'une solution:** C'est la phase algorithmique où l'on finalise les choix conceptuels, les structures et outils informatiques à utiliser pour résoudre le problème. Les questions : **AVEC QUOI? COMMENT?**
- ❑ **Le développement:** Plusieurs langages de programmation et architectures informatiques seront disponibles pour implémenter **l'algorithme qui ne doit être lié à aucun d'eux**. Il faut choisir ces éléments en restant cohérent avec la solution proposée.

# II. Notions d'algo. Et progr.(7/16)

## II.4. La programmation informatique (4/4)

- ❑ **Tests et validation:** Vérifier l'exactitude du comportement de l'algorithme, son bon déroulement. Si l'algorithme ne répond pas parfaitement à toutes les requêtes exprimées dans l'énoncé du problème, retournez à la phase n°1.
- ❑ **Maintenance:** activité qui assure l'entretien et le bon fonctionnement du programme tant et aussi longtemps qu'il sera utilisé.
- ❑ **Documentation du programme:** chaque étape énoncée doit être documentée pour assurer la pérennité du programme.

# II. Notions d'algo. Et progr.(8/16)

## II.5. Expression des algorithmes (1/2)

- Soit l'équation mathématique “  $ax^2 + bx + c = 0$  ”. Écrire un algorithme qui nous fait passer à une situation finale acceptable si notre situation initiale est acceptable.

**Analyse:**

**Description des situations initiales acceptables :**

- a, b et c sont des valeurs réelles bien définies et contenus dans des variables de même nom.
- Delta contient la valeur de l'expression mathématique «  $b^2 - 4*a*c$  »

**Description des situations finales acceptables :**

- a, b, c et Delta n'ont pas été modifiées.
- Nous avons affiché “ Pas de racine réelle. ” si et seulement si l'équation ne possède effectivement pas de racine réelle.
- Nous avons affiché “ Racine double. ” si et seulement si l'équation possède effectivement une racine réelle double.
- Nous avons affiché “ Deux racines réelles. ” si et seulement si l'équation possède effectivement deux racines réelles distinctes.
- Rien d'autre n'a été affiché ou imprimé depuis la situation initiale.

# II. Notions d'algo. Et progr.(9/16)

## II.5. Expression des algorithmes (2/2)

☐ Cet algorithme peut être spécifié par:

- Un organigramme
- Ou un pseudo-code

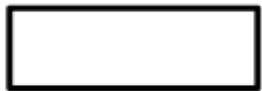
# II. Notions d'algo. Et progr.(10/16)

## II.3. Représentation schématique d'un algorithme (1/2)

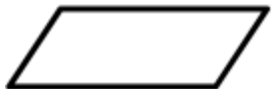
- La représentation schématique d'un algorithme est couramment désignée par le terme **organigramme** ou logigramme (norme **NF Z 67-010** ).
- L'organigramme permet donc la mise en œuvre de symboles représentant des traitements, des données, des liaisons... Il présente l'intérêt d'une visualisation globale mais reste limité aux études peu complexes (s'il ne tient plus sur une page, l'organigramme devient illisible....). En voici quelques symboles:



début, fin, interruption



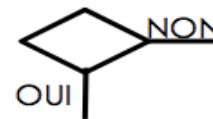
opérations, ou groupe d'opérations



entrées/sorties (lecture ou écriture)



Jonction



embranchement (test),  
conditions variables



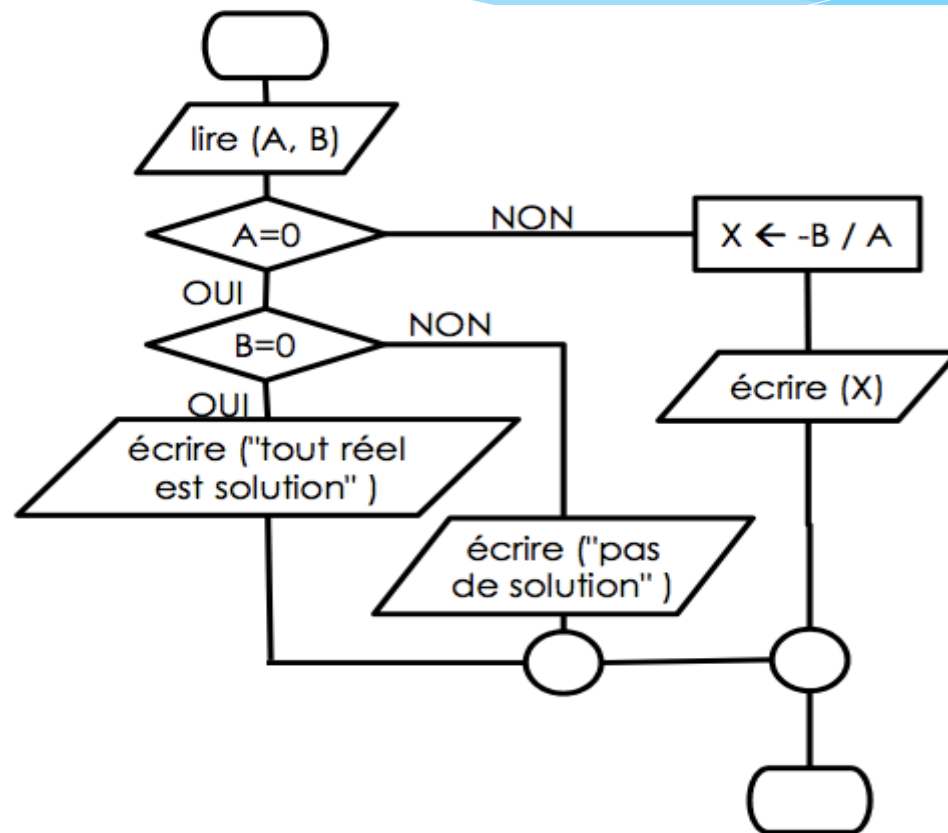
commentaires (indications)

Le sens général doit être de haut en bas, de gauche à droite (ex eq. 1er degré...).

# II. Notions d'algo. Et progr.(11/16)

## II.3. Représentation schématique d'un algorithme (2/2)

- Organigramme de résolution de l'équation du premier degré  $AX + B = 0$ .



# II. Notions d'algo. Et progr.(12/16)

## II.6. Langage de programmation (1/2)

□ Pour être compris et exécuté par un ordinateur, un algorithme doit être traduit dans un langage spécifique, qu'on appelle **langage de programmation**. On obtient ainsi ce qu'on appelle un **programme informatique** qui contient l'ensemble des actions consécutives que l'ordinateur doit exécuter. Ces actions sont appelées **instructions**.

□ Selon la méthode de traduction, on distingue:

- Les langages compilés
- Les langages interprétés.

# II. Notions d'algo. Et progr.(13/16)

## II.6. Langage de programmation (2/2)

### ❑ Les langages compilés

- Dans le cas d'un langage compilé (exemples: C, C++, Pascal...) , le programme réalisé, appelé programme source, est traduit complètement par ce qu'on appelle un **compilateur** avant de pouvoir être exécuté. La compilation génère un programme dit **exécutable**.
- Ce programme généré est autonome, c'est-à-dire qu'il n'a pas besoin d'un autre programme pour s'exécuter. Mais à chaque modification du fichier source (le programme source) il faudra le recompiler pour que les modifications prennent effet.

### ❑ Les langages interprétés

- Un programme écrit dans un langage interprété (exemples: Perl, Lisp, Prolog...) a besoin, pour chaque exécution, d'un programme annexe appelé **interpréteur** qui va lire le code source pour traduire et faire exécuter une à une, chacune des instructions. Dans ce cas, il n'y a pas de génération de programme exécutable.



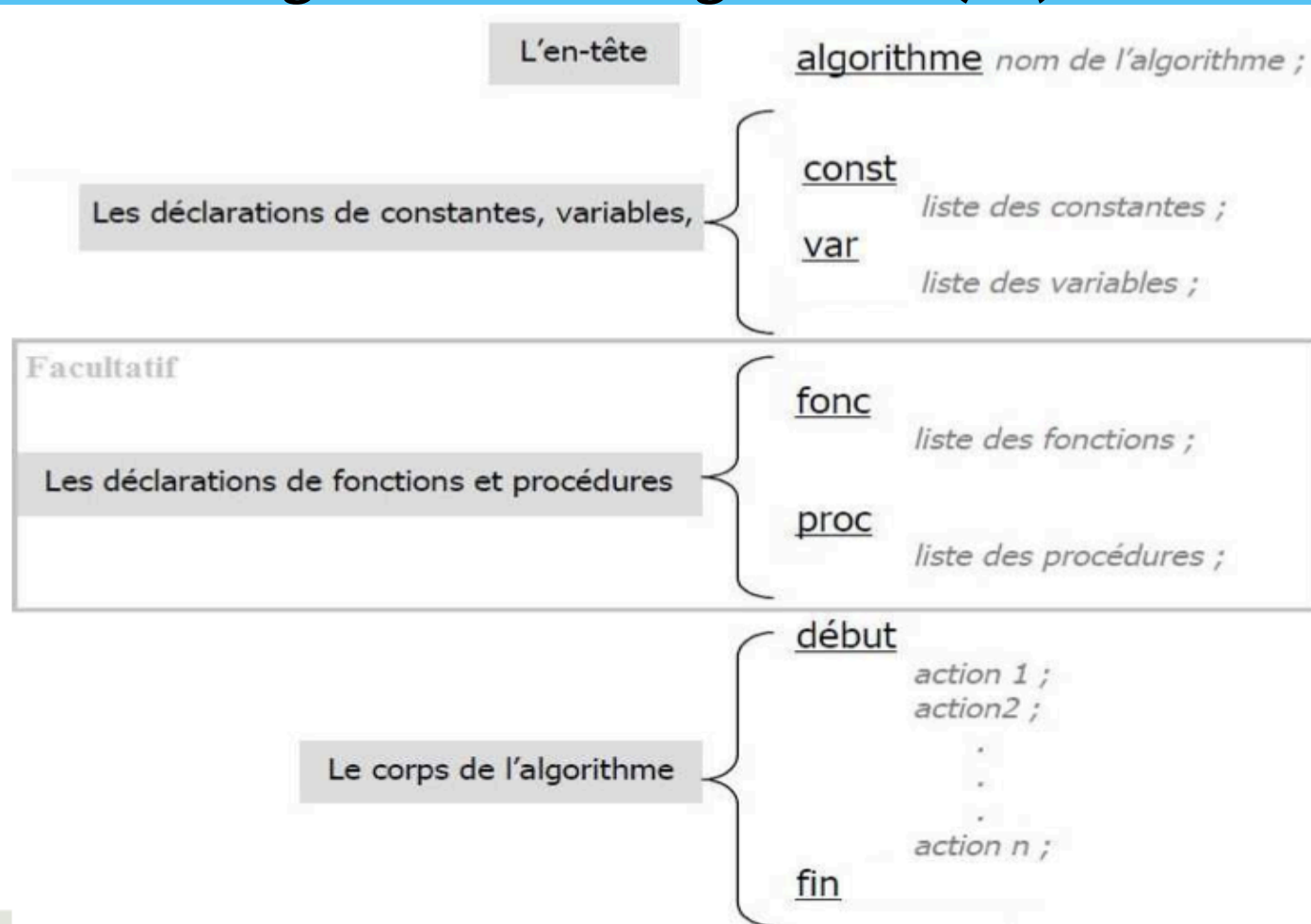
# II. Notions d'algo. Et progr.(14/16)

## II.7. Structures de données

- ❑ Une structure de données est une façon d'organiser des informations dans la mémoire de l'ordinateur pour en faciliter leur manipulation
- ❑ Il en existe plusieurs:
  - Tableau,
  - Structure
  - liste chaînée, etc.
- ❑ Il existe une forte dépendance entre structures de données et programmes:
  - influence sur la **simplicité** du programme
  - influence sur l'**efficacité** du programme

# II. Notions d'algo. Et progr.(15/16)

## II.8. La structure générale d'un algorithme (1/2)



# II. Notions d'algo. Et progr.(16/16)

## II.8. La structure générale d'un algorithme (2/2)

- ❑ **L'en-tête** : permet d'identifier tout simplement l'algorithme (donner un nom en un seul mot ou plusieurs mots composés reliés par – ou \_)
- ❑ **Les déclarations** : liste exhaustive des objets (constantes et variables principalement), structures et grandeurs utilisés et manipulés dans le corps de l'algorithme. Elle doit être située en début de l'algorithme (**tout doit être déclaré avant d'être utilisé**)
- ❑ **Le corps** : contient les instructions et opérations à exécuter une à une.
- ❑ **Les commentaires**: éventuellement présents dans l'algorithme, ils permettent de détailler, de commenter l'algorithme. Ces lignes de commentaires placées entre (**\* et \***) ne sont nullement exécutées, elles sont juste utilisées à titre d'information pour le programmeur (ou pour un autre utilisateur de l'algorithme). (**\* commentaires \***)



# **FIN CHAP2**

# UNIVERSITE ASSANE SECK ZIGUINCHOR



## Chapitre 3

# Concepts de base de l'algorithmique

Dr Ousmane DIALLO  
odiallo@univ-zig.sn

# PLAN

- **Constantes et variables**
- **Types de données**
- **Opérateur, opérande et expression**
- **Actions sur les variables**
- **Les entrées-sorties**
- **Notre 1er algorithme**
- **Exemples d'application**
- **Corrections des exemples d'application**

# III. Concepts de base de l'algorithmique(1/26)

## III.1. Les constantes et les variables (1/3)

### □ Les constantes:

Représentent des nombres, des chiffres, des caractères, des chaînes de caractères. **Une fois initialisées, leurs valeurs ne peuvent pas être modifiées** tout au long de l'exécution de l'algorithme.

**Mot clé: const**

**Les Constantes sont définies par trois caractéristiques fondamentales:**

- **l'identificateur:** correspond au nom et est composé de lettres et de chiffres
- **le type:** Une constante possède un type déterminé par sa valeur
- **Une valeur:** la valeur de la constante

**const identificateur = valeur;**

# III. Concepts de base de l'algorithmique(2/26)

## III.1. Les constantes et les variables (2/3)

### ☐ Les variables:

Une variable est une entité qui peut contenir une information (nombres, chiffres, caractères, chaînes de caractères, etc.). **Leurs valeurs peuvent être modifiées** tout au long de l'exécution de l'algorithme.

**Mot clé: var**

**Les variables sont définies par trois caractéristiques fondamentales:**

- **l'identificateur:** correspond au nom et est composé de lettres et de chiffres
- **le type:** détermine la nature et les propriétés (opérations réalisables sur ce type)
- **Une valeur:** la valeur de la variable



# III. Concepts de base de l'algorithmique(3/26)

## III.1. Les constantes et les variables (3/3)

### ☐ Les variables:

L'ensemble des variables sont stockées dans la **mémoire de l'ordinateur**

On peut faire l'analogie avec une armoire d'archive qui contiendrait des tiroirs étiquetés :

- l'armoire serait la mémoire de l'ordinateur
- les tiroirs seraient les variables (l'étiquette correspondrait à l'identifiant)
- le contenu d'un tiroir serait la valeur de la variable correspondante
- la couleur du tiroir serait le type de la variable (bleu pour les factures, rouge pour les bons de commande, etc.)

# III. Concepts de base de l'algorithmique(4/26)

## III.2. Les types de données (1/3)

❑ Le type d'une variable caractérise :

- l'ensemble des valeurs que peut prendre la variable
- l'ensemble des actions que l'on peut effectuer sur une variable

❑ Lorsqu'une variable apparaît dans l'entête d'un algorithme on lui associe un type en utilisant la syntaxe suivante

- **var Identifiant de la variable : Son type;**

❑ **Par exemple :**

- Var âge : Entier
- var nom : Chaîne de Caractères

❑ Une fois qu'un type de données est associé à une variable, **cette variable ne peut plus en changer**

❑ Une fois qu'un type de données est associé à une variable le contenu de cette variable doit **obligatoirement être du même type**

# III. Concepts de base de l'algorithmique(5/26)

## III.2. Les types de données (2/3)

- ❑ Par exemple, dans l'exemple précédent on a déclaré âge comme un entier
  - âge dans cet algorithme nous ne pourront pas stocker des réels
  - âge dans cet algorithme ne pourront pas changer de type
- ❑ Il y a deux grandes catégories de type :
  - les types simples
  - les types complexes (que nous verrons dans la suite du cours)

# III. Concepts de base de l'algorithmique(6/26)

## III.2. Les types de données (3/3): Les types simples (types de base)

- ❑ **L'entier (mot clé : entier)** 45, -565, 0 (déc.), 45h, 0FBh (hexadéc.), 1, 101 (binaire)

Il prend ses valeurs dans un sous-ensemble des entiers relatifs. C'est un ensemble fini dans lequel chaque élément possède un successeur et un prédécesseur.

- ❑ **Le réel (mot clé : réel)** -3.67, 4,2569, 18.2 e-6

Il prend ses valeurs dans un sous-ensemble de réels décimaux signés. Dans la plupart des langages, cet ensemble n'est pas un ensemble fini.

- ❑ **Le caractère (mot clé : char)** 'a', '!', '8', '\*', 'B'

Il prend ses valeurs dans l'ensemble des caractères de la table ASCII.

- ❑ **La chaîne de caractères (mot clé : chaîne)** 'bonjour', 'chaîne de caractères'

Ce type se compose d'une suite de symboles de type caractère.

- ❑ **Le type booléen (mot clé : booléen)** VRAI, FAUX

C'est un type logique qui peut prendre que les valeurs VRAI ou FAUX. Il servira pour les expressions et les conditions.

## III. Concepts de base de l'algorithmique(7/26)

### III.3. Opérateur, opérande et expression (1/12)

- ❑ Un **opérateur** est un symbole d'opération qui permet d'agir sur des variables ou de faire des calculs.
- ❑ Une **opérande** est une entité (variable, constante ou expression) utilisée par un opérateur
- ❑ Une **expression** est une combinaison d'opérateur(s) et d'opérande(s), elle est évaluée durant l'exécution de l'algorithme, et possède une valeur (son interprétation) et un type

# III. Concepts de base de l'algorithmique(8/26)

## III.3. Opérateur, opérande et expression (2/12)

□ Par exemple dans  $a+b$  :

- $a$  est l'opérande gauche
- $+$  est l'opérateur
- $b$  est l'opérande droite
- $a+b$  est appelé une expression
- Si par exemple  $a$  vaut 2 et  $b$  3, l'expression  $a+b$  vaut 5
- Si par exemple  $a$  et  $b$  sont des entiers, l'expression  $a+b$  est un entier

# III. Concepts de base de l'algorithmique(9/26)

## III.3. Opérateur, opérande et expression (3/12)

### Opérateur

- Un opérateur peut être unaire ou binaire:
  - Unaire s'il n'admet qu'une seule opérande, par exemple l'opérateur non
  - Binaire s'il admet deux opérandes, par exemple l'opérateur +
- Un opérateur est associé à **un type de donnée** et ne peut être utilisé qu'avec des variables, des constantes, ou des expressions de ce type
  - Par exemple l'opérateur + ne peut être utilisé qu'avec les types arithmétiques (naturel, entier et réel) ou (exclusif) le type chaîne de caractères dans le cas des concaténations.
  - **On ne peut pas additionner un entier et un caractère**
  - Toutefois **exceptionnellement** dans certains cas on accepte d'utiliser un opérateur avec deux opérandes de types différents, c'est par exemple le cas avec les types arithmétiques ( $2+3.5$ )

# III. Concepts de base de l'algorithmique(10/26)

## III.3. Opérateur, opérande et expression (4/12)

### Opérateur

- La signification d'un opérateur peut changer en fonction du type des opérandes
  - Par exemple l'opérateur + avec des entiers aura pour sens l'addition, mais avec des chaînes de caractères aura pour sens la **concaténation**
    - $2+3$  vaut 5
    - "bonjour" + " tout le monde" vaut "bonjour tout le monde"



# III. Concepts de base de l'algorithmique(11/26)

## III.3. Opérateur, opérande et expression (5/12)

### Opérateurs sur les entiers et réels

- ❑ On retrouve tout naturellement  $+$ ,  $-$ ,  $/$ ,  $*$
- ❑ Avec en plus pour les naturels et les entiers **div** et **mod**, qui permettent respectivement de calculer une *division entière* et le *reste de cette division*, par exemple :
  - 11 div 2 vaut 5
  - 11 mod 2 vaut 1

# III. Concepts de base de l'algorithmique(12/26)

## III.3. Opérateur, opérande et expression (6/12)

### Opérateurs sur les énumérés

- Pour les énumérés nous avons trois opérateurs **succ**, **pred**, **ord** :
- **succ** permet d'obtenir le successeur, par exemple avec le type JourDeLaSemaine :
  - succ Lundi vaut Mardi
  - succ Dimanche vaut Lundi
- **pred** permet d'obtenir le prédécesseur, par exemple avec le type JourDeLaSemaine :
  - pred Mardi vaut Lundi
  - pred Lundi vaut Dimanche
- **ord** permet d'obtenir le naturel de l'énuméré spécifié dans la bijection du type énuméré vers les naturels, par exemple avec le type JourDeLaSemaine :
  - ord Lundi vaut 0
  - ord Dimanche vaut 6

# III. Concepts de base de l'algorithmique(13/26)

## III.3. Opérateur, opérande et expression (7/12)

### Opérateurs sur les caractères

□ Pour les caractères on retrouve les trois opérateurs des énumérés avec en plus un quatrième opérateur nommé `car` qui est le dual de l'opérateur `ord` avec comme fonction de bijection la table de correspondance de la norme ASCII

- Par exemple
  - `ord A` vaut 65
  - `car 65` vaut A
  - `pred A` vaut @

# III. Concepts de base de l'algorithmique(14/26)

## III.3. Opérateur, opérande et expression (8/12)

### Opérateurs booléens

- Pour les booléens nous avons les opérateurs non, et, ou, ou Exclusif

Non

a	non a
Vrai	Faux
Faux	Vrai

ET

a	b	a et b
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Faux
Faux	Faux	Faux

OU

a	b	a ou b
Vrai	Vrai	Vrai
Vrai	Faux	Vrai
Faux	Vrai	Vrai
Faux	Faux	Faux

OU Exclusif

a	b	a ou Exclusif b
Vrai	Vrai	Faux
Vrai	Faux	Vrai
Faux	Vrai	Vrai
Faux	Faux	Faux

# III. Concepts de base de l'algorithmique(15/26)

## III.3. Opérateur, opérande et expression (9/12)

### Opérateur d'égalité, d'inégalité, etc.

#### ☐ L'opérateur d'égalité

- C'est l'opérateur que l'on retrouve chez tous les types simples qui permet de savoir si les deux opérandes sont égales
- Cet opérateur est représenté par le caractère =
- Une expression contenant cet opérateur est un booléen

#### ☐ On a aussi l'opérateur d'inégalité !=

#### ☐ Et pour les types possédant un ordre les opérateurs de comparaison <, >, <=, >=

# III. Concepts de base de l'algorithmique(16/26)

## III.3. Opérateur, opérande et expression (10/12)

### Les expressions conditionnelles

- Une expression conditionnelle (ou expression logique, ou expression booléenne ou condition) est une expression dont la valeur est soit VRAI soit FAUX. On peut donc affecter une expression conditionnelle à une variable booléenne. Il existe plusieurs types d'expressions conditionnelles:
  - **Les conditions simples:** une condition simple est une comparaison de deux expressions de même type.
    - Exemples  $a < 0$  ,  $op = 's'$  ,  $(a + b - 3) * c = (5 * y - 2) / 3$
  - **Les conditions complexes:** formées de plusieurs conditions simples ou variables booléennes reliées entre elles par les opérateurs logiques ET, OU, NON.
    - Exemples  $(a < 0)$  **ET**  $(b < 0)$   
 $((a + 3 = b)$  **ET**  $(c < 0)$  ) **OU**  $((a \neq c * 2)$  **ET**  $(b \neq c)$  )

# III. Concepts de base de l'algorithmique(17/26)

## III.3. Opérateur, opérande et expression (11/12)

### Priorité des opérateurs

- ☐ Tout comme en arithmétique les opérateurs ont des priorités
  - Par exemple  $*$  et  $/$  sont prioritaires sur  $+$  et  $-$
- ☐ Pour les booléens, la priorité des opérateurs est non, et, ou Exclusif et ou
- ☐ Pour clarifier les choses (ou pour dans certains cas supprimer toutes ambiguïtés) on peut utiliser des parenthèses

# III. Concepts de base de l'algorithmique(18/26)

## III.3. Opérateur, opérande et expression (12/12)

### Récapitulatif des opérateurs sur les types de base

Opérateur	Notation	Type des opérandes	Type du résultat
+ et – unaires négation logique	+ - NON	entier ou réel booléen	celui de l'opérande booléen
Elévation à la puissance	↑	entier ou réel	entier ou réel
Multiplication	*	entier ou réel	entier ou réel
Division entière	DIV	entier	entier
Division	/	entier ou réel	réel
reste(modulo)	MOD	entier	entier
Addition	+	entier ou réel	entier ou réel
Soustraction	-		
Comparaison	< <= > >= = <>	tout type	booléen
et logique ou logique	ET OU	booléen	booléen



# III. Concepts de base de l'algorithmique(19/26)

## III.4. Actions sur les variables (1/2)

❑ On ne peut faire que deux choses avec une variable :

1. Obtenir son contenu (*regarder le contenu du tiroir*)
  - Cela s'effectue simplement en nommant la variable
2. Affecter un (nouveau) contenu (mettre une (nouvelle) information dans le tiroir)
  - Cela s'effectue en utilisant l'opérateur d'affectation représenté par le symbole **←**

# III. Concepts de base de l'algorithmique(20/26)

## III.4. Action sur les variables (2/2): L'opérateurs d'affectation

### □ L'affectation

Elle permet d'affecter une valeur à une variable. La valeur doit être compatible avec le type de la variable ou de l'objet à gauche de l'affectation

**Identificateur ← expression; ou identificateur := expression;**

X ← 1;                    ( X prend la valeur 1 )

X ← 2\*3+5;            ( X prend la valeur du résultat de l'opération 2\*3+5)

D ← D+1;                ( D augmente de 1, il est **incrémenté**)

D ← D-1;                ( D diminue de 1, il est **décrémenté**)

prix\_total ← nb\_kg \* prix\_du\_kg;

Si nb\_kg est de type entier et prix\_du\_kg est de type réel alors prix\_total doit être de type réel pour recevoir le résultat de l'expression

# III. Concepts de base de l'algorithmique(21/26)

## III.5. Les opérations d'entrées-sorties

- ❑ Un algorithme peut avoir des interactions avec l'utilisateur
- ❑ Il peut afficher un résultat (du texte ou le contenu d'une variable) et demander à l'utilisateur de saisir une information afin de la stocker dans une variable
- ❑ En tant qu'informaticien on raisonne en se mettant à la place de la machine, à l'aide des opérateurs de:
  - De lecture

Pour donner la possibilité à l'utilisateur de saisir (au clavier) une information et de les enregistrer en mémoire (dans la RAM) on utilise la commande **lire** suivie entre parenthèses de la variable de type simple qui va recevoir la valeur saisie par l'utilisateur, par exemple :

**lire (variable1); lire (variable1, variable2, ...);**

- D'affichage

Pour afficher (à l'écran) une information on utilise la commande **écrire** suivie entre parenthèses de la chaîne de caractères entre guillemets et/ou des variables de type simple à afficher séparées par des virgules, par exemple :

**écrire (variable1); écrire (variable1, variable2,...); écrire ('le résultat est', variable); écrire ('Bonjour');**

# III. Concepts de base de l'algorithmique(22/26)

## III.6. Notre premier algrithme

### Algorithme Echange

/\* Déclarations \*/

**variable**

x,y:entier

tmp : **entier**

/\* Corps du programme \*/

**Début**

**Ecrire**('Donner la valeur de l'entier x:')

**Lire**(x)

**Ecrire**('Donner la valeur de l'entier y:')

**Lire**(y)

**Ecrire**('Avant échange: x vaut ',x, 'et y vaut ',y)

tmp ← x

x ← y

y ← tmp

**Ecrire**("Après échange: x vaut ",x, "et y vaut ",y)

**Fin**

## III. Concepts de base de l'algorithmique(23/26)

### III.7. Exemples d'application

1. Écrire un programme calculant la surface d'un cercle à partir de la valeur du rayon, saisie au clavier.
2. Écrire un programme qui demande à l'utilisateur de saisir le prix en francs d'une bouteille d'eau minérale de 1,5l. Dans un deuxième temps, le programme affiche le prix au litre, le prix d'un pack d'eau, sachant qu'il y a 6 bouteilles, dont une est offerte, ainsi que le prix d'une palette de packs d'eau, sachant qu'il y a 50 packs dans une palette dont 5 sont offerts.

# III. Concepts de base de l'algorithmique(24/26)

## III.8. Corrections des exemples d'application (1/3)

Algorithme CalculSurface

Constante

PI=3.1416

Variable

Rayon, Surface : REEL

Début

/\*saisie de la valeur du rayon\*/

Ecrire('Entrer la valeur du rayon')

Lire(Rayon)

/\*Effectuer le calcul de la surface du cercle\*/

Surface ← PI\*Rayon\*Rayon

/\*afficher le résultat\*/

Ecrire('La surface est:', Surface, 'Cm2')

Fin

# III. Concepts de base de l'algorithmique(25/26)

## III.8. Corrections des exemples d'application (2/3)

Algorithme CalculPrix

Variable

PrixBouteille, PrixLitre, PrixPack : REEL

Début

/\*saisir le prix d'une bouteille d'eau\*/

Ecrire('Entrer le prix d'une bouteille d'eau de 1.5L')

Lire(PrixBouteille)

/\*Effectuer le calcul\*/

PrixLitre ←  $(\text{PrixBouteille} * 2) / 3$

PrixPack ←  $\text{PrixBouteille} * 5$

/\*afficher le résultat\*/

Ecrire('Le prix au litre est de:', PrixLitre, 'FCFA')

Ecrire('Le prix d'un pack d'eau est de:', PrixPack, 'FCFA')

Ecrire('Le prix d'une palette est de:', PrixPack\*45, 'FCFA')

Fin

# III. Concepts de base de l'algorithmique(26/26)

## III.8. Corrections des exemples d'application (3/3)

Algorithme CalculPrix

Constante

Nb\_Bouteilles\_Pack=6

Nb\_Bouteilles\_Offertes=1

Nb\_Packs\_Par\_Palette=50

Nb\_Packs\_offerts=5

Nb\_Bouteilles\_Payantes=Nb\_Bouteilles\_Pack – Nb\_Bouteilles\_Offertes

Nb\_Packs\_Payants=Nb\_Packs\_Par\_Palette – Nb\_Packs\_Offerts

Variable

PrixBouteille, PrixLitre, PrixPack : REEL

Début

/\*saisir le prix d'une bouteille d'eau\*/

Ecrire('Entrer le prix d'une bouteille d'eau de 1.5L')

Lire(PrixBouteille)

/\*Effectuer le calcul\*/

PrixLitre ←  $(\text{PrixBouteille} \times 2) / 3$

PrixPack ←  $\text{PrixBouteille} \times \text{Nb\_Bouteilles\_Payantes}$

/\*afficher le résultat\*/

Ecrire('Le prix au litre est de:', PrixLitre, 'FCFA')

Ecrire('Le prix d'un pack d'eau est de:', PrixPack, 'FCFA')

Ecrire('Le prix d'une palette est de:', PrixPack \* Nb\_Packs\_Payants, 'FCFA')

Fin





# **FIN CHAP3**