INFORMATIQUE EN PREMIERES C & D

NOUVEAU PROGRAMME

Edition d'essai 100% Gratuit

COURS SELON L'APC

CLASSE DE 1eres C & D

Cours conçus et rédigés par :

- **ATATCHIM BRICE F.**
- **NZEAKOU ARNAUD**
- **POUKUE YVES E.**
- **ABOSSOUWA CLAUDE-MARIE**
- **NJOFANG HERMAN RAOUL**
- **MEZAH FONTABONG Yannick**
- **FEUWO TACHULA Christian**
- **BASSIROU OUSMANOU**
- **ZADANA SIDONIE**
- **WENDJEL Aaron**
- **NJINKEU JEAN JULES**

Sous la coordination de **NJINKEU JEAN JULES**
professeur des lycées

Une œuvre du groupe

WHATSAPP :

NOUVEAU PROGRAMME INFO

N°	LEÇONS	ENSIGNANTS AYANT CONÇU ET REDIGE LA LECON :
1.	INSTALLATION D'UN SYSTEME D'EXPLOITATION	FEUWO TACHULA CHRISTIAN et NJINKEU JEAN JULES
2.	UTILISATION DE L'INVITE DE COMMANDE	MEZAH FONTABONG YANNICK
3.	APPLICATION DES CONCEPTS FONDAMENTAUX DE LA SECURITE INFORMATIQUE	NJINKEU JEAN JULES et POUKUE MOUNDOU YVES E
4.	UTILISATION DES FICHIERS MULTIMEDIAS	NJINKEU JEAN JULES
5.	DESCRIPTION DES CONCEPTS DE BASE DES SYSTEMES D'INFORMATION	NJOFANG HERMAN RAOUL et ABOSSOUWA CLAUDE-MARIE
6.	DESCRIPTION DES CONCEPTS DES BASES DE DONNES	NJINKEU JEAN JULES
7.	UTILISATION DES STRUCTURES ALGORITHMIQUES	BASSIROU OUSMANOU
8.	UTILISATION DES FONCTIONS ET DES PROCEDURES	ZADANA SIDONIE
9.	PROGRAMMATION EN HTML	WENDJEL AARON
10.	PROGRAMMATION EN JAVASCRIPT	ATATCHIM FOUDJIN BRICE et NZEAKOU WONSO ARNAUD
11.	PROGRAMMATION EN LANGAGE C	NJOFANG HERMAN RAOUL

DECOMPOSITION DES UNITES D'APPRENTISSAGE

CHAPITRE 1: INSTALLATION D'UN SYSTEME D'EXPLOITATION

Lecon1 : Notions de base des systèmes d'exploitation

Lecon2 : Installation d'un système d'exploitation

CHAPITRE 2: UTILISATION DE L'INVITE DE COMMANDE

Lecon1 : Présentation de l'invite de commande

Lecon2 : Gestion des fichiers et des répertoires

CHAPITRE 3: APPLICATION DES CONCEPTS FONDAMENTAUX DE LA SECURITE INFORMATIQUE

Lecon1 : Application des concepts fondamentaux de la sécurité informatique

CHAPITRE 4: UTILISATION DES FICHIERS MULTIMEDIAS

Lecon1 : Utilisation des fichiers multimédias

CHAPITRE 5: DESCRIPTION DES CONCEPTS DE BASE DES SYSTEMES D'INFORMATION

Lecon1 : Notions de base des si

Lecon2 : Composantes et fonctions d'un si

Lecon3 : Méthodes de conception d'un si

CHAPITRE 6: DESCRIPTION DES CONCEPTS DES BASES DE DONNES

Lecon1 : Description des concepts des Bases de Données

CHAPITRE 7: UTILISATION DES STRUCTURES ALGORITHMIQUES

Lecon1 : Notions d'algorithme

Lecon2 : Structures algorithmiques

Lecon3 : Exécuter un algorithme

CHAPITRE 8: UTILISATION DES FONCTIONS ET DES PROCEDURES

Lecon1 : Utilisation des fonctions et des procédures

CHAPITRE 9: PROGRAMMATION EN HTML

Lecon1 : Programmation en HTML

CHAPITRE 10: PROGRAMMATION EN JAVASCRIPT

Lecon1 : Initiation au JavaScript

Lecon2 : Structures algorithmiques

Lecon3 : Exécuter un algorithme

CHAPITRE 11: PROGRAMMATION EN LANGAGE C

Lecon1 : Introduction au langage C

Lecon2 : Les structures de contrôle

Lecon3 : Les tableaux en langage c

Lecon4 : Le sous programme

Lecon5 : Exécution d'un programme C

MODULE1 : ENVIRONNEMENT NUMERIQUE, SECURITE, MULTIMEDIA ET GESTION DES DONNEES

CHAPITRE 1: INSTALATION D'UN SYSTEME D'EXPLOITATION

LECON 1 : NOTION DE BASE DES SYSTEMES D'EXPLOITATION

Situation problème

Votre grand Frère vient de vous offrir un ordinateur de marque Compaq , processeur 2GHZ dual core RAM 2G.O, HDD 500GB , DVD graveur , écran 17'', carte graphique 512 MB pour vous faciliter la tâche concernant vos études scolaire. Mais cependant vous remarquez que lors de son démarrage un message d'erreur s'affiche à l'écran : '**Opérating System not found**''. Expliquer ce qui c'est passe.

Consigne :

- Que veut dire operating system
- Pourquoi sans lui l'ordinateur ne peut démarrer ?
- Quel est son rôle ?

Compétences visées :

- ✓ Définir système d'exploitation et donner son rôle dans un ordinateur
- ✓ Décrire les composantes d'un Système d'exploitation

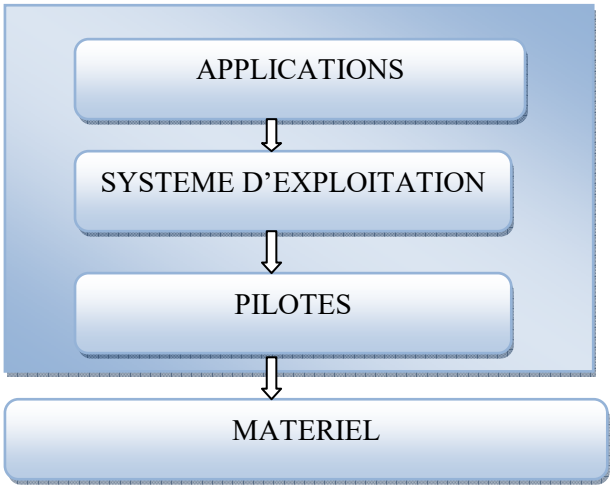
INTRODUCTION

Le system d'exploitation est un programme particulier qui permet de coordonner le bon fonctionnement d'un ordinateur c'est à dire qu'il assure la gestion de la mémoire, du système de fichiers, de l'ordonnancement des taches, et les périphériques. Dans cette partie du programme nous verrons les outils nécessaires pour installer, restaurer un système d'exploitation.

I. DESCRIPTION DU SYSTEME D'EXPLOITATION

Le **système d'exploitation** (noté *SE* ou *OS*, abréviation du terme anglais *Operating System*), est chargé d'assurer la liaison entre les ressources matérielles, l'utilisateur et les applications (traitement de texte, jeu vidéo, ...). Ainsi lorsqu'un programme désire accéder à une ressource matérielle, il ne lui est pas nécessaire d'envoyer des informations spécifiques au périphérique, il lui suffit d'envoyer les informations au système d'exploitation, qui se charge de les transmettre au périphérique concerné via son pilote. En l'absence de pilotes il faudrait que

chaque programme reconnaisse et prenne en compte la communication avec chaque type de périphérique !



Le système d'exploitation permet ainsi de "dissocier" les programmes et le matériel, afin notamment de simplifier la gestion des ressources et offrir à l'utilisateur une interface homme-machine (notée «IHM») simplifiée afin de lui permettre de s'affranchir de la complexité du matériel (de la machine physique).

II. ROLES DU SYSTEME D'EXPLOITATION

Les rôles du système d'exploitation sont divers :

- **Gestion du processeur** : le système d'exploitation est chargé de gérer l'allocation du processeur entre les différents programmes grâce à un **algorithme d'ordonnancement**. Le type d'ordonnanceur est totalement dépendant du système d'exploitation, en fonction de l'objectif visé.
- **Gestion de la mémoire vive** : le système d'exploitation est chargé de gérer l'espace mémoire alloué à chaque application.
- **Gestion des entrées/sorties** : le système d'exploitation permet d'unifier et de contrôler l'accès des programmes aux ressources matérielles par l'intermédiaire des pilotes (appelés également gestionnaires de périphériques ou gestionnaires d'entrée/sortie).
- **Gestion de l'exécution des applications** : le système d'exploitation est chargé de la bonne exécution des applications en leur affectant les ressources nécessaires à leur bon fonctionnement. Il permet à ce titre de «tuer» une application ne répondant plus correctement.
- **Gestion des droits** : le système d'exploitation est chargé de la sécurité liée à l'exécution des programmes en garantissant que les ressources ne sont utilisées que par les programmes et utilisateurs possédant les droits adéquats.
- **Gestion des fichiers** : le système d'exploitation gère la lecture et l'écriture dans le système de fichiers et les droits d'accès aux fichiers par les utilisateurs et les applications.
- **Gestion des informations** : le système d'exploitation fournit un certain nombre d'indicateurs permettant de diagnostiquer le bon fonctionnement de la machine.

III. COMPOSANTES DU SYSTEME D'EXPLOITATION

Le système d'exploitation est composé d'un ensemble de logiciels permettant de gérer les interactions avec le matériel. Parmi cet ensemble de logiciels on distingue généralement les éléments suivants :

- Le **noyau** (en anglais **kernel**) représentant les fonctions fondamentales du système d'exploitation telles que la gestion de la mémoire, des processus, des fichiers, des entrées-sorties principales, et des fonctiwqqwqqqqonnalités de communication.
- L'**interpréteur de commande** (en anglais **shell**, traduisez «*coquille*» par opposition au noyau) permettant la communication avec le système d'exploitation par l'intermédiaire d'un langage de commandes, afin de permettre à l'utilisateur de piloter les périphériques en ignorant tout des caractéristiques du matériel qu'il utilise, de la gestion des adresses physiques, etc.
- Le **système de fichiers** (en anglais «*file system*», noté *FS*), permettant d'enregistrer les fichiers dans une arborescence.

IV. EXEMPLES DE SYSTEMES D'EXPLOITATION

Il existe plusieurs types et catégories et familles de systèmes d'exploitation :

- Le système MS DOS (Microsoft Disc Opérating System)
- Les Système d'exploitation de la famille WINDOWS (WindowsXP, Windows vista, Windows 7, Windows 8, Windows 10, etc…) tous sont des systèmes multitaches et multiutilisateurs.
- Les Système d'exploitation de la famille LINUX(Debian, Ubuntu, Fedora, Mandriva, Knopix, etc..) tous sont des systèmes multitaches et multiutilisateurs.
- Le système MAC OS De Macintosh Multitache multiutilisateur

CONCLUSION

Le système d'exploitation est le programme de base nécessaire pour l'utilisation des applications et des périphériques de l'ordinateur. Des distributions de Systèmes existantes, celles de la maison Windows sont propriétaires et sont payantes ainsi donc lors du téléchargement sur le site officiel elle vous donne la possibilité d'acheter la clé d'activation (code compose de chiffre et de lettre permettant de prouver que la version Windows que vous utilisez est authentique).

Exercices :

1- Définir système d'exploitation,
2- Donner trois rôles qu'assure un système d'exploitation dans un ordinateur
3- Quelle différence faites-vous entre un système propriétaire et un système libre.
4- Tous les systèmes d'exploitation libres sont-ils gratuits ? Tout système propriétaire est il payant ?

LECON 2: INSTALLATION D'UN SYSTEME D'EXPLOITATION

Situation problème

Votre grand Frère vient de vous offrir un ordinateur de marque Compaq , processeur 2GHZ dual core RAM 2G.O, HDD 500GB , DVD graveur , écran 17'', carte graphique 512 MB pour vous faciliter la tâche concernant vos études scolaire. Mais cependant vous remarquez que lors de son démarrage un message d'erreur s'affiche à l'écran : **''system error''**. Expliquer ce qui c'est passe.

Consigne :

- Que faire pour résoudre ce problème ?
- Quel sont les outils nécessaires pour l'installation d'un SE ?
- Quels sont les opérations que l'on peut effectuer pendant le processus d'installation ?

Compétences visées :

- ✓ Déterminer les besoins matériels et logiciels pour l'installation d'un Système d'exploitation
- ✓ Configurer l'ordre de boot
- ✓ Créer un point de restauration
- ✓ Partitionner le disque dur
- ✓ Installer un système d'exploitation

INTRODUCTION

L'installation d'un system d'exploitation nécessite d'avoir une connaissance sur les caractéristiques de l'ordinateur, le BIOS, la notion de partition, et de formatage d'un disque dur. Les exemples de ce cours serons faits avec le système windows7.

1- Le BIOS (Basic Input Output System)

C'est un ensemble de micro logiciel contenu dans la ROM permettant d'effectuer des opérations de base lors de la mise sous tension d'un ordinateur. Comme vérifier la connectivité des périphériques, détecte les panne de l'ordinateur et planifier l'ordre de démarrage des composantes (disque dur, clés USB, lecteur cd-rom etc..) . Pour accéder à la configuration de ce dernier il nous suffit d'appuyer sur la touche F2 ou F10 souvent la touche ESC.

2- Le partitionnement d'un disque dur

Partitionner un disque dur c'est le fait de subdiviser ce dernier de façon logique en plusieurs parties. Cette opération offre comme avantage la possibilité d'installer plusieurs systèmes d'exploitation dans une même machine, une meilleure organisation de ses données. Après partitionnement, on distingue une partition primaire ou principale et plusieurs partitions logiques.

3- Le formatage d'un disque dur

C'est le fait de préparer un disque dur en y inscrivant un système de fichier (FAT File Allocation Table, NFTS New Technology File System, HFS, EXT FS, etc..) de façons a ce qui soit reconnu par le SE de l'ordinateur. On distingue deux types de formatage à savoir :

- Le formatage de bas niveau (physique) qui se fait par le fabriquant
- Le formatage de haut niveau (logique) qui se par fait par l'utilisateur

4- Processus d'installation d'un se dans un ordinateur

Pour installer un SE dans un ordinateur il nous suffit de :

1- Choisir la distribution du SE en fonction des caractéristiques du pc

2- Le mettre sur un support (cd,dvd ou usb)

3- Démarrer ce pc avec le dit support de stockage

4- Appuyer sur F9 pour choisir sur quel périphérique démarrer voir figure 2 (en annexe)

5- Choisissez le type d'installation voir figure 3 (en annexe)

a- Mise à niveau de windows a une version plus recente.

b- Personnaliser l'installation c'est l'installer sans tenir compte que l'ordinateur possède une version antérieure.

6- Choisir la partition ou vous voulez installer le SE voir figure 4 (en annexe)

7- Si vous voulez partitionner ou formater le disque cliquer sur drive option voir figure 5 (en annexe)

a- Cliquez sur Delete (Supprimer) pour supprimer la partition existante.

b- Cliquez sur New (Nouveau) pour créer la partition.

c- Modifiez les paramètres de taille de la partition, puis cliquez sur Apply (Appliquer).

d- Cliquez sur Suivant.

8- L'installation démarre et votre pc redémarrera plusieurs fois

9- Lorsque l'installation de Windows est terminée, Windows démarre et vous invite à modifier le mot de passe utilisateur. Cliquez sur OK et commencez à configurer le compte utilisateur initial.

5- Installation d'un pilote

Un pilote ou driver est un petit programme permettant au système d'exploitation de communiquer avec un périphérique. Dans ce cas de figure il est recommandé de faire les mise a jour des driver et pour le faire il faut :

- Allez dans panneau de configuration voir figure 6 (en annexe)
- Allez dans gestion de périphérique voir figure 7 (en annexe)
- Localiser le périphérique

- Faire un clic droit puis cliquez sur mise à jour et assurer vous que vous êtes connecte sur internet.

6- Le point de restauration

La restauration d'un système d'exploitation est une opération qui consiste à retourner le système à un état antérieur de bon fonctionnement. Pour restaurer un système, il faut avoir créé au préalable un point de restauration. Un point de restauration C'est une image de votre system à une date donnée. C'est-à-dire que toute modification faite après la création de ce dernier sera tout simplement ignorée lors de la restauration de ce dernier. Il est très utile en cas d'atteinte grave à l'intégrité du SE.

Pour créer un point de restauration il faut :

- Faire un clic droit sur ordinateur ou poste de travail
- Clique sur propriété voir figure 8 (en annexe)
- Cliquez sur protection du system voir figure 9 (en annexe)
- Cliquer sur créer
- Vous donnez un nom puis vous confirmer

Pour restaurer le système il faut :

- Aller dans la protection du system
- Cliquez sur restaurer puis suivez la procédure et votre SE redémarrera la fin.

CONCLUSION

Le Système d'exploitation ne contient pas toujours tous les pilotes (programme servant d'interface entre le périphérique et le SE) Ça peut arriver que certain ne sont pas disponible. Donc parfois après l'installation du SE il peut etre possible de télécharger et d'installer certains pilotes.

Exercices

1- Quelle différence faites vous entre un système 32bits et un Système 64bits
2- Peut-on formater un disque et revoir après cette opération ses données ?
3- Citer deux étapes rencontrées pendant d'installation d'un système d'exploitation.
4- Nono Boris insère son DVD d'installation

Planche

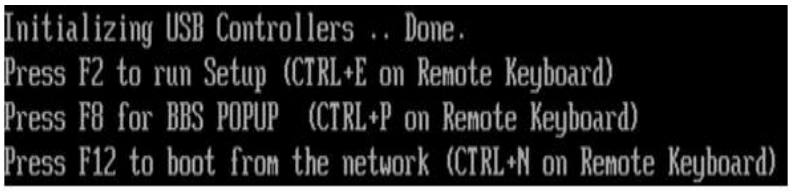


Figure 1

Figure 2 boot menu

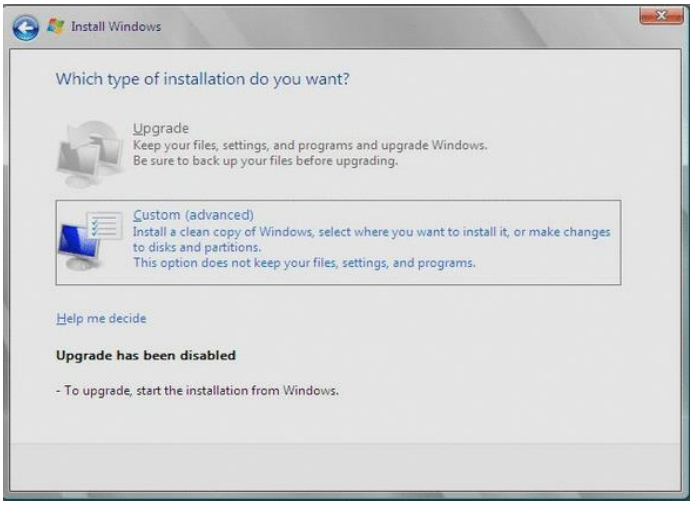


Figure 3 type d'installation

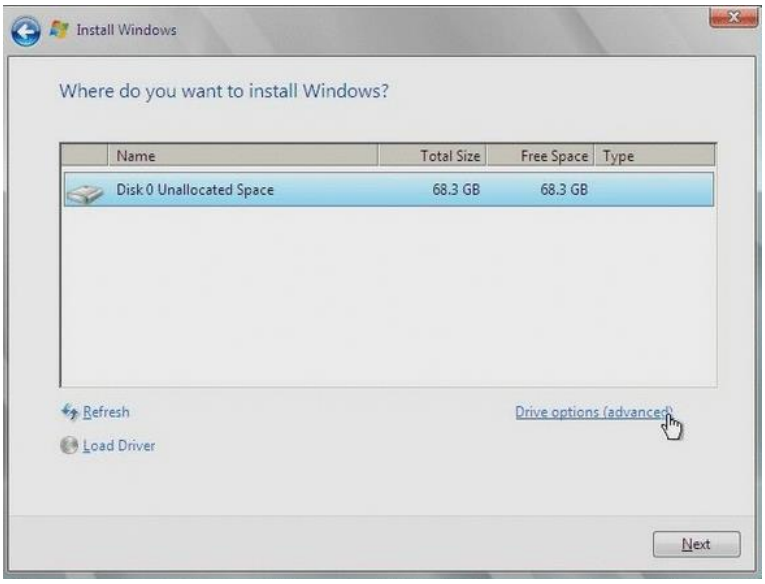


Figure 4 choix de la partition

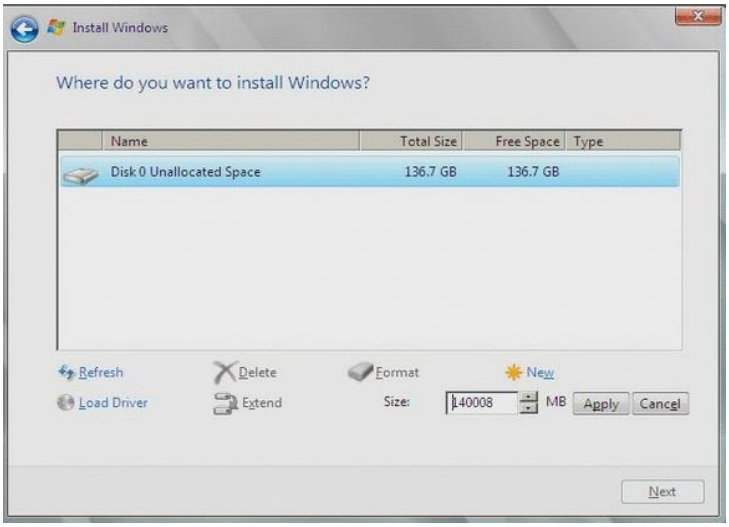


Figure 5 configuration du disque dur

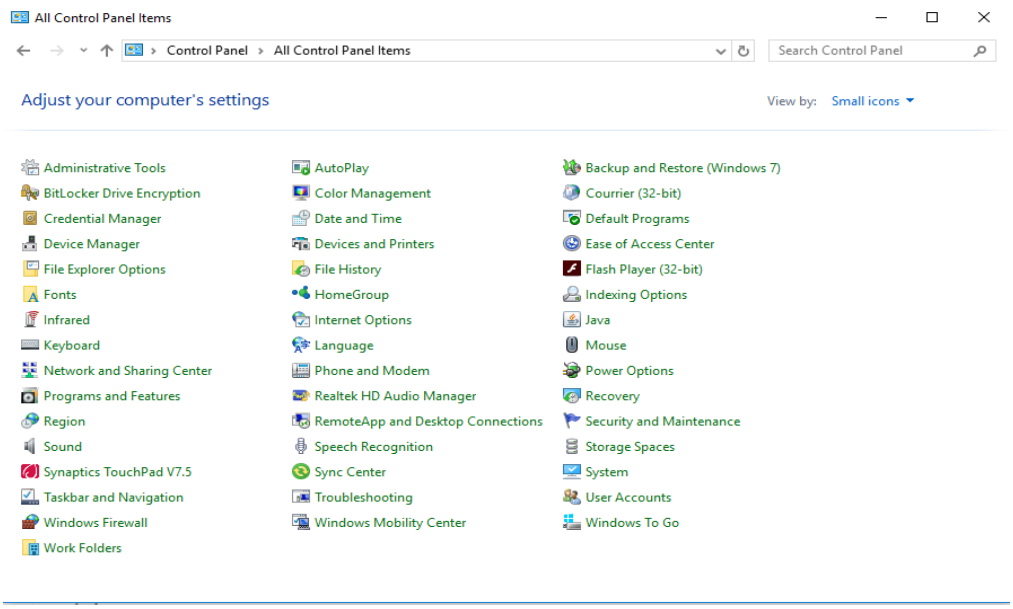


Figure 6 panneau de configuration windows

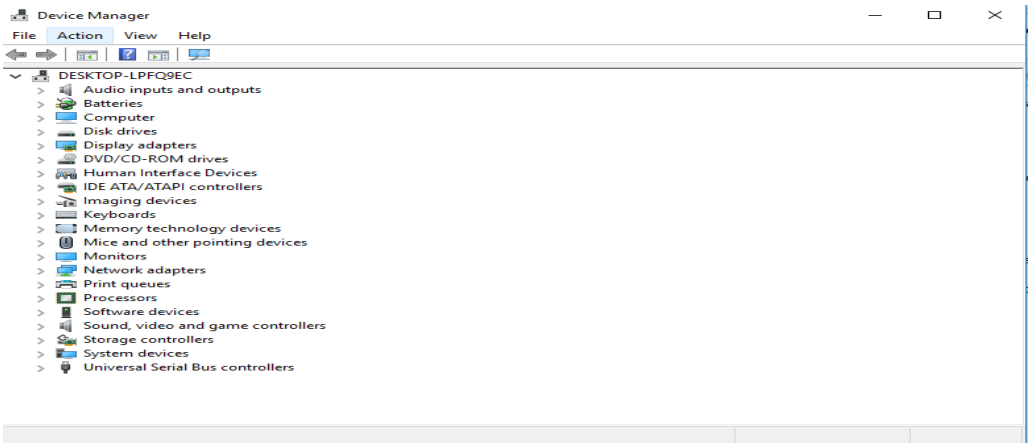


Figure 7 gestionnaire de périphérique

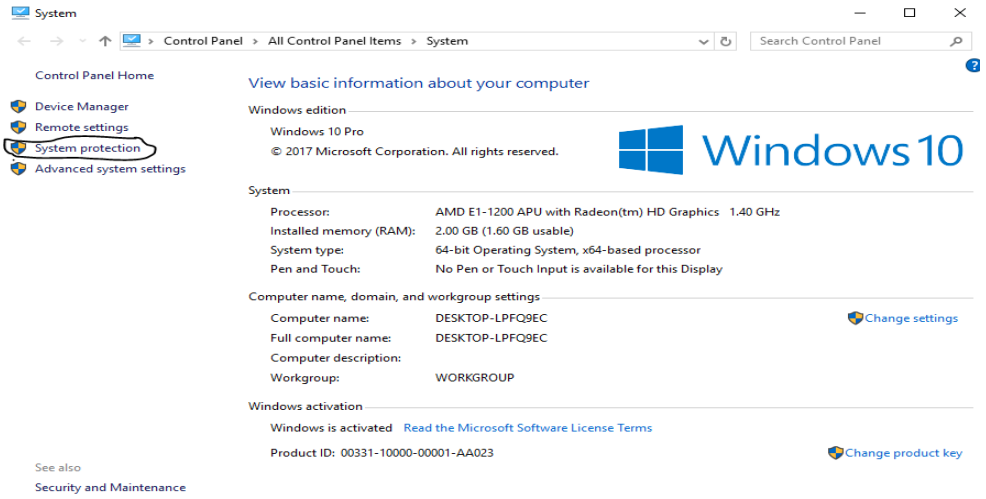


Figure 8

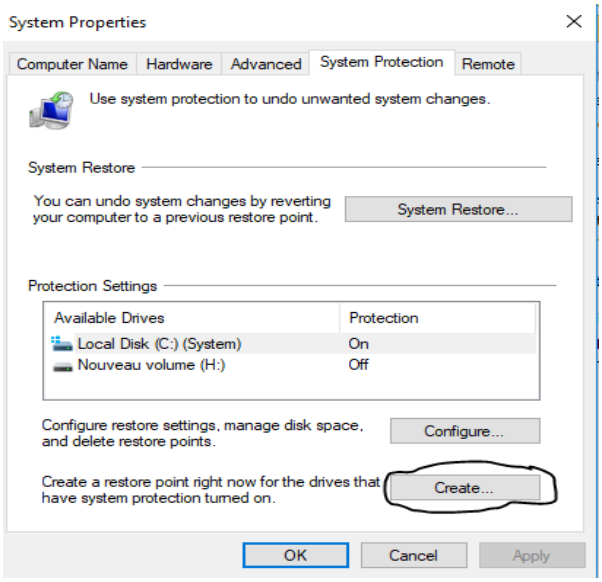


Figure 9

MODULE1 : ENVIRONNEMENT NUMERIQUE, SECURITE, MULTIMEDIA ET GESTION DES DONNEES

CHAPITRE2 : UTILISER L'INVITE DE COMMANDES

Compétences visées :

- ***Donner l'importance de l'utilisation de la ligne de commande***
- ***Accéder à la ligne de commande.***
- ***Utiliser les commandes de manipulation des répertoires et des fichiers.***

Situation problème :

LECON1 : PRESENTATION DE L'INVITE DE COMMANDE

INTRODUCTION

MS-DOS (**M**icro**s**oft **D**isk **O**perating **S**ystem). C'est un système mono-utilisateur, mono-tâche, très peu sécurisé. Fonctionne uniquement sur les processeurs compatibles Intel. MS-DOS a vu le jour en 1981 pour accompagner la sortie de l'IBM PC. Il a donc été fait en collaboration entre IBM et Microsoft.

I. IMPORTANCE DE L'UTILISATION DES LIGNES DE COMMANDES

De nos jours, avec la prolifération des systèmes d'exploitation avec interface graphique (MacOS, Windows, Linux + KDe, Linux + Gnome), il semble inutile et difficile d'utiliser les lignes de commande. Elles sont pourtant indispensables dans plusieurs cas :

- Lorsqu'on utilise un serveur Linux qui n'a pas d'interface graphique.
- Lorsqu'on se connecte à un ordinateur en prise de contrôle à distance par Telnet ou SSH.
- Lorsqu'on souhaite utiliser un logiciel en ligne de commande, par exemple MySQL qui dispose de nombreux outils en lignes de commande, pas tous interfacés dans PHPMyAdmin.
- Lorsqu'on souhaite automatiser des tâches d'administration : création des utilisateurs, planification de sauvegardes ou de tâches diverses...

II. ACCEDER A L'INVITE DE COMMANDE

On peut accéder à l'invite de commande de 03 façons différentes :

- ✓ Menu démarrer → Accessoire → Invite de commande
- ✓ Démarrer → Exécuter → taper « **cmd** »
- ✓ Windows + R → taper « **cmd** »

III. LES UNITES DE DISQUES

Les unités de disques sont des dispositifs de stockage des informations traitées par l'ordinateur tels que le disque dur, le(s) lecteur(s) de disquette, le lecteur de CD-ROM…

Chaque unité disque est désignée par une lettre. Les différentes désignations selon les lettres de l'alphabet sont :

A : pour désigner le lecteur de disquettes des micro-ordinateurs équipés d'un seul lecteur ;
B : pour désigner le second lecteur de disquettes des micro-ordinateurs ;
C : pour désigner le disque dur ;
D, E, F, ... : pour désigner soit le lecteur de CD-ROM et les partitions du disque dur.

Généralement après l'allumage de l'invite de commande, l'unité en cours est celle du disque dur "C:\>". Pour passer d'une unité de disque à une autre, il faut préciser la lettre de l'unité ciblée suivi de « **:** » (deux points superposés).

NB *: Le curseur indique l'endroit où taper la commande, il est représenté par un trait de soulignement clignotant.*

Exemple : Pour passer du disque dur au lecteur de disquettes, tapez : C:\>**A:**
L'invite devient alors **"A:\>"** et toutes les opérations pourront être effectuées sur la disquette.

IV. LES COMMANDES MS-DOS

1. Structure d'une commande

Les commandes MS-DOS peuvent comprendre jusqu'à trois parties ; le nom, les paramètres et les commutateurs. Elle se présente généralement de la manière suivante :

Nomcommande paramètres commutateurs

i. Le nom

Toute commande a un nom qui précise l'action que vous voulez que MS-DOS exécute.

Exemple :

- ✓ Le nom **CLS** (CLear Screen) désigne la commande qui efface l'écran
- ✓ Le nom **VER** (VERsion) affiche la version de MS-DOS
- ✓ Le nom **DATE** affiche la date du système et **TIME** affiche l'heure du système

ii. Les paramètres

Ils précisent l'objet de l'action d'une commande MS-DOS et se trouvent placés après le nom de la commande.

Exemples : Supposons que l'on veuille supprimer un fichier dont le nom est « Premiere.txt ». Vous devrez donc taper la commande DEL (DELete : effacer) comme suit :

C:\>**DEL Premiere.txt**

NB *: Certaines commandes peuvent même utiliser plusieurs paramètres.*

iii. Les commutateurs

Ils servent à modifier l'action d'une commande. Ils sont formés d'un **slash** (/) suivie soit d'une lettre, soit d'un chiffre, le tout placé après le nom de cette commande.

Exemple : La commande DIR (DIRectory) possède des commutateurs. Par exemple, le commutateur /P (DIR/P) permet d'afficher la liste des fichiers d'un répertoire écran par écran et le commutateur /W (DIR/W) permet de l'afficher sur toute la largeur de l'écran. Cette commande peut même s'utiliser avec plusieurs commutateurs tels que "DIR/P/S".

2. Notes générales relatives aux commandes

- ✓ Une commande s'écrit lors de l'apparition de l'invite et du curseur.

- ✓ La commande peut s'écrire en majuscules, minuscules ou les deux mélangées.
- ✓ Une commande est incorrecte si :
  - ➢ Il s'agit d'une mauvaise version de MS-DOS.
  - ➢ La syntaxe est incorrecte.
  - ➢ Ses commutateurs et paramètres ne sont pas valides.
  - ➢ La commande, le paramètre ou le commutateur n'existe pas.
- ✓ Pour annuler ou interrompre une commande lancée, appuyez en même temps sur les touches **CTRL** et. **C.**

V. COMMANDE D'AIDE

1. Aide rapide sur une commande

Pour obtenir de l'aide rapide sur une commande, tapez le nom de celle-ci suivi d'un slash **(/)** et d'un point d'interrogation **(?)** : C:\> ***[NOMCOMMANDE] /?***

Exemple : C:\>DIR /?

2. Aide plus détaillée

Pour obtenir une aide plus détaillée sur toutes les commandes MS-DOS, tapez uniquement : C:\>**HELP** ou C:\>**HELP [NOM COMMANDE]** pour une commande bien précise.

Exemple : Pour obtenir de l'aide sur la commande DIR, tapez : C:\>**HELP DIR**

Exercice de consolidation

1. Lancer l'invite de commande
2. A l'aide commande :
   i. Afficher la date du système
   ii. Afficher l'heure du système
   iii. Changer de disque **C :** en **D :**

LECON2 : GESTION DES FICHIERS ET DES REPERTOIRES

INTRODUCTION

Le fichier constitue l'unité de base de stockage de votre ordinateur. Il permet à MS-DOS de distinguer différents groupes de données. Par exemple, si vous écrivez une lettre en utilisant un programme de traitement de texte, elle sera stockée dans un fichier qui lui est propre. Tout fichier porte un nom qui indique le type de données qu'il contient.

I. OPERATIONS SUR LES REPERTOIRES

Les commandes les plus importantes relatives aux répertoires sont les suivantes : MD (Make Directory), PATH, CD (Current Directory), JOIN, TREE, DELTREE (Delete tree), SUBST, CHDIR, MOVE, XCOPY.

1. Accéder aux répertoires

Pour accéder à un répertoire ou à un sous-répertoire, il suffit de taper la commande "**CD**" ou "**CHDIR**" suivie du nom du répertoire, ou du sous-répertoire, cible.

Syntaxe : C:\>**CD[Nom du répertoire cible]**

L'invite devient : **C:\ [Nom du répertoire courant]>**

Si l'on veut accéder à des sous-répertoires, il faut soit accéder au répertoire père jusqu'aux répertoires fils, pas à pas, en exécutant la commande CD avec la syntaxe précédente, soit accéder aux répertoires fils directement en tapant tous les répertoires (répertoire père, les sous-répertoires intermédiaires et le sous-répertoire cible) sur une seule ligne de commande de la manière suivante :

C :\>CD\[Nom répertoire père]\[Nom sous-répertoire]\.....\[Nom sous-répertoire cible]

Pour passer d'un répertoire courant au répertoire parent, il faut :

C:\[Nom de répertoire courant]>CD..

Si l'on se trouve dans des sous-répertoires et que l'on veuille en sortir pour accéder directement au répertoire racine du disque dur ou de la disquette, il suffit de taper la commande suivante : **C:\[Répertoire/sous-répertoire courant]>CD**

Pour passer d'un répertoire à un autre sans passer par le répertoire racine, tapez la commande de la manière suivante :

C:\[Nom de répertoire courant]>CD\[Nom de répertoire cible]

2. Afficher le contenu des répertoires et leur arborescence ?

Pour afficher le contenu d'un répertoire, il faut se servir de la commande "**DIR**" et de ses commutateurs. Par contre, pour visualiser l'arborescence d'un répertoire, il faut utiliser la commande "**TREE**" et ses commutateurs.

i. La commande DIR

Pour afficher le contenu du répertoire racine d'un disque dur, tapez la commande comme suit : C:\>**DIR**

Le commutateur /P permet d'afficher la liste des fichiers et répertoires écran par écran en appuyant à chaque fois sur n'importe quelle touche du clavier. C:\>**DIR/P**

Le commutateur /W permet, quant à lui, d'afficher sur toute la largeur de l'écran en cinq colonnes. C:\>**DIR/W**

ii. La commande TREE

Pour obtenir l'arborescence d'un répertoire, utilisez la commande **TREE**. L'arborescence du répertoire racine du disque dur s'effectue en tapant : C:\>**TREE**
Et pour l'afficher écran par écran, tapez la commande comme suit : C:\>**TREE |MORE**

3. Créer un répertoire ?

La création de répertoires s'effectue à l'aide de la commande "***MD***" ou "***MKDIR***" avec la syntaxe suivante : C:\>**MD [Nom de répertoire]**

La création d'un sous-répertoire s'effectue de la même manière que pour un répertoire, il suffit de rendre l'invite dans le répertoire courant où l'on veut introduire un sous-répertoire.

Exemples : La création du répertoire MUSICAL sur le disque dur (comme sur une disquette) s'effectue de la manière suivante : C:\>**MD MUSICAL**

Pour créer les sous-répertoires, par exemple, CLASSIC et JAZZ, dans le répertoire MUSICAL, il faut d'abord y accéder en tapant : C:\>**CD MUSICAL**
Ensuite, créez les sous-répertoires un à un comme suit :
C:\MUSICAL>MD CLASSIC Puis **C:\MUSICAL>MD JAZZ**

Pour accéder à l'un de ces sous-répertoires, il suffira, à partir du répertoire courant MUSICAL, de taper la commande CD tel que vu précédemment.

4. Supprimer ou effacer un répertoire ?

Un répertoire ne peut être supprimé qu'à partir du répertoire père directement ascendant. La commande **RD** ne peut supprimer un répertoire que si ses répertoires fils et tous ses fichiers sont eux-mêmes supprimés au préalable. **Syntaxe :** C:\>**RD [Nom de répertoire]**

La commande **DELTREE**, quant à elle, permet d'effacer les répertoires, les sous-répertoires et tous les fichiers. C'est pour cela qu'elle est suivie, après son exécution, d'un message de confirmation d'effacement. La commande **DELTREE** peut également permettre d'effacer tous les répertoires se trouvant sur une unité disque, il suffit de taper :

C:\>**DELTREE *.***

Exemple : supposons que l'on veuille supprimer un sous-répertoire de MUSICAL, par exemple, CLASSIC. Il faut, bien entendu, se trouver dans le répertoire courant MUSICAL. Ensuite, tapez la commande comme suit : C:\MUSICAL>**DELTREE CLASSIC**

Avec la commande RD, il vous faudra accéder au sous-répertoire CLASSIC et effacer tous ses fichiers. Vous devrez ensuite revenir au répertoire père, MUSICAL, pour supprimer le sous-répertoire CLASSIC : C:\MUSICAL>**RD CLASSIC**

5. Renommer un répertoire ?

Un répertoire ne peut être renommé qu'à partir du répertoire père.

Pour renommer un répertoire ou un sous-répertoire, il faut utiliser la commande **MOVE** suivie du nom du répertoire qu'on veut changer et du nouveau nom désiré. La syntaxe est la suivante : C:\>**MOVE [Ancien Nom de répertoire] [Nouveau Nom de répertoire]**

Exemple : Pour renommer le répertoire MUSICAL en MUSIQUE, tapez la commande, à partir du répertoire racine, comme suit : C:\>**MOVE MUSICAL MUSIQUE**

Il existe toutefois une autre manière de renommer un répertoire. Il faut utiliser, après la commande MOVE, l'unité disque et la destination de l'ancien nom suivies de l'unité disque et la destination du nouveau nom de répertoire, en respectant la concordance.

Exemple : Pour renommer le répertoire CLASSIC du répertoire principal MUSICAL en CLASIQUE, et l'invite en cours étant celle du répertoire racine, tapez :

C:\>**MOVE C:\MUSICAL\CLASSIC C:\MUSICAL\CLASIQUE**

II. OPERATIONS SUR LES FICHIERS

1. Les jokers

Les jokers ne sont en fait que des caractères spéciaux qui permettent de simplifier certaines commandes relatives aux fichiers telles que DEL, DIR, COPY, REN, etc. Les deux jokers disponibles, dans MS-DOS, sont l'astérisque (*) et le point d'interrogation (?)

- ✓ L'astérisque permet de sélectionner des fichiers d'après leur nom ou extension.

[Nom de fichier].* : les fichiers qui ont le nom [Nom de fichier] quelle que soit l'extension.
***.[Extension]** : les fichiers dont l'extension est [Extension].
. : tous les fichiers quelque soient le nom et l'extension)

- ✓ Le point d'interrogation permet de sélectionner des fichiers d'après une partie de leur nom en substituant les lettres manquantes par autant de symboles (?).

CONT???.* noms de fichiers dont la longueur est de sept caractères et commençant par CONT

2. Les différentes commandes sur les fichiers

Les commandes relatives aux fichiers sont : ATTRIB, COMP, COPY, DEL, DIR, ERASE, EXE2BIN, EXPAND, FC, FIND, MOVE, MSBACKUP, PATH, PRINT, REN, RENAME, REPLACE, RESTORE, SYS, TYPE, XCOPY.

N.B : Il n'existe pas de commande permettant de créer des fichiers sous MS-DOS. Mais MS-DOS possède un éditeur de textes qui permet la création de fichiers contenant des textes. La création de fichiers ne se fait qu'à l'aide de logiciels (Traitement de texte, langage de programmation, etc.).

3. Comment afficher les fichiers des unités disques ?

Pour afficher les fichiers contenus dans les unités disques, il faut se servir de la commande **DIR** et de ses commutateurs.

Syntaxe : C:\>**DIR[commutateur] [Unité disque]:**

Exemple : Pour afficher le contenu en fichiers du répertoire racine d'une disquette ainsi que de ses répertoires pères, s'ils existent, placez la disquette cible dans le lecteur et tapez : C:\>**DIR A:** Ou sous l'invite A:, tapez : C:\>**A:** ensuite : A:\>**DIR**
Ou, si la disquette contient beaucoup de fichiers : A:\>**DIR/P**

4. Copies de fichiers

La copie de fichiers signifie la duplication des fichiers sélectionnés dans un autre emplacement. Toute copie de fichiers s'effectue avec la commande **COPY**.

i. *Copier un fichier*

Pour copier un fichier, il suffit de préciser son nom, son extension, et la destination, après le nom de la commande COPY.

Syntaxe : C:\>**COPY[Nom.extension] [Unité disque]:\ [Destination]**

Exemple1 : Pour copier le fichier MOZART.TXT, qui se trouve dans le sous-répertoire Classic de Musical du disque dur, vers un répertoire père "Musique" de la disquette, tapez :

C:\MUSICAL\CLASSIC>**COPY MOZART.TXT A:\MUSIQUE**

Exemple2 : Pour copier le même fichier, MOZART.TXT vers le répertoire racine du disque dur, tapez : C:\MUSICAL\CLASSIC>**COPY MOZART.TXT C:**

ii. *Copier un groupe de fichiers ou tous les fichiers d'un répertoire*

Pour effectuer la copie de tous les fichiers ou d'un groupe de fichiers d'un répertoire, il faut utiliser les jokers de MS-DOS avec la commande COPY suivant sa syntaxe.

Exemple1 : Pour copier tous les fichiers du répertoire DOS vers le répertoire racine du disque dur, tapez : C:\DOS>**COPY *.* C:**

Exemple2 : Pour copier les fichiers à extension .EXE du répertoire DOS vers le répertoire racine d'une disquette, tapez : C:\DOS>**COPY *.EXE A:**

Exemple3 : Pour copier tous les fichiers du répertoire DOS dont le nom commence par "MS" vers le répertoire racine du disque dur, tapez : C:\DOS>**COPY MS*.* C:**

5. Les déplacements de fichiers

Le déplacement de fichiers dans une même unité disque ou vers d'autres, s'effectue à l'aide de la commande **MOVE**.

Pour effectuer le déplacement d'un ou plusieurs fichiers, il faut que l'invite en cours (unité disque et répertoire courant) soit celle qui contient les fichiers spécifiés.

i. *Déplacer un fichier*

Il suffit de préciser après la commande MOVE, le nom du fichier à déplacer et son extension si elle existe.

Syntaxe : C:\>**MOVE [Nom.Extension] [Unité disque]:\[Destination]**

Exemple : Pour déplacer le fichier VIVALDI.TXT se trouvant dans le répertoire racine du disque dur vers le sous-répertoire CLASSIC du répertoire père MUSICAL, tapez : C:\>**MOVE VIVALDI.TXT C:\MUSICAL\CLASSIC**

ii. *Déplacer un groupe de fichiers*

Le déplacement d'un groupe de fichiers ou de tous les fichiers d'un répertoire vers un autre s'effectue, en plus de la commande MOVE, à l'aide des jokers de MS-DOS.

Syntaxe : C:\>**MOVE [Combinaison avec les jokers] [Unité disque]:\[Destination]**

Exemple1 : Pour déplacer tous les fichiers du répertoire DOS vers le répertoire racine du disque dur, tapez : C:\DOS>**MOVE *.* C:**

Exemple2 : Pour déplacer tous les fichiers à extension .BAT du répertoire DOS vers le répertoire père MUSICAL, tapez : C:\DOS>**MOVE *.BAT C:\MUSICAL**

6. Comment s'effectuent les suppressions de fichiers ?

La suppression de fichiers (ou l'effacement d'un fichier ou d'un groupe de fichiers) s'effectue à l'aide de la commande "**DEL**" ou "**ERASE**". Pour l'effacement des répertoires et de leurs contenus en fichiers, voir la commande **DELTREE**.

Pour effectuer la suppression d'un ou plusieurs fichiers, il faut que l'invite en cours (unité disque et répertoire courant) soit celle qui contient les fichiers spécifiés.

i. *Supprimer un fichier*

Il suffit de préciser après la commande DEL ou ERASE, le nom du fichier à supprimer et son extension, si elle existe.

Syntaxe : C:\>**DEL [Nom.extension]**

ii. *Supprimer un groupe de fichiers*

La suppression d'un groupe de fichiers, ou de tous les fichiers d'un répertoire, s'effectue, en plus de la commande DEL ou ERASE, à l'aide des jokers de MS-DOS.

Syntaxe : C:\>**DEL[Combinaison avec les jokers]**

Exemple1 : Pour supprimer le fichier MOZART.TXT se trouvant dans le sous-répertoire CLASSIC de MUSICAL, tapez : C:\MUSICAL\CLASSIC>**DEL MOZART.TXT**
Si vous n'êtes pas dans le bon répertoire courant, c'est-à-dire là où se trouvent les fichiers, le message d'erreur suivant s'affiche : **Fichier introuvable**
Exemple2 : Pour supprimer tous les fichiers se trouvant dans le sous-répertoire CLASSIC, il suffit de taper : C:\MUSICAL\CLASSIC>**DEL *.***
Exemple3 : Pour supprimer tous les fichiers à extension .COM du répertoire DOS, par exemple, tapez : C:\DOS>**DEL *.COM**

7. Comment renommer des fichiers ?

Pour renommer un ou plusieurs fichiers, il faut se servir de la commande "**REN**" ou "**RENAME**". Ces commandes doivent obligatoirement 2 paramètres : l'ancien nom et l'extension d'un ou plusieurs fichiers et le nouveau nom et l'extension que l'on souhaite attribuer au nouveau fichier ou groupe de fichiers

Pour renommer un ou plusieurs fichiers, il faut que l'invite en cours (unité disque et répertoire courant) soit celle qui contient les fichiers à renommer.

i. *Renommer un fichier*

La syntaxe de renommage d'un fichier est la suivante :

C:\>**REN NOM1.EXTENSION1 NOM2.EXTENSION2**

Exemple1 : Pour renommer le fichier MOZART.TXT en VERDI.TXT, tapez :

C:\>MUSICAL\CLASSIC**>REN MOZART.TXT VERDI.TXT**

Exemple2 : Pour renommer le fichier MOZART.TXT en MOZART.DOC, tapez :

C:\>MUSICAL\CLASSIC**>REN MOZART.TXT MOZART.DOC**

ii. *Renommer un groupe de fichiers*

La syntaxe de renommage d'un groupe de fichiers est la suivante :

C:\>REN [NOM1 combiné avec jokers] [NOM2 combiné avec jokers]

Exemple1 : Pour renommer tous les fichiers à extension .EXE du répertoire racine en extension .TXT, tapez : C:\>**REN *EXE *TXT**

Exemple2 : Pour renommer tous les fichiers du répertoire WINDOWS, commençant par AUT en OUT, tapez : C:\WINDOWS>**REN AUT*.* OUT*.***

8. Les attributs des fichiers

Les fichiers possèdent quatre attributs (a, r, h, s). Ces attributs indiquent les spécificités des fichiers qui peuvent soit être effaçables ou non, visibles (cachés) ou non, systèmes ou d'archives.

Pour afficher les attributs d'un fichier, il suffit de faire précéder celui-ci de la commande **ATTRIB**.

Exemple1 : Pour afficher tous les attributs des fichiers d'un répertoire d'une unité disque quelconque, il faut taper : C:\Nom de répertoire>**ATTRIB |MORE**

Exemple2 : Pour afficher les attributs d'un fichier, il suffit de taper la commande ATTRIB suivie du nom du fichier : C:\>**ATTRIB [Nom fichier.Extension]**

9. La copie de fichiers et de répertoires avec la commande XCOPY

La commande **XCOPY** permet d'enregistrer les fichiers des répertoires et sous-répertoires d'un support magnétique de stockage à l'autre. La syntaxe générale de cette commande est la suivante :

C:\>**XCOPY[Origine][Destination]/[Commutateur]**

Le paramètre [**Origine**], doit indiquer l'emplacement où sont enregistrés les fichiers ou les répertoires à copier. Le paramètre [**Destination**] doit préciser l'emplacement où seront stockés les fichiers ou les répertoires et leurs contenus.

Exemple1 : Pour copier le contenu du répertoire DOS du disque dur vers un autre emplacement du disque dur sous le nom MS-DOS, tapez :

C:\>**XCOPY DOS C:\MS-DOS /S**

Exemple1 : Pour copier le contenu d'une disquette (lecteur A:) vers le disque dur, tapez :

A:\>**XCOPY A: C: /S**

MODULE1 : ENVIRONNEMENT NUMERIQUE, SECURITE, MULTIMEDIA ET GESTION DES DONNEES

CHAPITRE 3 : DESCRIPTION DES CONCEPTS DE BASE DE LA SECURITE INFORMATIQUES

LECON : DESCRIPTION DES CONCEPTS DE BASE DE LA SECURITE INFORMATIQUES

Compétences visés :

- Définir les concepts fondamentaux de la sécurité informatique(confidentialité, disponibilité, intégrité, non répudiation)
- Définir les concepts de cybercriminalité, cybersécurité
- Décrire les techniques de protection des données

Situation problème1 : vous recevez un message d'une personne inconnue X vous disant que votre numéro de téléphone à été tiré au sort vous donnant ainsi la possibilité de gagner une importante cagnotte de 1000000 frs(un million de francs). Mais pour entrer en possession de votre argent l'inconnu X vous propose de composer et de valider un code qu'il va vous donner. Très ému de cette offre mielleuse, vous composez et validez le code qu'il vous a transmis. Quelque seconde plus tard vous recevez un message mobile money vous stipulant que votre compte à été vidé. C'est alors que vous comprenez que vous venez de vous faire arnaquer. Comment cela à t'il été possible ? comment appelle t-on ce genre de personnes et de delits commis à travers les réseaux informatiques ?

Situation problème2 : La société Alpha est une entreprise qui fait dans la recherche et le développement. A la fin d'une étude basée sur le développement rural, le fruit de ses recherches est divulgué sur internet. Ce posant plusieurs questions, le responsable de cette structure demande aux experts comment leurs données ont pu se retrouver dans la rue sans leur autorisation ?.

INTRODUCTION

Avec le développement de l'utilisation d'internet, de plus en plus d'entreprises ouvrent leur système d'information à leurs partenaires ou leurs fournisseurs, il est donc essentiel de connaître les ressources de l'entreprise à protéger et de maîtriser le contrôle d'accès et les droits des utilisateurs du système d'information. Il en va de même lors de l'ouverture de l'accès de l'entreprise sur internet.

La sécurité informatique, d'une manière générale, consiste à assurer que les ressources matérielles ou logicielles d'une organisation sont uniquement utilisées dans le cadre prévu. La sécurité informatique vise à réduire au maximum le **risque** qui est une combinaison des

notions (menace, vulnérabilité et contre-mesure). La menace représente le type d'action susceptible de nuire dans l'absolu, tandis que la **vulnérabilité** (en anglais « *vulnerability* », appelée parfois *faille* ou *brêche*) représente le niveau d'exposition face à la menace dans un contexte particulier. Enfin la **contre-mesure** est l'ensemble des actions mises en oeuvre en prévention de la menace.

I. OBJECTIFS DE LA SECURITE INFORMATIQUE

La sécurité informatique vise généralement cinq principaux objectifs :

- L'**intégrité**, c'est-à-dire garantir que les données sont bien celles que l'on croit être ; consiste à déterminer si les données n'ont pas été altérées durant la communication (de manière fortuite ou intentionnelle).
- La **confidentialité**, consistant à assurer que seules les personnes autorisées aient accès aux ressources échangées ; consiste aussi à rendre l'information inintelligible à d'autres personnes que les seuls acteurs de la transaction.
- La **disponibilité**, permettant de maintenir le bon fonctionnement du système d'information ; L'objectif de la disponibilité est de garantir l'accès à un service ou à des ressources.
- La **non répudiation**, permettant de garantir qu'une transaction ne peut être niée ; La **non-répudiation** de l'information est la garantie qu'aucun des correspondants ne pourra nier la transaction.
- L'**authentification**, consistant à assurer que seules les personnes autorisées aient accès aux ressources. L'authentification consiste à assurer l'identité d'un utilisateur, c'est-à-dire de garantir à chacun des correspondants que son partenaire est bien celui qu'il croit être. Un contrôle d'accès peut permettre (par exemple par le moyen d'un mot de passe qui devra être crypté) l'accès à des ressources uniquement aux personnes autorisées.

II. LES ATTAQUES

Il existe quatre catégories principales d'attaque :

- ✓ **Les attaques par l'accès**. C'est une tentative d'accès à l'information par une personne non autorisée.

  Exemples : ***Sniffing*** cette attaque est utilisée par les pirates informatiques pour obtenir des mots de passe. ***Craquage de mot de passe*** il consiste à faire de nombreux essais jusqu'à trouver le bon mot de passe
- ✓ **Les attaques par modification,** elle consiste, pour un attaquant à tenter de modifier des informations. Quelques exemples de ce type d'attaque : ***Virus, et les vers etc…*** **(**Un vers comme un programme caché dans un autre qui peut s'exécuter et se reproduire en infectant d'autres programmes ou d'autres ordinateurs il se déplace à travers un réseau.**)**
- ✓ **Les attaques par déni de service,** elle consiste à envoyer des milliers de messages depuis des dizaines d'ordinateurs, dans le but de submerger les serveurs d'une société,

de paralyser pendant plusieurs heures son site Web et d'en bloquer ainsi l'accès aux internautes.

Exemple ***Le flooding*** Cette attaque consiste à envoyer à une machine de nombreux paquets IP de grosse taille. ***Le smurf*** *c'est une* attaque qui s'appuie sur le ping et les serveurs de broadcast. On falsifie d'abord son adresse IP pour se faire passer pour la machine cible.

- ✓ **Les attaques par répudiation.** La répudiation est une attaque contre la responsabilité. Autrement dit, la répudiation consiste à tenter de donner de fausses informations ou de nier qu'un événement ou une transaction se soit réellement passé.

Ces attaques sont basées sur les types de virus suivant :

- **virus**: un programme malveillant ayant pour but de nuire au bon fonctionnement de l'ordinateur.
- **Ver**(worm): programme malveillant qui se propage en utilisant un réseau informatique (comme Internet)
- **cheval-de-troie** (trojan): un programme malveillant qui ouvre une porte dérobée dans un système afin de l'exploiter ultérieurement.
- **Spam**: courrier indésirable envoyé dans l'intention de créer des deni de service, ou alors de distraire et de faire oublier l'essentiel.

III. CONCEPTS DE CYBERSECURITE ET CYBER CRIMINALITE

La cyber-sécurité renvoie à la protection des personnes, des idées et des données dans le cyber espace. La sécurisation des données informatiques ou cyber sécurité, passe par une sécurisation des matériels et logiciels informatiques constituant le support de transmission, de stockage et de traitement de l'information. L'augmentation de la sécurité des matériels et logiciel informatique, permettrait donc d'augmenter la sécurité des informations.

Deux types de personnes animent ces concepts :

- ✓ **Black hat hacker:** pirate des systèmes informatiques, s'introduisant de façon furtive ou masquée pour voler des informations sensibles ou créer des dénis de service. Ces genres de personnes font dans ce qu'on appelle **cybercriminalité**(le fait d'utiliser ses connaissances informatique pour commettre des délits des actes préjudiciables).
- ✓ **white-hat hacker** : ce sont des « hackers éthiques » des experts en sécurité informatique qui utilisent leurs capacités à des fin honnêtes éthiques et du côté de la justice. Un **hacker white-hat** qui trouve une faille de sécurité dans une application la rapportera à son développeur lui permettant ainsi d'améliorer la sécurité de celle-ci avant qu'elle ne soit compromise. Ces genres de personnes font dans ce qu'on appelle **cybersécurité** (le fait d'utiliser ses connaissances informatique pour empêcher la

défaillance, détecter et fermer les failles d'un système informatique dans une entreprise).

IV. PROTECTION DES DONNEES

La protection des données consiste à déployer des moyens et des dispositifs visant à sécuriser le système d'information ainsi que de faire appliquer les règles définies dans la politique de sécurité. Les principaux dispositifs permettant de sécuriser un réseau contre les intrusions sont :

- ✓ **Les mots de passe** Lors de la connexion à un système informatique, celui-ci demande la plupart du temps un **identifiant** (en anglais *login* ou *username*) et un **mot de passe** (en anglais *password*) pour y accéder. Ce couple *identifiant/mot de passe* forme ainsi la clé permettant d'obtenir un accès au système. Pour des données sensibles à protéger il est conseillé d'utiliser un mot de passe d'au moins 14 caractères contenant des lettres , des chiffres et des caractères spéciaux.

- ✓ **Les systèmes pare-feu**. Dispositif matériel et logiciel qui protège un système informatique connecté à internet des tentatives d'intrusion qui pourraient en provenir. Le pare feu définit les types de communications autorisés, surveille et contrôle les applications et les flux de données. Néanmoins ce type de dispositif ne protège pas la confidentialité des données circulant sur le réseau.

- ✓ **La cryptographie** discipline de la cryptologie s'attachant à protéger les messages en utilisant des clés. La cryptographie vise à rendre le message inintelligible à toutes autres personnes sauf le destinataire du message. Elle permet ainsi de garantir la confidentialité des échanges. On s'appuis généralement sur plusieurs algorithmes cryptographiques tels que DES, RSA, etc..

- ✓ **Antivirus** ce sont des programmes permettant de détecter et de neutraliser éliminer tout programme malveillant présent dans l'ordinateur. Il existe plusieurs antivirus telque : Kaspersky, Avast, Norton, Avira etc..

- ✓ Le **VPN** (Virtual Private Network) correspond à la mise en place de tunnels sécurisés. Ce système permet de créer un lien direct entrer des ordinateurs distants qui isole leurs échanges du reste du trafic se déroulant sur le réseau. (permet d'obtenir un niveau de sécurisation supplémentaire dans la mesure où l'ensemble de la communication est chiffrée.

- ✓ **Droit d'accès et privilèges** les droits d'accès sont des privilèges attribués à un utilisateur pour accéder à des informations. L'utilisateur peut bénéficier des privilèges suivants : lecture, écriture ou modification, privilège d'administration dans le cas des serveurs ou des appareils de sécurité comme le firewall (pare feu matériel). L'attribution des privilèges peut se faire à un utilisateur spécifique ou à un groupe d'utilisateurs. Il existe plusieurs niveaux de privilèges :

    - **Le super administrateur :** c'est celui qui est propriétaire de tous les dossiers et fichiers du système

    - **L'administrateur :** reçois du super administrateur le droit aussi d'accéder et de modifier les données sur tous les fichiers

- **L'invité :** dispose d'un accès limité dans le système ; ne peut pas modifier ou accéder aux données dont il n'a pas créer.

CONCLUSION

La sécurité d'un système informatique doit être abordée selon une approche globale. Ainsi, une porte blindée est inutile dans un bâtiment si les fenêtres sont ouvertes sur la rue.

Cela signifie que la sécurité doit être abordée dans un contexte global et notamment prendre en compte les aspects suivants :

- **La sensibilisation des utilisateurs aux problèmes de sécurité**
- **La sécurité logique**, c'est-à-dire la sécurité au niveau des données, notamment les données de l'entreprise, les applications ou encore les systèmes d'exploitation.
- **La sécurité des télécommunications : technologies réseau, serveurs de l'entreprise, réseaux d'accès, etc.**
- **La sécurité physique**, soit *la sécurité au niveau des infrastructures matérielles* : salles sécurisées, lieux ouverts au public, espaces communs de l'entreprise, postes de travail des personnels, etc.

EXERCICES

Question1 : Définir la notion d'intégrité des données ainsi que les objectifs du contrôle d'intégrité

Question2 : quelles relations existent entre les critères d'intégrité et de confidentialité ?

Question3 : Dans quelle mesure peut on considérer des principes d'éthique comme faisant partie d'une démarche de sécurité informatique ?

Question4 : Sur quels principes se fonde la réalisation des attaques informatiques ?

Question5 : Quelle est la place de la maitrise de la cybercriminalité dans une démarche de cybersécurité ?

Question6 : Ressortez clairement la différence entre un virus, un ver, un cheval de troie, une bombe logique.

Question7 : Parmis les infrastructures qui composent un système d'information laquelle ne peut etre concernée par une cyberattaque ?

A - Matérielle B- Réseau C- Logicielle D- Exploitation E- Structurelle

Question8 : Parmis les attaques informatiques suivantes quelle est celle qui peut etre qualifiée d'attaque passive ?

A - modification B- interception C- Fabrication D- Interruption E- Destruction

Question9 : Parmis les propositions suivantes laquelle correspond le mieux à la définition d'une infraction informatique ?

A – « Tout comportement illégal, immoral ou non autorisé qui implique la transmission et/ou le traitement automatique des données »

B- « tout comportement illégal qui implique l'usage de l'informatique et des télécommunications »

Question10 : Expliquez comment le concept de « séparation des taches » contribue à la sécurité informatique d'une organisation.

MODULE1 : ENVIRONNEMENT NUMERIQUE, SECURITE, MULTIMEDIA ET GESTION DES DONNEES

CHAPITRE4 : UTILISER LES FICHIERS MULTIMEDIAS

LECON 4 : UTILISER LES FICHIERS MULTIMEDIAS

Compétences visées :

- Décrire les caractéristiques (définition, résolution, taille) d'une image.
- Calculer la résolution d'une image matricielle
- Calculer la taille d'un fichier multimédia

Situations problème : Le jour de votre baptême votre petit frère Tamo, muni de plusieurs appareil photographiques, se charge de faire des prises successives de photos afin d'en faire une vidéo à la fin de la cérémonie. À la fin, il constate que certaines images sont plus nettes, plus claires, que d'autres.

Qu'est ce qui peut expliquer cet état des choses ? Pourquoi les photos prises le même jour pendant une cérémonie n'ont pas la même qualité ?

(reponse : les appareils utilisés, n'ont pas la même résolution, c'est a dire elles ne produisent pas les photos avec la même résolution)

Voulant alors transférer ces images dans sa clé USB un message lui est affiché disant que la l'espace de stockage est insuffisant. Tamo veut alors calculer par lui-même la taille des fichiers image et vidéos pris pendant la fête.

INTRODUCTION

Un média est support d'information. Les média les plus courant en informatique sont : les disquettes, les cd-rom, les disques dur. **Le multimédia** fait référence à l'ensemble des matériels et des techniques permettant de traiter tout à la fois du texte, du son, des images, de la vidéo, etc.

I. L'IMAGE NUMERIQUE

1. Qu'est-ce que l'image numérique ?

On désigne sous le terme d'image numérique toute image (dessin, icône, photographie ...) acquise, créée, traitée, et stockée sous forme binaire (suite de 0 et de 1) :

- Acquise par des dispositifs comme les scanners, les appareils photo.
- Créée directement par des programmes informatiques.
- Traitée grâce à des outils informatiques.
- Stockée sur un support informatique (disquette, disque dur, CD-ROM, clé USB ...).

2. Type d'images

Il existe 2 sortes d'images numériques : les images matricielles et les images vectorielles.

- **L'image matricielle (bitmap)**

Une image matricielle est composée d'un ensemble de points (pixels). Chacun pouvant avoir une couleur différente. Une image matricielle est caractérisée notamment par (sa dimension en pixels, sa résolution, son mode colorimétrique). Les images vues sur un écran de télévision ou une photographie sont des images matricielles. On obtient également des images matricielles à l'aide d'un appareil photo numérique, d'une caméra vidéo numérique ou d'un scanner. L'image matricielle perd en netteté et en qualité lorsqu'elle est redimensionnée ou agrandie. Les formats d'images matricielles sont : PNG, GIF, JPEG, TIFF,BMP, …

- **L'image vectorielle**

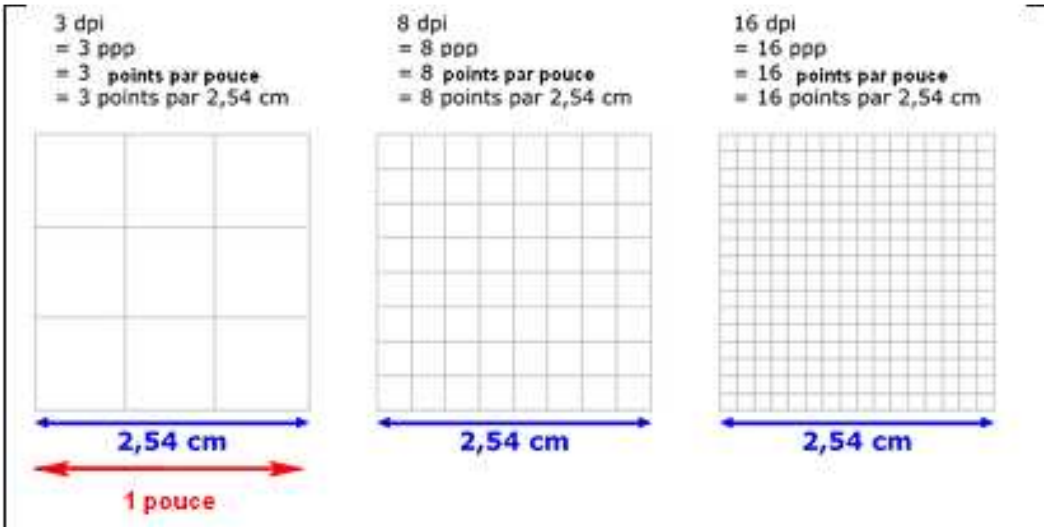
Elle est définie par des fonctions mathématiques qui décrivent des lignes, des courbes, des formes géométriques, etc. Dans ce cas on manipule des objets et non des pixels. Par exemple, un cercle est décrit par une fonction du type (cercle, position du centre, rayon). Une image vectorielle se redimensionne aisément, sans aucune perte de qualité. Ces images sont très utiles pour des reproductions à grande échelle, des calculs pouvant être réalisés à chaque fois, afin d'obtenir l'image exacte, quelle que soit la taille choisie. Ces images sont essentiellement utilisées pour réaliser des schémas ou des plans. Ces images présentent 2 avantages : elles occupent peu de place en mémoire et peuvent être redimensionnées sans perte d'information c'est-à-dire sans perte de qualité. Quelques formats d'images vectorielles : SVG, EPS, WMF,

3. Définition ou Dimension de l'image matricielle

Une image matricielle contient un nombre fixe de pixels en hauteur et en largeur. Sa dimension en pixels correspond au nombre total de pixels qui la constituent. La définition d'une image ne doit pas être confondue avec sa résolution. La définition s'exprime via le nombre de pixels en largeur et le nombre de pixels en hauteur. Le pixel (Picture Element, px) est une unité de base, c'est le plus petit élément constitutif d'une image matricielle. Elle est de taille carrée ou rectangulaire. Elle permet de mesurer la définition d'une image matricielle.
Il faut en tenir compte pour l'affichage de l'image sur l'écran. Une image de 1385 sur 741 pixels s'affichera en réduction sur un écran de 800 par 600 pixels. Si on l'a fait s'afficher en taille réelle (100%), elle ne s'affichera pas complètement sur l'écran.

4. Résolution de l'image matricielle

La résolution d'une image est définie par le nombre de pixels par unité de longueur que renferme cette image. Elle s'exprime en dpi (dots per inch) ou ppp (points ou pixels par pouce). Un pouce = 2,54 centimètres.
La résolution d'une image numérique définit le degré de détail qui va être représenté sur cette image.
Une image de résolution élevée compte un plus grand nombre de pixels (elle contient plus d'informations), elle est donc plus volumineuse qu'une image basse résolution de mêmes dimensions.
La résolution est un paramètre qui se définit lors de la numérisation d'une image (scanner, création,…).



Une image est bidimensionnelle (largeur et hauteur).

$$résolution\ de\ l'image = \frac{définition\ de\ l'image}{Dimension\ réelle\ en\ pouce}$$

Exemple :
Prenons l'exemple d'une image de 26cm de de largeur et 19,51cm de hauteur et dont la définition est 1024 X 768 pixels (largeur X hauteur).
Calculons la résolution de cette image.
Solution
La résolution cest le nombre de pixels par pouce : déterminons le nombre de pouces dans 26cm.
26cm/2,54= 10,23 pouces
Résolution= 1024 /10,23
= 100,09DPI

Le même raisonnement s'applique avec la largeur pour obtenir le même résultat.

5. Calcul de la taille d'une image

Une image numérique occupe une certaine place en mémoire. Cette place utilisée s'exprime en octets, Bytes ou bits. 1octet=1Byte=8bits.
Le nombre de couleurs utilisées influence la précision de l'image et sa qualité. Voici des exemples illustratifs. Une image indexée par 4bits peut gérer jusqu'à 16 couleurs. Avec 8bits on peut avoir jusqu'à 256 couleurs. 16 bits correspond à 65536 couleurs. On retiendra que :

- 1 pixel noir ou blanc occupe 1 bit.
- En 16 couleurs, un pixel occupe 4 bits.
- En 256 couleurs (ou 256 niveaux de gris), un pixel occupe 8 bits (1 octet). Il s'agit de couleurs fixes, définies à l'avance dans une palette.
- En 65536 couleurs (high colors), un pixel correspond à 16 bits (2 octets).
- En 16,7 millions de couleurs (true colors, 256 X 256 X 256), un pixel occupe 3 octets (24 bits). Grâce à une composition réalisée au départ de trois couleurs de base (Rouge, Vert, Bleu), on peut obtenir des couleurs très précises, au niveau de chaque pixel. Il s'agit du mode RVB (RGB, Red, Green, Blue). Notons qu'un octet supplémentaire est possible pour la transparence ou la texture (4 octets, 32 bits).

Exemple

Déterminer le poids (taille sur le disque) d'une image 1024x768 pixels. Sachant qu'elle est composée de 256 couleurs.

Solution

Si l'image est composée de 256 couleurs=2^8, alors un pixel occupe 8 bits en mémoire.
1024x768=786432 pixels au total. Or 1pixel=8bits
D'où 786432pixels occupera 786432x8=6 291 456 bits=786432octets=768Ko=0,75Mo

EXERCICES DE CONSOLIDATION

Exercice1 :
Une image a une largeur de 190 pixels (définition). Sachant que sa résolution est de 96 DPI, calculer la dimension réelle en centimètre de cette image.

Solution

Cette largeur se calcule comme suit.
190/96 = 1,98 pouces Sachant que 1 pouce = 2,54 cm (approximativement), on obtient la largeur suivante. 1,98 X 2,54 = 5,03 centimètres.

Exercice2 :
Une image a une résolution de 72 DPI. Avec cette résolution, sa hauteur mesure réellement 8 centimètres. Quelle est la hauteur en pixels de cette image (définition) ?

Solution

Cette définition portant sur la hauteur se calcule en convertissant tout d'abord en pouces la mesure réelle. Exprimée en centimètres. 1 pouce = 2,54 cm
8/2,54 = 3,15 pouces
La hauteur de notre image en pixels (définition) est donc de 72 X 3,15 = 227 pixels (approximativement)

Exercice3 : une image est composée de 70000 couleurs.

- Combien de bit minimums faut-il pour stocker un pixel en mémoire ?
- Quelle est la taille minimale de cette image sur le disque si elle à une résolution de 72dpi une largeur de 10cm et une hauteur de 12cm.

Exercice4 :

La photographie Monument.png est téléchargeable sur le site www.unesco.com. Cette photo est orientée paysage, elle est en couleurs, avec une palette de couleurs de 16 Millions de couleurs et les options de téléchargement nous informent que sa définition est de 600 pixels x 1000 pixels. On prendra : 1 Mo = 1000 Ko et 1 Ko = 1000 octets.

Calculer :

a) Le nombre de lignes de pixels pour cette photo.

b) Le nombre de colonnes de pixels pour cette photo.

c) Le nombre de pixels contenus dans cette photo.

d) La taille du fichier Monument.png, en octets, en kilo-octets et en méga-octets.

e) La taille du fichier Monument.png, en bits.

II. LE SON ET LA VIDEO

1. Définition

D'un point de vue physique, un son est une énergie qui se propage sous forme de vibrations dans un milieu compressible (dans l'eau, dans l'air, dans les matériaux solides, mais pas dans le vide!). La vidéo est une suite d'images animées en mouvement et accompagnées d'un son.

2. Émission, Propagation, Réception

- Pour qu'un son soit émis, une énergie doit avant tout mettre en mouvement un corps pour produire une vibration. Ainsi, la chute d'un objet sur le sol, ou la tension électrique dans un haut-parleur, provoqueront l'énergie nécessaire pour produire cette vibration.
- Ensuite, pour que ce son puisse se propager, il faut un milieu élastique favorable à la transmission de la vibration. En créant des surpressions ou des dépressions, l'air permet la propagation de l'onde. Les matériaux solides ont aussi cette capacité de transmettre le son. Dans le vide par contre, aucun son ne peut se propager, car il n'y a aucun de support.
- Enfin, pour être perçue, il doit y avoir un récepteur sensible. Chez l'homme, l'oreille possède une membrane (le tympan) capable de transmettre les informations de vibration en signaux nerveux jusqu'au cerveau, grâce au nerf auditif. De même, le microphone possède également une membrane permettant de transformer les déplacements de l'air en signaux électriques

3. Caractéristiques du son

Comme tout phénomène vibratoire, le son peut être analysé comme un signal qui varie dans le temps. Deux caractéristiques essentielles sont l'amplitude et la fréquence.

- **L'amplitude :**Appelée aussi intensité ou volume sonore, c'est l'expression de la pression de l'air qui se mesure en décibels (dB). 0 dB correspond au minimum que l'oreille humaine puisse percevoir (seuil d'audibilité).
- **La Fréquence :** La fréquence, exprimée en Hertz (Hz), est le nombre de répétition d'une période par seconde. Plus elle est élevée et plus le son paraitra « aiguë », à l'inverse, il paraitra « grave ». En musique, la fréquence définit donc la hauteur d'un son, soit, la note.

4. Calculer la taille d'un fichier son

Le poids d'un fichier son est tributaire de :

- La fréquence d'acquisition ou encore le nombre d'échantillons considérés par seconde. Il faut comprendre le nombre de fois par seconde ou l'on va prélever un instantané du son produit pour le coder et le placer après l'échantillon prélevé juste avant. Il s'agit de la fréquence d'échantillonnage choisie pour numériser le son à coder. Généralement 44,1KHZ
- La profondeur de codage utilisé (généralement 16bits) qui permet d'affiner la numérisation du signal analogique, en sa valeur la plus proche de la précision près conditionné par le nombre de bits utilisés.

- Le format stéréo ou mono de la piste
- Sa durée

Taille d'un fichier son (bits) = **profondeur de codage** (bits) x **Fréquence d'échantillonnage** (Hz) x **Durée** (s)

Exemple : pour un CD échantillonné à 44,1Khz sur 16bits stéréo en 2 secondes on aura une taille de donnée de :

44100x16x2=1411200bits=176400octets

5. Calculer la taille d'un fichier vidéo

Le poids d'une vidéo dépend du nombre de points constituant cette vidéo, la fréquence d'image par seconde, la durée de la vidéo, le nombre de bits nécessaires pour coder un couleur. Il depend aussi de son débit d'encodage et de sa durée. Si le poids s'exprime généralement en méga-octets (Mo), le débit est généralement connu en kilobits/seconde (Kbps).

Taille vidéo(bits)=**nbre point** x **profondeur**(bits) x **frequence**(img/s) d'image x **durée**(s)

Taille vidéo (Mo) = **débit** (Kbps) x **durée** (s) / 8 / 1024

Exemple : le poids en Mo d'une vidéo de 2' encodée à 500Kbps est de 500 x 120 / 8 / 1024 = 7,32Mo

EXERCICES DE CONSOLIDATION

Exercice1 : calculer la taille (GB) de 20min vidéo en 1028x640pts en 4K couleurs. Sachant que le standard d'image est de 24img/sec.

Solution

4Kilo couleurs=4000 couleurs et ceci demande 12 bits pour coder une couleurs.

20min= 20x60= 1200s

1028x640pt=657920

Taille=657920x12x1200x24=227377152000bits=26,47GB

Exercice2 : Calculer la taille (MB) d'une heure audio en stéréo 12bits à 128Khz.

Solution

1h=3600s

Steréo=2voies

Taille= 128000x12x3600x2=11059200000bits=1318,35MB

MODULE1 : ENVIRONNEMENT NUMERIQUE, SECURITE, MULTIMEDIA ET GESTION DES DONNEES

CHAPITRE 5: DESCRIPTION DES CONCEPTS DE BASE DES SYSTEMES D'INFORMATION

Compétences visées :

- *Décrire les systèmes d'une entreprise (pilotage, d'information, opérant)*
- *Énumérer les composants d'un SI (les ressources humaines, matérielles et logicielles)*
- *Décrire les fonctions d'un SI (collecte, stockage, traitement, diffusion).*
- *Décrire les intérêts d'un SI ;*
- *Enumérer quelques méthodes de conception d'un SI.*
- *Écrire les composants d'un SGBD ;*
- *Décrire les fonctions de manipulation d'une base de données (insertion, mise à jour, suppression, consultation) ;*

LEÇON1 : NOTION DE BASES DES SI

Pré – Requis :

- *Définir : Information, Données, Traitement ;*
- *Lister les qualités d'une bonne information;*
- *Donner le rôle de l'unité centrale*

Situation de vie

La société Njofruit's fait dans la production des jus de fruit au Cameroun. Comme matière première, elle utilise des fruits tels que la mangue, la papaye, la pomme, le fruit de la passion…

Consigne :

a) Enumérer les éléments de l'entreprise qui s'occupent de la fabrication des jus de fruits ?

- *On a : mangues, papaye, pomme, fruit de la passion…*

b) ***A base de ces fruits, que doit fait l'entreprise pour obtenir les jus ?***

- *L'entreprise doit transformer et ou traiter ces fruits pour avoir des jus.*

Objectifs : A la fin de cette leçon, l'apprenant sera capable de mobiliser les ressources pour pouvoir:

- Définir les notions de : Information, Donnée, Entreprise, traitement
- Décrire les systèmes constituant une entreprise
- Donner la structure pyramidale d'une entreprise

Introduction

Le SI peut être comparé à une sorte de **système nerveux** primaire de l'organisation tel que: la circulation rapide d'une information de qualité entre les différents « organes » entre autre la Délivrer la bonne information, au bon interlocuteur, au bon moment (Prise de décisions appropriées et Action de l'entreprise adaptée à la situation) enfin Le SI contribue donc de manière évidente aux performances de l'organisation.

I. DEFINITION

- **Information :** C'est une connaissance codée susceptible d'être transmise et conservée. Elle peut encore considérer comme une donnée qui a un sens ;
- **Donnée :** C'est la représentation d'une information sous sur forme conventionnelle destinée à faciliter son traitement ;
- **Traitement** : C'est l'ensemble d'opérations effectuées sur une ou plusieurs données pour obtenir un résultat (information).ou encore plus simplement c'est la Transformation des données en informations ;
- **Entreprise :** unité économique, juridiquement autonome, organisée pour produire des biens ou des services pour le marché. Elle aussi un ensemble de moyen (projet, lieu de décision et de gestion économique).

II. LES SYSTÈME CONSTITUANT UNE ENTREPISE

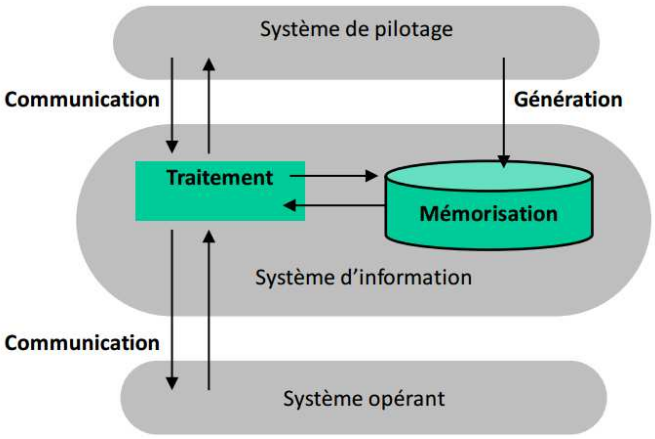
Comme tout système, l'entreprise est un système car, il est ouvert sur l'environnement, Il est finalisé[1] (but = profit…) et surtout, il est en constante évolution. Pour parvenir à son but, le système tient compte de son environnement et régule son fonctionnement en s'adaptant aux changements.

Les éléments du système sont eux-mêmes des systèmes (ou sous-systèmes). L'entreprise peut se décomposer en 3 sous-systèmes :

- ➔ Le système de décision ;
- ➔ Le système d'information ;
- ➔ Le système opérant.

[1] C'est-à-dire son but principal c'est le profit, le gain

Ces trois sous-systèmes sont liés car chaque système apporte ses services à l'autre comme l'indique la figure ci – dessus :



1. Le système de pilotage ou système de décision

Le système de pilotage définit les objectifs de l'entreprise et s'efforce de tout mettre en œuvre pour qu'ils soient atteints. Pour cela, il prend des décisions. Ces décisions sont prises à partir de paramètres venant du système opérant. **Exemple : la société Njofruit's doit définir une politique commerciale en fonction du marché et de la concurrence.**

Le système de décision a pour but de :

- ➔ Exploiter les informations qui circulent ;
- ➔ Organiser le fonctionnement du système ;
- ➔ Décider des actions à conduire sur le système opérant ;
- ➔ Raisonner en fonction des objectifs et des politiques de l'entreprise

2. Le système opérant ou opérationnel

Le système opérant constitué de la partie du système qui s'occupe effectivement de transformation des matières premières (les machines, les ouvriers, les techniciens…). Exemple : la société Njofruit's doit produire les jus de fruit

Le système opérant a pour rôle de:

- ➔ Recevoir les informations émises par le système de pilotage ;
- ➔ Se charger de réaliser les tâches qui lui sont confiées ;
- ➔ Générer à son tour des informations en direction du système de pilotage (Qui peuvent ainsi contrôler les écarts et agir en conséquence) ;
- ➔ Il en englobe toutes les fonctions liées à l'activité propre de l'entreprise (Facturer les clients, régler les salaires, gérer les stocks, …)

3. Système d'information

Le système d'information est vu comme la partie qui relie les deux sous-systèmes précédents. Exemple : la société Njofruit's doit gérer les informations de la relation clientèle, de la

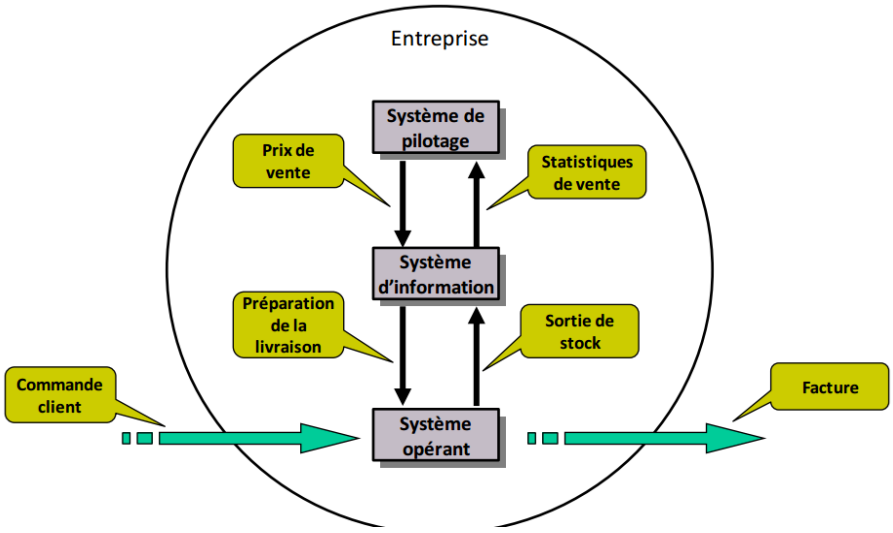
production des jus de fruit, de sa publicité de son rapport à l'environnement de la validation économique et de sa stratégie

- ➔ Pour organiser son fonctionnement, le système a besoin de mémoriser des informations (Pour comparer, prévoir, …) ;
- ➔ Ce rôle est joué par le Système d'Information ;
- ➔ Ce système a aussi la charge de :
  - ❖Diffuser l'information ;
  - ❖Réaliser tous les traitements nécessaires au fonctionnement du système.

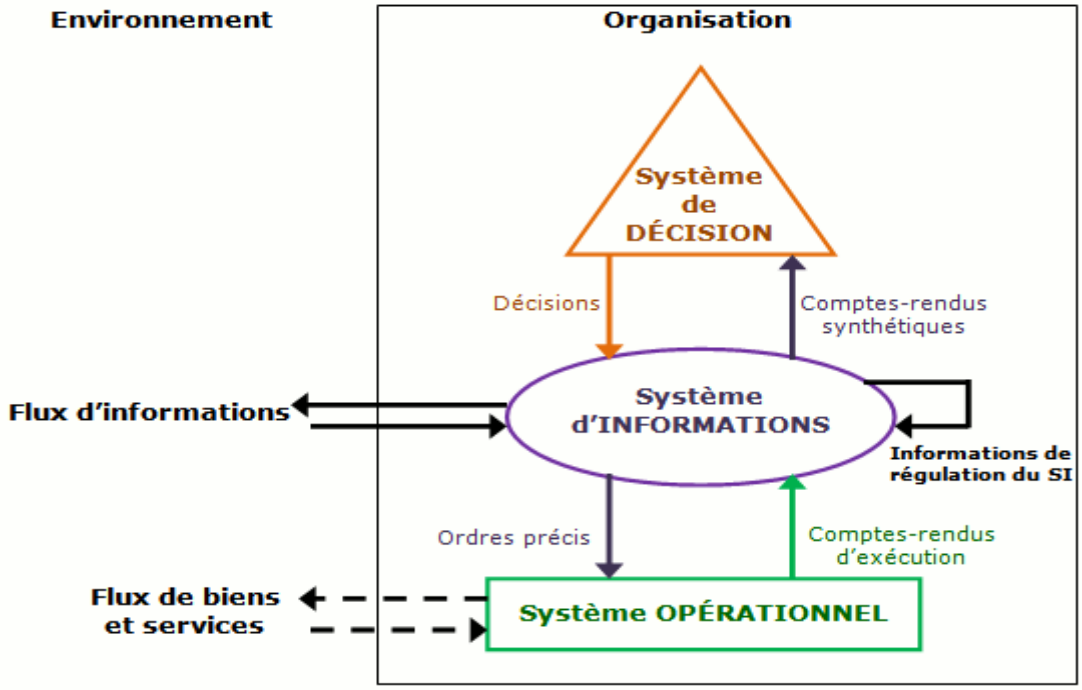
En bref on peut donc dire que un système d'information est l'ensemble des ressources (matériels, logiciels, données, procédures, humains, …) structurés pour acquérir, traiter, mémoriser, transmettre et rendre disponible l'information (sous forme de données, textes, sons, images, …) dans et entre les organisations.

Le schéma ci-dessous présente l'activité des constituants de l'entreprise :

Exemple de flux d'information



III. STRUCTURE PYRAMIDALE D'UNE ENTREPRISE



Intégration

1. Définir système d'information
2. Une organisation peut être vue comme un système pouvant être divisé en trois sous-systèmes, Citez-les.
3. Enumérer les activités de ces trois systèmes.

Solution

1. **Système d'information:** Ensemble des flux d'information circulant dans l'organisation.
2. Ces trois systèmes sont :
   - Système de pilotage ;
   - Système d'information ;
   - Système opérant
3. Les activités de ces trois systèmes.
   - Système de pilotage :
     - **Réfléchir** : adaptation à l'environnement, conception
     - **Décider** : prévisions, allocation, planification
     - **Contrôler** : qualité
   - Système d'information
     - **Générer des informations**
     - **Mémoriser**
     - **Diffuser**
     - **Traiter** (Système opérant)
     - **Transformer**
     - **Produire**

Exercice 1

L'entreprise de transport RAMIA EXPRESS décide de mettre sur pied système d'information.

1. Définir système d'information
2. Citer deux exemples de système d'information
3. Dans cette entreprise, quel sont les ressources appartenant au système opérant ?
4. Répondre par Vrai ou Faux
   a) Un système d'information permet à une entreprise de prendre des bonnes décisions…….
   b) Un système d'information permet à une entreprise d'augmenter sa production……
   c) Un système d'information permet d'améliorer les conditions de vie des employés au sein d'une entreprise……

Exercice 2

Votre établissement rencontre des difficultés concernant le traitement sur les dossiers portant sur les élèves et les enseignants. Face à cette difficulté, le chef de votre établissement décide de mettre sur pied un système d'information composé d'un ordinateur, des logiciels et d'un modem. En considérant votre établissement comme une entreprise qui dispense des cours aux élèves, Faites une représentation pyramidale de votre établissement.

Pré – Requis :

- *Définir système d'information*
- *Enumérer les activités de ces trois systèmes.*

Situation de vie

La société Njofruit's spécialiser dans la production et la vente des jus de fruit voudrait étudier le taux de consommation et d'écoulement de ces produits sur le marché. Mais ne sait comment s'y prendre, elle fait appel à vos connaissances dans ce domaine

Consigne :

a) ***Comment dois – tu procéder pour avoir l'avis des consommateurs?***
   - *Pour avoir ces informations, il faut procéder par des enquêtes sur le terrain.*

b) ***En supposant que tu as envoyé une équipe sur le terrain pour les enquêtes. Comment appelle – t – on cette actions ? et son résultat ?***
   - *Cette action se nomme la collecte des données et son résultat la donnée*

c) ***Après traitement de ce résultat (la donnée), comment appelle – t – on ce que tu auras à la fin?***
   - *Après traitement des données, on obtient une information.*

d) ***Enumérer les éléments qui t – ont permis de traiter ces données ?***
   - *Ces éléments sont : les ordinateurs équipés d'un logiciel de traitement de texte, des hommes pour les enquêtes et des supports de stockage pour conserver les résultats.*

Objectifs : A la fin de cette leçon, l'apprenant sera capable de mobiliser les ressources pour pouvoir:

- Décrire les systèmes d'information
- Enumérer les fonctions d'un SI

INTRODUCTION

Le SI peut être défini comme étant l'ensemble des flux d'information circulant dans l'organisation associé aux moyens mis en œuvre pour gérer les Infrastructures matérielles et logicielles (Réseau, Serveurs, Postes individuels, …, Progiciels, SGBD, Applications de gestion, Applications métier…) et les Moyens humains.

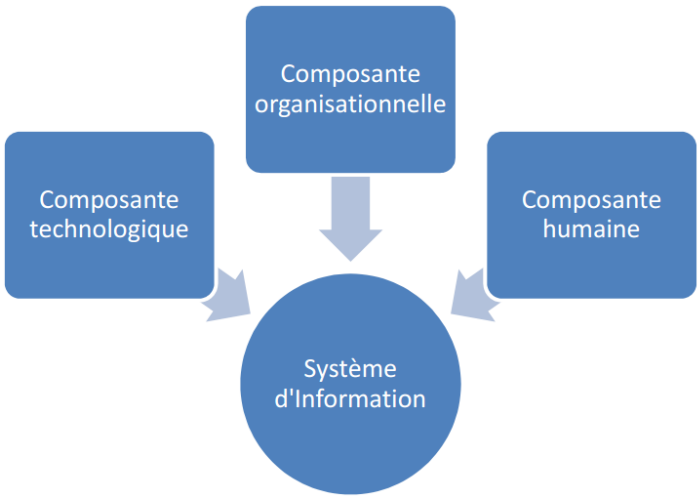
I. COMPOSANTS D'UN SYSTÈME D'INFORMATION

Le système d'information (SI) est une composante du système général de l'organisation comprenant un ensemble organisé de moyens humain et technique de façon à atteindre les objectifs décidés par l'organisation.

Le système d'information d'une organisation est structuré autour de trois composantes à savoir :

- La composante organisationnelle ;
- La composante matérielle ;
- La composantes technologique et ou logicielle.

Comme l'illustre le schéma ci - dessus



1. La composante Organisationnelle

La composante organisationnelle permet de définir la stratégie qui doit donner à chacun des acteurs les informations dont il a besoin pour effectuer de façon optimale les tâches qui lui incombent dans le cadre des objectifs que s'est fixée l'organisation.

2. Composante humaine

La composante humaine caractérisée par les acteurs internes ou externes à l'organisation qui animent le système d'information en formulant les besoins d'informations et les choix répondant à ces besoins ; avec l'objectif d'enrichir la connaissance de chacun donc du système d'information qui devient alors plus pertinent et plus cohérent.

3. La composante technologique ou logicielle

La composante technologique comprend l'ensemble des techniques et technologies d'information et de communication (TIC) permettant le recueil, la mémorisation, le traitement et la diffusion de l'information de façon à rendre le système d'information plus performant et plus efficace.

Remarque : *on pourra noter que la composante technologique (souvent le système informatique) ne représente le plus souvent qu'une partie du système d'information.*

II. RÔLE DU SYSTÈME D'INFORMATION

Le système doit assurer 4 fonctions essentielles :

- ✓ **Recueillir l'information** de façon cohérente, pertinente et la plus exhaustive possible (Enregistrer une information (support papier, informatique…) avant son traitement) ;
- ✓ **Stocker l'information** de façon à pouvoir la réutiliser directement en fonction des besoins c'est-à-dire : la Conservée, l'archivée (utilisation ultérieure ou obligation légale);
- ✓ **Traiter l'information** de façon à répondre aux diverses sollicitations des utilisateurs dans le cadre de leur activité ;
- ✓ **Diffuser l'information** de façon à permettre à chaque acteur de pouvoir exploiter les résultats des traitements dont il a besoin.

En bref, le système d'information doit mettre l'information au service de l'organisation. Sa matière première est l'information (un ensemble de données valorisées par des représentations concrètes définies par un sens). Il est perpétuellement en interaction avec le système de décision ou de pilotage et le système opérant ou opérationnel (production des objectifs fixés).

Enfin, le SI est structuré autour de ses composantes organisationnelles, humaines et technologiques de façon à assurer les fonctions permettant de recueillir, de stocker, de traiter et de diffuser l'information entre l'ensemble des acteurs et ainsi de pérenniser l'action de l'organisation.

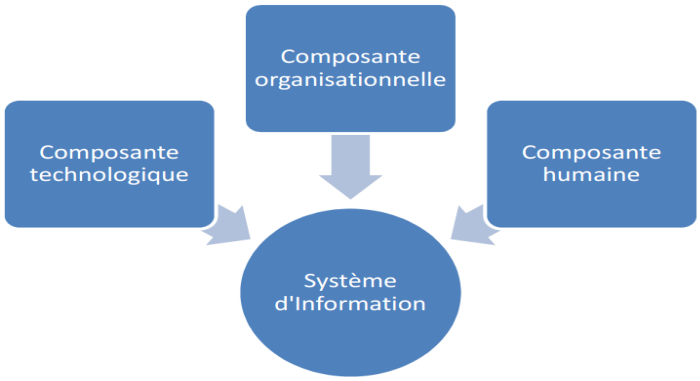
Ce schéma illustre le rôle d'un SI

Intégration

1. Donner la matière première d'un SI
2. Enumérer les rôles d'un SI.
3. Faites un schéma illustrant la structure d'un SI

Solution

1. **La** matière première d'un SI est l'information
2. **Le SI** à pour rôles:
   - ➢ Collecter l'information ;
   - ➢ Traiter l'information ;
   - ➢ Stocker l'information ;
   - ➢ Diffuser l'information.
3. Schéma illustrant la structure d'une information



Exercice

Votre établissement rencontre des difficultés concernant le traitement sur les dossiers portant sur les élèves et les enseignants. Face à cette difficulté, le chef de votre établissement décide de mettre sur pied un système d'information composé d'un ordinateur, des logiciels et d'un modem.

1. Donner le rôle d'un système d'information ?
2. En considérant votre établissement comme une entreprise qui dispense des cours aux élèves.
   a) Quel est Le rôle du système opérant au sein de cette entreprise ?
   b) Citer les ressources appartenant au système opérant de cette organisation ?

Pré – Requis :

- *Définir système informatique*
- *Donner le rôle d'un SI*
- *De quoi est composé un SI*

Situation de vie

La société Njofruit's après avoir recueillir les informations issues des enquêtes. Il désire les exploitées à d'autres fins. Mais le directeur de cette entreprise a besoin d'un professionnel pour cela. Il fait appel à vous

Consigne :

a) ***Donner l'objectif de cette enquête lancé par la société vers les clients?***

- *Pour développer de nouvelle stratégie d'écoulement des produits ;*
- *Connaître les clients;*

Objectifs : A la fin de cette leçon, l'apprenant sera capable de mobiliser les ressources pour pouvoir:

- Décrire les intérêts d'un SI
- Enumérer quelques méthodes de conception d'un SI

INTRODUCTION

Le système informatique[2] est aujourd'hui un des éléments vitaux des entreprises. Il est une composante essentielle de la gestion des informations appelée « ***Système d'information*** »

I. IMPORTANCE D'UN SI

Le système d'information est l'ensemble des actions coordonnées de recherche, de traitement, de distribution et protection des informations utiles. A la base de toutes les décisions, il met les technologies informatiques et les réseaux au service du contenu informationnel.

Les objectifs du système d'information sont :

- Identifier, collecter et diffuser les besoins d'informations des différentes activités ;
- Réduire les coûts de la collecte et du traitement des informations ;
- Actualiser les bases de données de l'entreprise ;
- Partager les informations entre les services et le personnel ;
- Rechercher et développer de nouvelles idées produites ;
- Connaître les clients d'un secteur d'activité donné ;

[2] Ensemble de moyens informatique et de télécommunications ayant pour finalité d'élaborer, traiter, stocker, acheminer, présenter ou détruire des données

- Connaître les réglementations en cours.

II. QUELQUES METHODES DE CONCEPTION D'UN SI

Plusieurs méthodes de conception de S.I. coexistent et sont exploitées différemment selon les pays, dont **Merise, UML, AXIAL, OMT, RUP ? IDEF**.

1. Méthode AXIAL

AXIAL (Analyse et Conception de Système d'Information Assistées par Logiciels) est une méthode d'analyse, de conception et de gestion de projet initialement utilisée dans le monde IBM. Issue de l'analyse systémique, la méthode AXIAL est une méthode qui a été concurrente de la méthode Merise.

2. Méthode OMT

(En anglais « Object Modeling Technique », c'est-à-dire « Technique de Modélisation Objet ») est une technique de modélisation destinée à la conception et la modélisation pour la programmation orientée objet.

3. Méthode RUP

La méthode RUP (Rational Unified Process), quant à elle, est une des émanations de la méthode **PU**[3], qui s'attache à donner un cadre précis au développement du logiciel. C'est une méthode générique, itérative et incrémentale assez lourde mais qui s'adapte très facilement aux processus et aux besoins du développement.

4. Méthode MERISE

La méthode Merise (Méthode d'étude et de réalisation informatique pour les systèmes d'entreprise) est née à la fin des années 1970 en France, avec pour objectif de définir une démarche de conception de S.I. Le principe de base repose sur la séparation des données et des traitements.

Cette approche définit l'entreprise comme une boite noire assurant une fonction de transformation de ressources. Dans un second temps, l'entreprise perçue en tant que système est considérée comme un ensemble organisé en vue d'une finalité dont les éléments sont en interaction permanente entre eux.

Intégration

1. Donner la signification des sigles suivant : AXIAL, OMT, MERISE, RUP

[3] Processus Unifié

2. Enumérer quatre intérêts des SI
3. Pourquoi dit – on que le système informatique est un élément vital de l'entreprise ?

Solution

1. Signification des sigles:

- **AXIAL** : Analyse et Conception de système d'Information Assistée par Logiciel
- **OMT** : Object Modeling Technique
- **MERISE** : Méthode d'Analyse et de Conception des Système d'Information
- **RUP** :Rationnal Unified Process

2. Quatre intérêts des SI :

- Actualiser les bases de données de l'entreprise ;
- Partager les informations entre les services et le personnel ;
- Rechercher et développer de nouvelles idées produites ;
- Connaître les clients d'un secteur d'activité donné ;

3. On que système informatique est un élément vital de l'entreprise parce qu'il est une composante essentielle de la gestion des informations.

MODULE 1 : ENVIRONNEMENT NUMERIQUE, SECURITE, MULTIMEDIA ET GESTION DES DONNEES

CHAPITRE 1 : SYSTEMES D'INFORMATION

Leçon 1 : Les concepts fondamentaux d'un système d'information

I. Indicateurs de compétence :

- ✓ Décrire les systèmes d'une entreprise (pilotage d'information, opérant) ;
- ✓ Enumérer les composants d'un SI (les ressources humaines, matérielles et logicielles)

II. Prérequis :

- ➢ Définir : système, traitement, Donnée, information, informatisation
- ➢ Quel est le rôle des périphériques d'entrée ? citer deux exemples
- ➢ Quel est le rôle des périphériques de sortie ? citer deux exemples
- ➢ Quel est le rôle de l'unité centrale ?

III. Situation problème

La société TAMPICO fabrique des jus de fruit au Cameroun. Comme matière première, elle utilise des fruits tels que la mangue, la goyave, l'ananas…

- ➢ Quels sont les éléments de l'entreprise qui s'occupent de la fabrication des jus de fruit ?
- ➢ Quels sont les éléments qui s'occupent de la gestion, du traitement, du transport et de la diffusion de l'information au sein de cette entreprise ?

IV. Résumé

Tout acte de la vie d'une organisation s'accompagne ou est conditionné par des informations pour améliorer son fonctionnement et faciliter la prise de décision. Ainsi une organisation (entreprise par exemple) peut être vue comme un système qui transforme les entrées en sorties. Ce système peut être divisé en deux sous-systèmes :

- ➢ Le système opérant constitué de la partie du système qui s'occupe effectivement de transformation des matières premières (les machines, les ouvriers, les techniciens, ...)
- ➢ Le système de pilotage qui définit les objectifs de l'entreprise et s'efforce de tout mettre en œuvre pour qu'ils soient atteints. Pour cela, il prend des décisions. Ces décisions sont prises à partir de paramètres venant du système opérant.

C'est le système d'information qui relève ces paramètres, les traites et les transmet au système de pilotage. Un système d'information est ensemble organisé de ressources (personnel, données, procédures, matériel, logiciel, etc.) permettant d'acquérir, de stocker, de structurer et de communiquer des

informations sous forme de textes, images, sons, ou de données codées dans des organisations.

Un système d'information peut être vue comme la partie qui relie les deux sous-systèmes précédents.

V. Question de cours

1. Définir système d'information
2. Une organisation peut être vu comme un système pouvant être divisé en deux sous-systèmes. Citez-les

VI. Situation d'intégration 1

Votre établissement rencontre des difficultés concernant la collecte, la sauvegarde et l'envoie des informations portant sur les élèves et les enseignant. Face à ces nombreuses difficultés, le chef de votre établissement décide de mettre sur pied un système d'information composé d'un ordinateur, des logiciels et d'un modem

1. Quel est le rôle d'un système d'information
2. En considérant votre établissement comme une entreprise qui dispense des cours aux élèves.
   a) Quel est Le rôle du système opérant au sein de cet entreprise ?
   b) Citer les ressources appartenant au système opérant de cet organisation ?
   c) Comment appelle-t-on l'autre partie du système ?

VII. Situation d'intégration 2

L'entreprise de transport DANAY EXPRESS décide de mettre sur pied système d'information.

1. Définir système d'information
2. Citer deux exemples de système d'information
3. Dans cette entreprise, quel sont les ressources appartenant au système opérant ?
4. Répondre par Vrai ou Faux
   a. Un système d'information permet à une entreprise de prendre des bonnes décisions
   b. Un système d'information permet à une entreprise d'augmenter sa production
   c. Un système d'information permet d'améliorer les conditions de vie des employés au seins d'une entreprise
   d. Un système d'information permet de collecter les informations au sein d'une entreprise, de les traiter et de les distribuer

CHAPITRE 6 : DESCRIPTION DES CONCEPTS DES BASES DE DONNÉES

Compétences visées : maitriser et décrire les concepts des Bases de données.

Situation problème : Généralement la consultation dans un hôpital se fait d'abord par l'achat du billet de session, qui oriente vers service dans lequel le malade doit se faire consulter. Dans ce service les infirmières ont un registre divisé en rubriques (date, numéros, noms et prénoms, sexe, domicile, plaintes) dans lequel elles prennent les informations concernant le malade avant de le faire passer chez le médecin. Toutes ces d'informations forment ce qu'on appelle la base de données des clients de cet hôpital. A partir de ce registre on peut rechercher les données sur un patient au cas où il n'aurait pas de billet de session lors d'une prochaine visite médicale? Le registre d'enregistrement des patients d'un hôpital est donc une base de données.

LEÇON 1: INTRODUCTION AUX BASES DE DONNEES

Objectif pédagogique opérationnel : A la fin de ce cours, l'élève doit être capable de définir et de donner le rôle des BD et des SGBD, décrire les fonctions de définitions, de manipulation d'une BD, décrire les éléments caractéristiques d'une table.

INTRODUCTION

La notion de Base de données peut être apparentée à une collection d'informations avec un objectif commun. On parle de Base de données lorsque les données sont rassemblées et stockées dans un support quelconque (papier, fichiers) et d'une manière organisée dans un but spécifique. Plusieurs éléments peuvent être assimilés aux Bases de données. (Le registre d'enregistrement des patients d'un hôpital, Le registre d'enregistrement des livres et des emprunts dans une bibliothe, etc). Peu importe le support utilisé pour rassembler et stocker les données (papier, fichiers, etc.), dès lors que des données sont rassemblées et stockées d'une manière organisée dans un but spécifique, on parle de base de données.

Plus précisément, on appelle base de données un ensemble structuré et organisé permettant le stockage de grandes quantités d'informations afin d'en faciliter l'exploitation (ajout, mise à jour, recherche de données). Bien entendu, dans le cadre de ce cours, nous nous intéressons aux bases de données informatisées. Une base de données informatisée permet l'accès aux données enregistrées sur des supports numérique par l'ordinateur et pouvant être interrogées et mises à jour par une communauté d'utilisateurs.

Exemple 1:

Le registre d'enregistrement des patients d'un hôpital est une base de données. Dans ce registre les données sont stockées de manière organisée. Il est divisé en plusieurs rubriques : Les dates pour séparer les journées. Les numéros pour identifier les patients, les noms et prénoms du patient, son sexe, son domicile, etc.

Exemple 2 :

La structure d'une bibliothèque s'apparente à une base de données. En effet, les livres qui représentent les informations sont rangés par thème dans des rayons ou étagères suivant un ordre bien précis. On peut : consulter, emprunter, ajouter, enlever ou supprimer un ou plusieurs livres.

I. DEFINITIONS

Une Donnée est un symbole ou convention aidant à la constitution d'une information.
Une Information est un ensemble de données qui peut être aussi définit comme la signification que l'on apporte à une données.
Une Base de Données ou encore (Data Base en anglais) est une entité dans laquelle il est possible de stocker des données de façon structurée et avec le moins de redondance possible, afin d'en faciliter l'exploitation. (Ajout, mise à jour, recherche des donnés).

II. Rôle d'une Base de Données

De manière générale, une base de données est utilisée pour enregistrer des faits, des opérations au sein d'une structure. Une base de données permet de mettre des données à la disposition des utilisateurs pour une consultation, une saisie ou une mise à jour tout en s'assurant des droits accordés à ces derniers. *Cela est d'autant plus utile que les données informatiques sont de plus en plus nombreuses. Un autre avantage de l'utilisation d'une base de données est l'accès simultané de plusieurs utilisateurs à la base.*

III. SYSTÈME DE GESTION DE BASE DE DONNEES (SGBD)

Pour mieux gérer les données et les utilisateurs dans une base de données, on fait appel à un système de gestion de base de données (SGBD). Un SGBD (Système de gestion de Base de données) appelé en anglais DBMS (Data Base Management System) est un logiciel qui permet de décrire, de manipuler, de traiter les ensembles de données formant la base de données. Le SGBD est un **serveur** de BD car c'est lui qui répond à toutes les questions (requêtes) concernant la BD. [c'est le SGBD qui crée la BD, effectue des opérations de recherche, de sélection, de mise à jour, de suppression dans la BD,etc.] **Un serveur** est un programme ou une machine qui offre des services à des clients en répondant à des demandes envoyées par ces derniers. Dans un réseau, on appelle serveur un ordinateur qui met ses ressources à la disposition d'autres ordinateurs (ordinateurs clients). *Un serveur web est un ordinateur connecté à Internet qui héberge des données et fichiers et qui gère les requêtes provenant des navigateurs des internautes.*

Quelques exemples de SGBD

Il existe de nombreux systèmes de gestion de bases de données, les plus utilisés sont :
Microsoft ACCESS, MySQL, Oracle, Sybase, DB2 de IBM, SQL Server

IV. LES PRINCIPALES CARACTERISTIQUES D'UN SGBD

Un bon SGBD doit avoir les caractéristiques suivantes :

- Manipulabilité (facilité d'emploi) : des personnes ne connaissant pas la base de données doivent être capables de décrire leurs requêtes sans faire référence à des éléments techniques de la base de données.
- Rapidité des accès : le système doit pouvoir fournir les réponses aux requêtes le plus rapidement possibles, cela implique des algorithmes de recherche rapide.
- Administration centralisée : le SGBD doit permettre à l'administrateur de pouvoir manipuler les données, insérer des éléments, vérifier son intégrité de façon centralisée.
- Non redondance : le SGBD doit pouvoir éviter dans la mesure du possible des informations redondantes, afin d'éviter d'une part un gaspillage d'espace mémoire mais aussi des erreurs.
- Partage des données : le SGBD doit permettre l'accès simultané à la base de données par plusieurs utilisateurs.
- Sécurité des données : le SGBD doit présenter des mécanismes permettant de gérer les droits d'accès aux données selon les utilisateurs.

Pour atteindre certains de ces objectifs (surtout les deux premiers), trois niveaux de description des données ont été définis :

Le niveau externe

On appelle cette description le schéma externe ou vue. Le niveau externe assure l'analyse et l'interprétation des requêtes en primitives de plus bas niveau et se charge également de

convertir éventuellement les données brutes, issues de la réponse à la requête, dans un format souhaité par l'utilisateur.

Le niveau conceptuel

Décrit la structure de toutes les données de la base, leurs propriétés (i.e. les relations qui existent entre elles : leur sémantique inhérente), sans se soucier de l'implémentation physique ni de la façon dont chaque groupe de travail voudra s'en servir

Le niveau interne ou physique

S'appuie sur un système de gestion de fichiers pour définir la politique de stockage ainsi que le placement des données.

V. LES MODELES(TYPES) DE SGBD

a) Le Modèle hiérarchique

C'est le premier modèle de SGBD. Ici, les données sont classées hiérarchiquement selon une structure arborescente descendante de façon à ce que chaque donnée n'ait qu'un seul processeur.

b) Le Modèle réseau

Il est presque semblable au modèle hiérarchique à la seule différence que sa structure n'est plus forcément arborescente dans le sens descendant.

c) Le Modèle relationnel SGBDR

Une base de données relationnelle est une base de données structurée suivant les principes de l'algèbre relationnel. Dans ce modèle, les données sont enregistrées dans un tableau (lignes et colonnes). C'est le modèle le plus utilisé. C'est avec ce modèle que nous allons travailler dans la suite du chapitre.

d) Le Modèle objet SGBDO

Les données sont stockées sous forme d'objets, de structures appelées classes présentant des données membres.

Les deux modèles de SGBD les plus répandus sont : les SGBD relationnel et les SGBD objet (viennent de paraitre. Ils sont plus puissant, plus proche de la réalité).

VI. FONCTIONS DE DEFINITIONS ET DE MANIPULATION D'UNE BASE DE DONNEES

Dans un SGBDR les données sont stockées de façon structurées dans des **tables** formées de **lignes** (tuples ou enregistrements) et de **colonnes** (champs, propriétés). Une table nommée *élève* peut se représenter de la façon suivante :

matricule	Nom	Prénom	Date_naiss	Lieu_naiss	sexe	Taille
20NP777	NONO	Boris	27/01/1990	Douala	M	1.77
20NP27	Fouda Etoundi	Marie	10/10/2000	Nkolbisson	F	1.70
20NP80	NFOR Ayuk	John	12/04/2002	Bamenda	M	1.65

1) Eléments caractéristiques d'une table

Une table est une structure de données permettant de stocker les données dans une base de données relationnelle.

Chaque table doit disposer d'un champs ou colonne qui doit permettre d'identifier de facon unique chaque enregistrement de la table : c'est la clé primaire. Dans l'exemple de la table ci-dessus, la clé primaire c'est le champ **matricule.** Si un champs clé primaire d'une table se retrouve dans une autre table comme simple champs, alors dans cette table est sera appelée **clé étrangère**. Les données à enregistrer doivent respecter un certain nombre de **contraintes d'intégrités.** Une Contrainte d'intégrité Une est une règle ou un ensemble de règles que doivent respecter les données pour assurer la cohérence de la base de données. Dans la définition d'une table, on peut indiquer des contraintes d'intégrités portant sur une ou plusieurs colonnes. Les contraintes possibles sont :

PRIMARY KEY (clé primaire**), UNIQUE** (interdit qu'une colonne (ou la concaténation de plusieurs colonnes) contienne deux valeurs identiques.), **FOREIGN KEY...REFERENCES (**clé étrangère**), CHECK(**condition que doit respecter la donnée**).** Toute définition de table doit comporter au moins une contrainte de type **PRIMARY KEY**, **NULL**(les données de cette colonne peuvent ne pas exister), **NOT NULL**(les données de cette colonne doivent toujours exister). Un attribut ou un champ, d'une table est clé étrangère dans une table s'il est clé primaire dans une autre table.

Les programmes utilisateurs doivent utiliser certaines fonctions pour pouvoir accéder aux données de la base de données. Ces fonctions peuvent êtres repartis en trois grandes catégories : les fonctions de définitions de données, les fonctions de manipulation des données, les fonctions de contrôles d'accès aux données.

2) Fonctions de définitions

Les opérations de définition des données sont celles qui permettent de créer une base de données, créer une table, modifient la structure d'une table (ajout/suppression des colonnes, renommer table/colonne, changer le type d'une colonne etc..). les opérations de définitions de données ne touchent pas les données contenues dans les lignes de la tables elles se limites à la structure de la table sans toutes fois entrer dans les données de la table.
Exemple : dans la table ci-dessus, l'opération de suppression de la colonne taille, nous renvoit une nouvelle table dont la structure est la suivante :

matricule	Nom	Prénom	Date_naiss	Lieu_naiss	sexe
20NP777	NONO	Boris	27/01/1990	Douala	M
20NP27	Fouda Etoundi	Marie	10/10/2000	Nkolbisson	F
20NP80	NFOR Ayuk	John	12/04/2002	Bamenda	M

3) Fonctions de manipulations

Les instructions de manipulations des données permettent d'effectuer sur les tables des manipulations telles que des insertions, des mises à jour, des suppressions et des interrogations de données.

a) L'interrogation des données ou sélection

Elle permet de parcourir une ou plusieurs tables en sélectionnant les lignes qui respectent une certaine condition donnée.
Exemple : dans la table ci-dessus on peut vouloir afficher le nom et le prenom des elèves de sexe masculin. Alors le résultat de cette interrogation sera une table ayant la configuration:

Nom	Prénom
NONO	Boris
NFOR Ayuk	John

b) L'insertion

L'opération d'insertion des données permet d'ajouter un enregistrement ou tuple ou ligne dans une table.
Exemple : Dans la table élève ci-dessus on peut vouloir ajouter l'élève Boukar Amidou Etienne de matricule NP2122 né à Garoua le 15-09-1999.

matricule	Nom	Prénom	Date_naiss	Lieu_naiss	sexe	Taille
20NP777	NONO	Boris	27/01/1990	Douala	M	1.77

20NP27	Fouda Etoundi	Marie	10/10/2000	Nkolbisson	F	1.70
20NP80	NFOR Ayuk	John	12/04/2002	Bamenda	M	1.65
20NP80	Boukar Amidou	Etienne	12/04/2002	Garoua	M	

NB : on peut déduire que le champs taille à la contrainte d'intégrité NULL. Car elle peut ne pas avoir de valeur. Si la colonne taille avait une contrainte NOT NULL, cette insertion ne serait pas possible sans introduire la taille de l'élève.

c) La Mise à Jour

L'opération de mise à jour permet de modifier les données dans les lignes existantes.
Exemple : Dans la table élève donné au debut du cours on peut vouloir modifier le l'année de naissance de Fouda en 11/10/2003. On aura donc une table

matricule	Nom	Prénom	Date_naiss	Lieu_naiss	sexe	Taille
20NP777	NONO	Boris	27/01/1990	Douala	M	1.77
20NP27	Fouda Etoundi	Marie	11/10/2003	Nkolbisson	F	1.70
20NP80	NFOR Ayuk	John	12/04/2002	Bamenda	M	1.65

d) La suppression

L'opération de suppression est utilisée pour supprimer une ou plusieurs lignes dans une table.

Exemple : Dans la table élève donné au debut du cours on peut vouloir supprimer tous les élèves ayant une taille inférieure ou égale à 1.70 le résultat qui sera obtenu est le suivant :

matricule	Nom	Prénom	Date_naiss	Lieu_naiss	sexe	Taille
20NP777	NONO	Boris	27/01/1990	Douala	M	1.77

4) Fonctions de control d'accès aux données

Les opérations de control d'accès aux données, permettent de retirer ou d'attribuer aux utilisateurs d'une base de données des privilèges sur l'utilisation des données. Par mesure de sécurité tous les utilisateurs ne doivent pas avoir le même niveau de visibilité dans le système.
Exemple dans le logiciel de gestion de votre établissement scolaire, on a plusieurs utilisateurs : les élèves, les enseignants, les surveillants généraux, les censeurs, l'intendant, le proviseur, etc.
Les élèves peuvent voire les notes sans toute fois les modifier.
Les enseignants peuvent en plus les modifier dans un délai donné sans toutes fois pouvoir modifier les heures d'absence des élèves.
Les surveillants généraux peuvent en plus modifier les heures d'absences.
Les censeurs peuvent modifier les notes en toutes périodes sans toutes fois avoir accès aux frais exigibles

CONCLUSION : Le logiciel qui manipule les bases de données est appelé système de gestion de base de données (SGBD). Il permet de créer, d'organiser, de contrôler, de consulter et de modifier la base de données. Les opérations de définitions et de manipulations des données sont parfois formulées dans un langage de requête structuré tel que **SQL** (Stuctured Query Language). L'étude de ce langage qui permettra d'écrire des requêtes , donner les instructions pour interroger les bases de données sera faite l'années prochaine en terminale.

Exercice1 : quel avantage présente une base de données informatisée sur une base de données traditionnelle ?

Réponse :

- Rapidité des accès aux données
- accès simultané par plusieurs utilisateur ce qui n'est pas le cas avec une BD traditionnelle.
- Capacité de stocker une très grande quantité d'information en minimisant au maximum l'encombrement. L'encombrement est un problème majeur dans les BD traditionelle lorsque le volume de données devient très important.

Exercice2 : quelle différence faite vous entre un fichier et une base de données ?

Réponse : (une base de données est composée de plusieurs fichiers virtuels appelé tables ; un fichier est très lié aux programmes qui permettent de le manipuler tandis qu'une base de données peut être manipulée par divers programmes; les données sont structurées et non redondantes dans une bd.)

Exercice3 : Dans l'exemple donné dans le cours de la table élève :

a) identifiez et nommez les champs de la table

b) donnez le nombre d'enregistrement de la table.

c) Peut on insérer dans cette table l'élève Boukar Oumarou réné né le 10-10-2000 à Maroua de matricule 20NP27 ? justifier.

MODULE 2 : ALGORITHME ET PROGRAMMATION WEB

Leçon 1 : Utiliser les structures algorithmiques

Compétences visées :

- Identifier les éléments d'un algorithme ;.
- Identifier les structures de contrôle ;
- Construire un organigramme ;
- Exécuter un algorithme ayant une structure alternative ;
- Exécuter un algorithme simple ;
- Exécuter des algorithmes itératifs.

Unité 1 : Notion d'algorithme

Contrôle de presrequis :

1. Enumerer les étapes de résolution d'un problème.

Compétences visées :

- Identifier les éléments d'un algorithme ;.
- Construire un organigramme ;

Situation probleme :

De nos jours, il n'est pas rare d'utiliser un navigateur GPS pour obtenir un itinéraire (but de l'algorithme). On entre alors le point de départ et le point d'arrivée (données d'entrée – 1 ère phase). Une série d'instructions (traitement des données – 2 ème phase) fournit en sortie une ligne brisée (résultat – 3 ème phase) qui symbolise le chemin à parcourir pour joindre ces deux points.

Mais comment écrire un algorithme pour qu'il soit universellement compréhensible ? Un algorithme peut être soit écrit sous forme littérale (langage algorithmique), soit représenté graphiquement (algorigramme).

Guide d'interactivité :

- **Qu'est-ce qu'un algorithme ?**

Réponse : un algorithme **u**ne suite finie et ordonnée d'opérations élémentaires donc l'exécution pas à pas permet de résolution un problème;

- **Quels sont les caractéristiques d'un algorithme ?**

Réponse : Un algorithme doit être simple, non ambigüe, compréhensible, ordonné, durable, lisible et finitude.

- **Donner sa structure générale.**

Réponse : un algorithme a 3 partie : l'entête, la partie déclarative et le corps.

- **Identifier les éléments dans un algorithme.**

Un algorithme comporte plusieurs éléments : les variables et constantes, les fonctions et procédures, les instructions(éciture, lecture, affectation, incrémentation/décrementation, les opérateurs (arithmétique, logique, bouléen, …), etc.

Résumé

1- DÉFINITION

Un algorithme est: une suite finie et ordonnée d'opérations élémentaires donc l'exécution pas à pas permet de résolution un problème;

Un algorithme est: une méthode de résolution systématique d'un problème pouvant être réalisé de façon mécanique;

L'algorithmique : est la science qui étudie les algorithmes.

Un programme : est la réalisation d'un algorithme dans un langage donné (proche de celle de la machine);

Un langage de programmation : est un langage destiné à décrire un ensemble d'actions consécutives qu'un ordinateur doit exécuter.

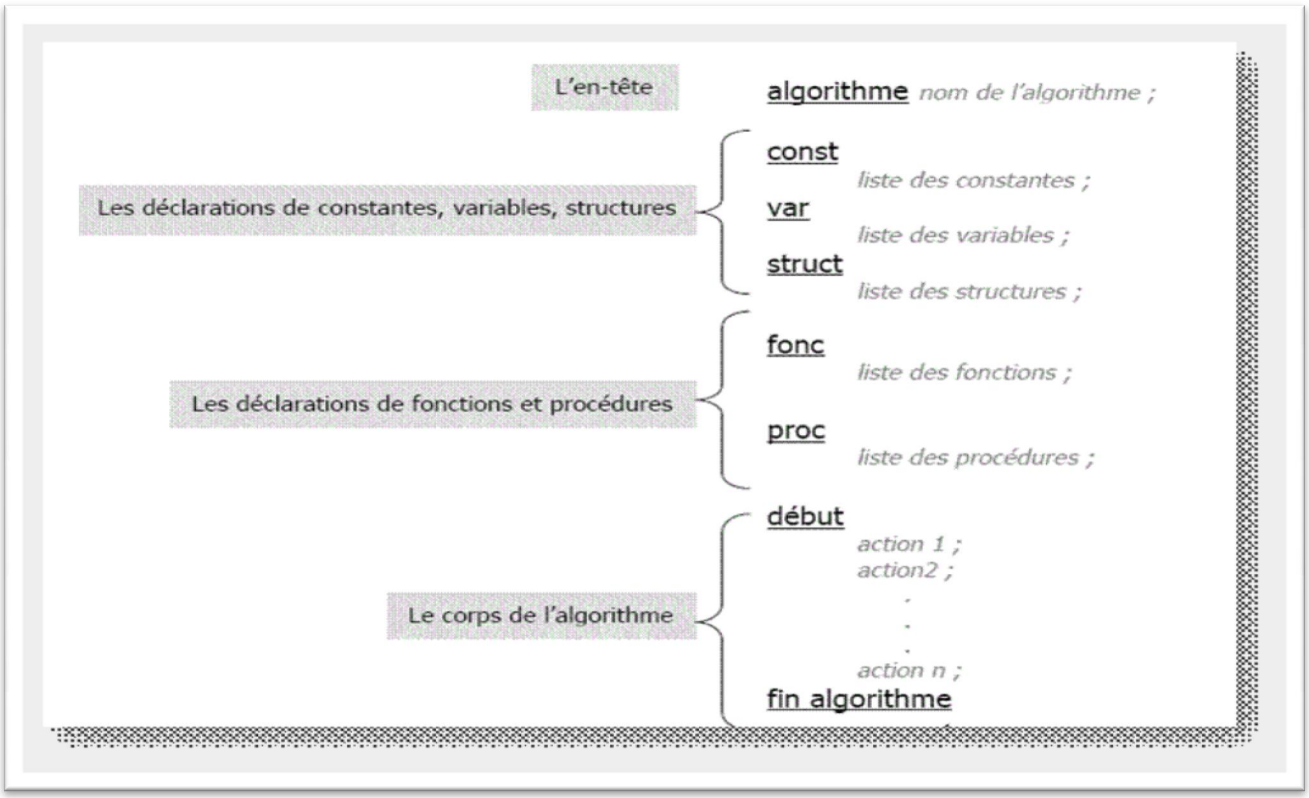
2- Caractéristiques d'un algorithme

Un algorithme doit être: simple, non ambigüe, compréhensible, ordonné, durable, lisible et finitude :

- **La finité:** la description des procédées doit être de longueur finie.
- **Le déterminisme:** un algorithme est dit déterministe si les étapes d'exécution sont bien fixés et ne conduisent à des choix aléatoires.
- **La terminaison (finitude):** l'algorithme doit produire la sortie souhaité avec un nombre fini d'étapes.
- **La généralité:** l'algorithme s'applique à tous les problèmes d'une même classe.
- **La correction d'un algorithme** signifie qu'il doit répondre au problème pour lequel il a été conçu.
- **La clarté (lisibilité)** d'un algorithme implique que le concepteur s'assure qu'il est facile à comprendre et à interpréter. Chaque opération doit être définie d'une manière précise.
- **La documentation** d'un algorithme consiste à l'insertion des connecteurs qui facilite la compréhension du programme.

NB : Un algorithme est dit efficace lorsque les opérations sont suffisamment simples et qu'elle s'exécute le plus rapidement possible.

3- La structure générale d'un algorithme



Tout comme le corps humain, un algorithme a trois parties:

- **L'entête:** Permet tout simplement d'identifier l'algorithme. L'algorithme peut prendre n'importe quel nom.
- **La partie déclarative:** Permet de déclarer tous les objets à utiliser dans le corps de l'algorithme: variable, constante, structure, fonction, procédure.
- **Le corps de l'algorithme:** Compris entre les mots ''début'' et ''fin'' contient les instructions, les délimiteurs, les opérations de traitements et les commentaires.

4- VARIABLES ET CONSTANTES

Une constante: est un objet ayant une valeur fixe tout au long d'un algorithme. En d'autres termes sa valeur ne change pas. Elle est caractérisée par deux éléments : son nom (identificateur) et sa valeur.

Une variable: est un objet pouvant prendre différentes valeurs tout au long d'un algorithme. Elle est caractérisée par trois éléments: son identificateur (nom), son type et son contenu.

Il existe **5 types de variable existent:** Entier (integer), réel (real), booléen (boolean), caractère (character), chaine de caractère (string).

5- INSTRUCTIONS

Les instructions sont les ordres de traitement respectant les actions simple dans l'exécution d'un algorithme.

Il existe plusieurs instructions :

- **L'instruction d'affectation**: Elle consiste à attribuer une valeur à une variable. On utilise le symbole ← qui signifie égal (=) en mathématiques.
  Exemple: a←2 signifie que a prend la valeur 2 ou 2 est affecté à a ou a=2.
- **L'instruction de sortie :** Elle consiste à écrire une donnée sur un périphérique de sortie tel que l'écran ou l'imprimante,… Elle se réduit au verbes: afficher (); ecrire (); writeln (); Sa syntaxe est: Ecrire (''entrer la valeur de a);
- **L'instruction d'entrée:** Une entrée consiste à introduire une donnée à partir des sources d'entrée (clavier, souris, crayon optique, …). Elle permet d'affecter à un objet en mémoire une valeur de même type que l'objet.
- Elle s'utilise par les mots ''lire ()'', ''saisir ()'', ''Readln ()''. Syntaxe : Saisir (a).
- **Les instructions d'incrémentation/ de décrémentation:** Elles se rencontrent dans les boucles. L'incrémentation peut être assimilé à un compteur qui à chaque cycle augmente d'une unité (1); La décrémentation est la diminution d'une unité à chaque cycle. Pour ce faire on utilise les variables du compteur (i ou j).

Syntaxe: i← i+1; j←j-1

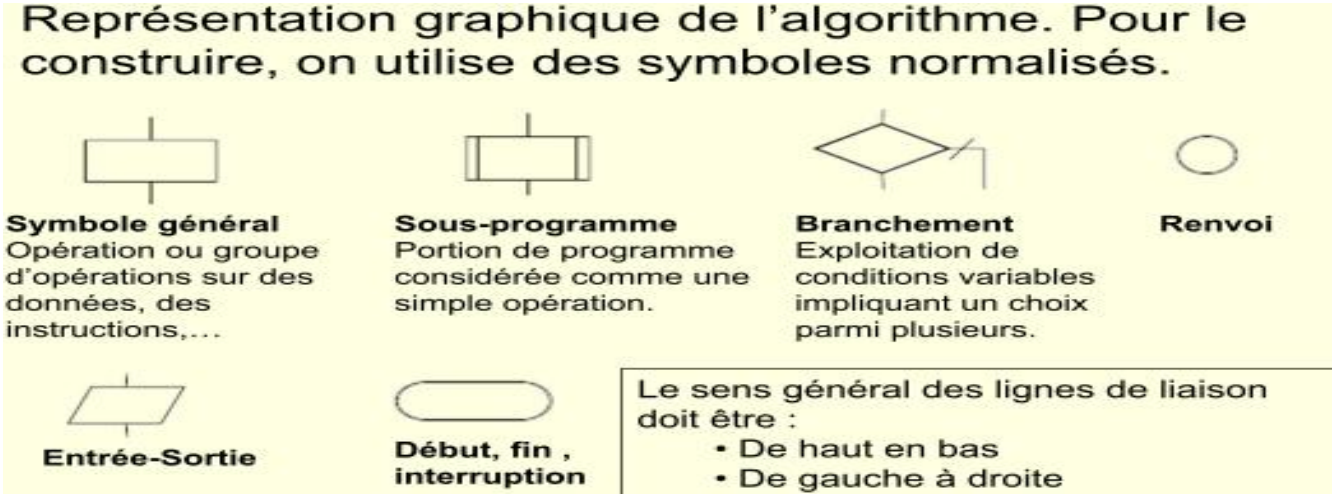
NB : Les trois premières instructions sont dites élémentaires.

6- LES OPÉRATEURS

On distingue plusieurs types d'opérateurs:

- **Les opérateurs arithmétiques:**
  - **Les opérateurs unaires ou** monodiques (appliqué à un seul opérande) : -27;
  - **Les opérateurs binaires ou diadique** (lient deux opérandes) **: 12+5;**
- **Les opérateurs logiques ou booléens**: ET, OU, NON.
- **Les opérateurs relationnels:**
  - Inférieur (<, <=); supérieur (>, >=);
  - égalité (=); différence (<>).

7- Organigramme



Activité d'intégration

Algorithme Plusgrand **Variable** a, b, pg : réel ; Début Ecrire ('entrer les deux valeurs') ; Lire (a, b) ; Si a>b alors Pg← a ; Sinon Pg← b ; FinSi Ecrire ('le plus grand nombre est :', pg) ; Fin	a) Que fait cet algorithme ? b) Combien de variables utilise cet algorithme ? donner leur type. c) Quelles sont les différentes instructions présentes dans cet algorithme ? Donner leurs types. d) Expliquer les éléments : « Ecrire ('entrer les deux nombres'), pg← a.

Solution

a) Cet algorithme détermine le plus grand entre deux nombres donnés.
b) Les variables utilisées par cet algorithme sont tous des réels : a, b, pg.
c) Il y a 3 instructions dans cet algorithme : lecture, écriture et affectation.
Il y a un seul opérateur logique : supérieur (>).
d) Les instructions :

- **Ecrire ('entrer les deux nombres')** signifie que l'utilisateur doit saisir deux nombres au clavier.
- **Pg←a**signifie que **pg** prend la valeur a

Réinvestissement

Algorithme Dagaarii Variable x,y, i : entier Début y ← 1 ; Ecrire ('entrer la valeur de x') ; Lire (x) ; Ecrire ('entrer la valeur de n') ; Lire (n) ; Pour i ← 1 à n faire y ← y*x ; i ← i+1 ; FinPour Ecrire (y) ; Fin	Dans cet algorithme : a) Donner le nombre d'instruction par type b) Identifier les différentes variables et leur type. c) Quels sont les opérateurs présents ici ? d) Quelle différence y a-t-il entre les instructions : Ecrire('entrer la valeur de n') et Ecrire (y)

Unité 2 : Structures algorithmiques (séquentielle et contrôle)

Contrôle de presrequis :

- Connaitre la structure générale d'un algorithme ;
- Identifier les instructions.

Compétences visées :

- Identifier les structures (contrôle et séquentielle);
- Construire un organigramme ;
- Connaitre quelques opérations mathématiques de base.

Situation problème

Écrire un algorithme permettant de résoudre une équation à coefficients réels de la forme ax+b = 0 (a et b seront entrés au clavier).

Guide d'interactivité

- **Lister les différentes structure de contrôle.**

Réponse attendue : structures alternatives ou conditionnelles, les structures itératives ou boucles.

- **Quelle structure convient-il ici ?**

Réponse attendue : structure alternative car il ya une condition sur le coefficiet a (a=0 ou a≠0)

- **Donner sa syntaxe**

Si (condition) alors
    (actions)
    sinon (actions)

RESUME

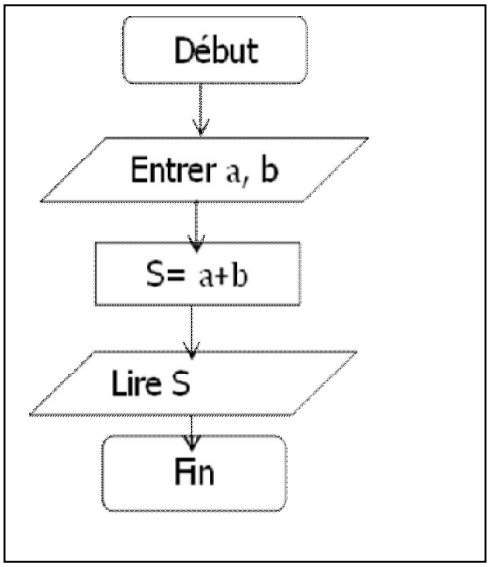
On distingue 3 structures algorithmiques de base: linéaire ou séquentielle, conditionnelle, itérative ou boucle.

1- **STRUCTURE LINÉAIRE OU SÉQUENTIELLE**

Dans cette structure, les actions se suivent, la fin d'une action déclenche le début d'une autre sans interruption. Une action se termine par un point virgule (;).

Exemple d'algorithme

- Algorithme somme
  - Variable a, b: réel
  - s: réel
- Début
  - Écrire (''entrer le premier nombre'');
  - Lire (a);
  - Écrire (''entrer le deuxième nombre'');
  - Lire (b);
  - s← a+b;
  - Écrire (s);
- Fin



2- STRUCTURES CONDITIONNELLES

Une structure conditionnelle est dite à forme alternative lorsque le traitement dépend d'une condition à deux états:

- Si la condition est évaluée à vrai, le premier traitement est exécuter;
- Si la condition est évaluée à faux, le 2ème traitement est exécuté.

Une structure conditionnelle est dite à choix lorsque le traitement dépend de la valeur que peut prendre le sélecteur. Ce sélecteur doit être du type scalaire (entier ou caractère);

Une structure conditionnelle est dite généralisée lorsqu'elle permet de résoudre les problèmes comportant plus de deux traitements en fonction des conditions. L'exécution d'un traitement entraine automatiquement la non exécution des autres (alternative multiple).

a) Structure conditionnelle à forme alternative

On distingue deux structures à forme alternative:

- L'alternative réduite:

Elle ne s'exécute que si la condition est satisfaisante.

syntaxe : si (condition) alors (instruction)

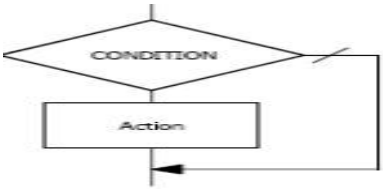
Exemple : Si x ≠ 0 alors s ← Ø

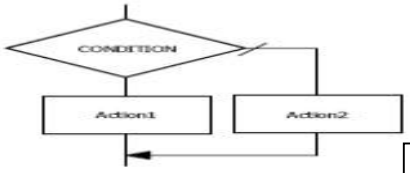
- **L'alternative complète**

Dans ce cas, une instruction est prévue pour chaque valeur logique de condition.

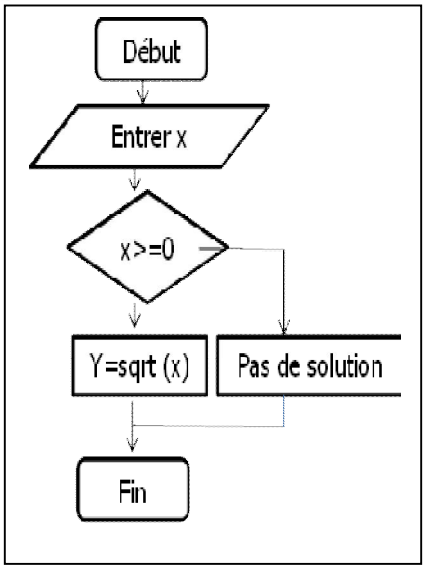
syntaxe : si (condition) alors (instruction) sinon (instruction)

Exemple: Si x ≠ 0 alors s ← b sinon s ← Ø





Exercice : racine carré d'un nombre



- Algorithme Squart
  - Var x, y: réel
- Début
  - Ecrire (''enter un nombre'');
  - Lire(x);
  - Si x >=0;
    - y← sqrt (x);
    - Ecrire (y);
  - Sinon
    - Ecrire (''pas de racine réelle'');
  - Finsi
- Fin

b) L'aternative multiple

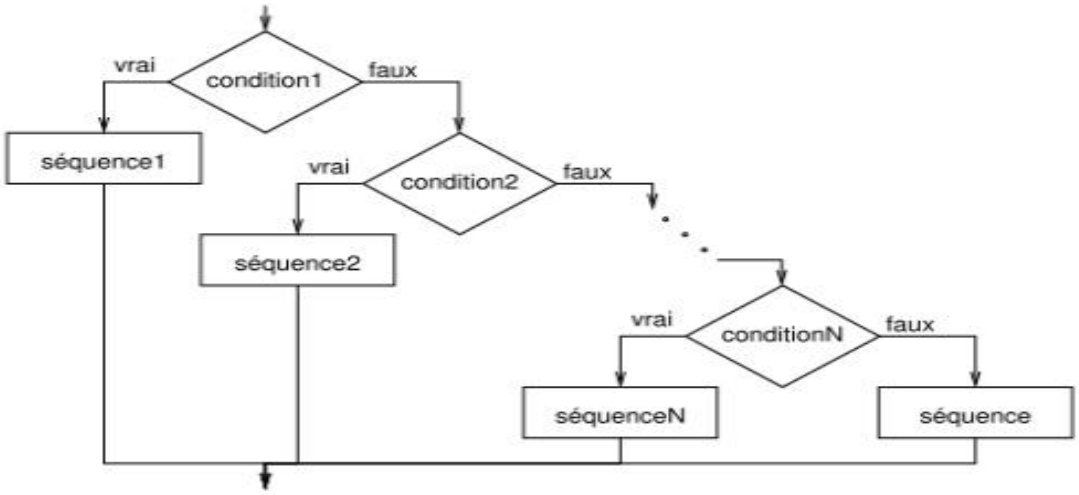
La conditionnelle Si ... Alors ... Sinon ... FinSi peut être complétée avec des clauses

Sa syntaxe est la suivante :

```
Si  condition1   Alors
    séquence1
    SinonSi  condition2   Alors
       séquence2
            ….
        SinonSi  conditionN   Alors
            séquenceN
            Sinon {  Expliciter  la  condition  !  }
               séquence
FinSi
```

Évaluation : Les conditions sont évaluées dans l'ordre d'apparition. Dès qu'une condition est vraie, la séquence associée est exécutée. L'instruction suivante à exécuter sera alors celle qui suit le FinSi. Si aucune condition n'est vérifiée, alors la séquence associée au Sinon, si elle existe, est exécutée.

Organigramme :



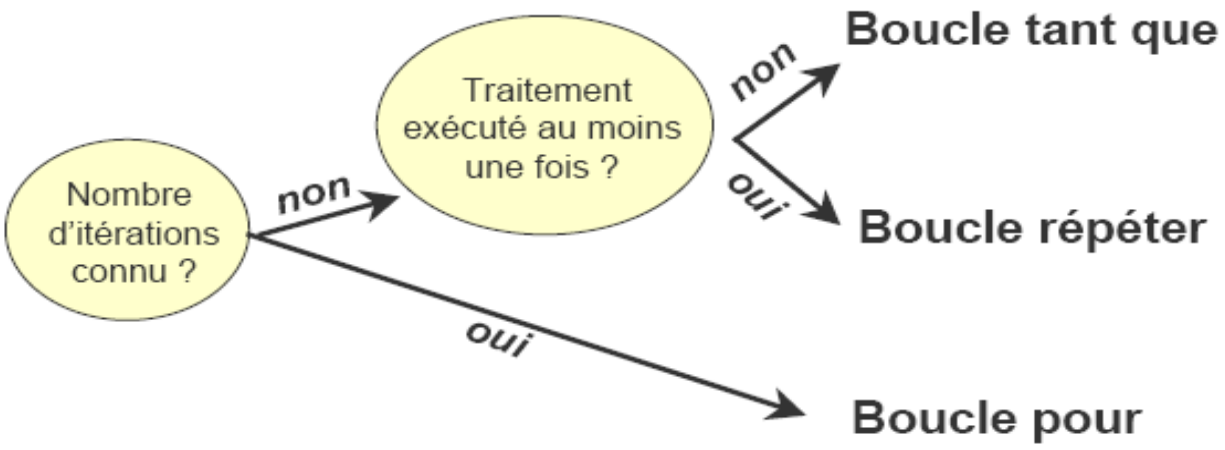
2- STRUCTURES ITÉRATIVES OU BOUCLES

a) Définition

Une itération ou boucle est une séquence d'instructions destinée à s'exécuter plusieurs fois.

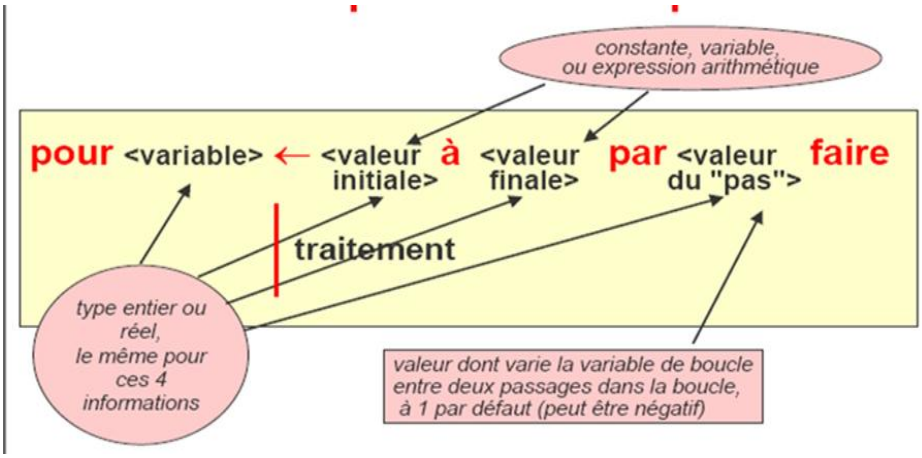
Un cycle est un tour de la boucle.

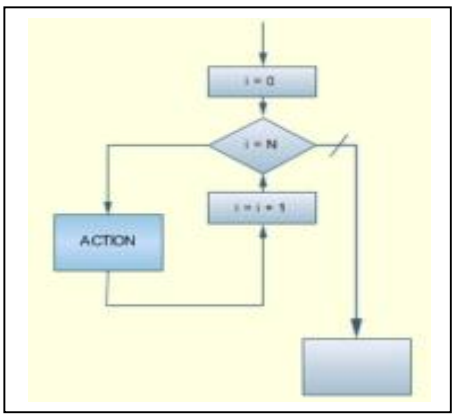
On distingue deux types d'itérations: itérative complète (pour) et itérations à condition d'arrêt (tant que, répéter).



➢ **Itération complète : la boucle pour**

- Permet de passer n fois dans une séquence d'instructions.
- Utilise le compteur;
- Le compteur incrémente la variable à chaque pas, vérifie que cette variable ne dépasse pas la borne supérieure.
- S'utilise lorsque le nombre d'itération à exécuter est bien connu.
- Sa syntaxe est Pour (compteur)←initial au final faire.



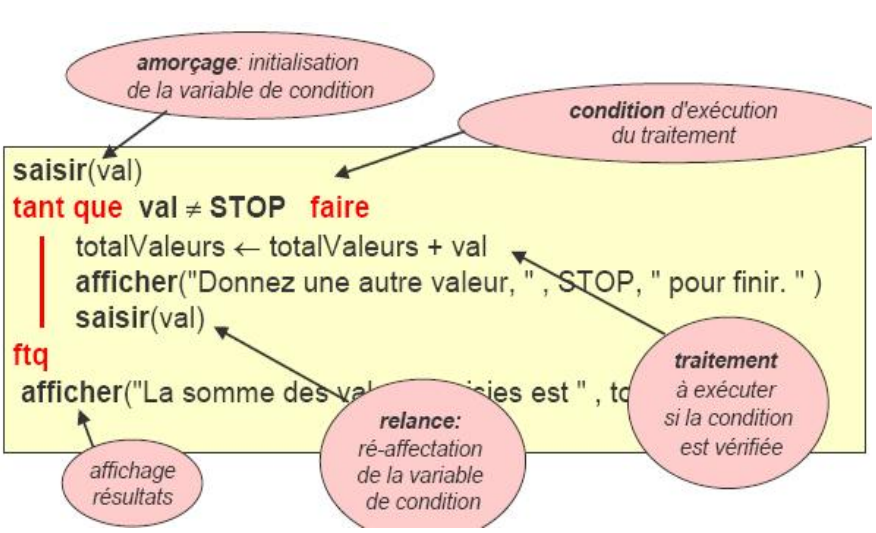


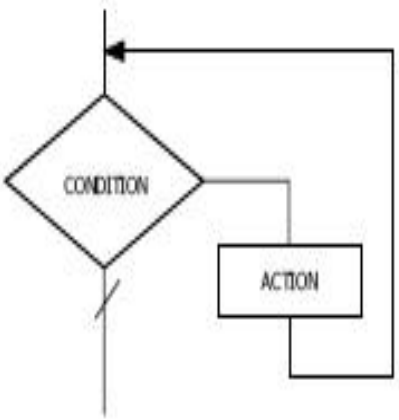
Exemple calculer la puissance d'un nombre

```
Algorithme puissance
  Var n, i : entier; x, p: réel
  Début
        P ←1;
       Pour i←1 à n faire
              p ← p*x
       finPour
  Fin
```

- **Itération à condition d'arrêt : La boucle Tant que**

- Est une structure dans laquelle la condition est d'abord testée et le traitement exécuté si la condition est remplie.
- S'utilise lorsqu'aucun traitement ne s'exécute avant la condition (le test) et le nombre d'itération n'est pas connu.
- Doit initialiser un compteur (amorçage), incrémenter le compteur à chaque pas (relance), vérifie que le compteur ne dépasse pas la borne supérieurs (test de boucle).
- Sa syntaxe est: tant que <condition> faire (instruction)





Exercice

Vous êtes dans une station de service de transport public à attendre le moyen de transport désiré. Le dernier arrive, le convoyeur procède à faire monter les voyageurs avec la condition qu'aucun ne doit voyager debout.

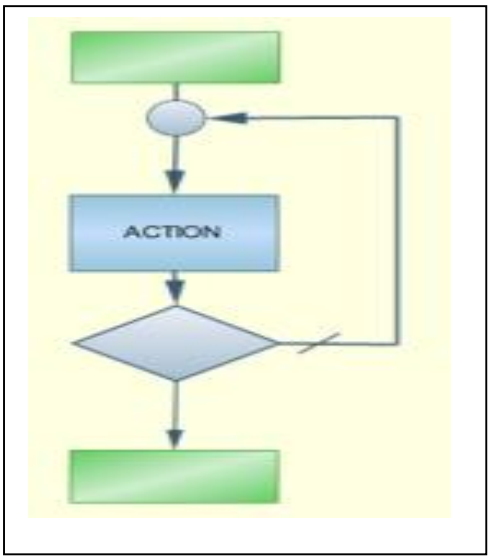
Solution

- Le convoyeur avertit le chauffeur d'ouvrir les portes s'il reste des places vacantes. Il commence l'action itérative suivante:
  - Tant qu'il ya encore de la place, faire monter un voyageur.
- La boucle peut être formulée ainsi:
- (arrêter le bus)
- tant que (il ya de place vacante) faire
  - monter le passager
- Fin tanque

➢ **Itération à condition d'arrêt : La boucle Répéter**

- Est une structure dans laquelle le traitement est exécuté une fois avant le test et le nombre d'itération n'est pas connu.
- Syntaxe: Répéter (instruction) jusqu'à (condition)
- Exemple: un algorithme qui calcule la moyenne des notes.

- Algorithme Moyenne
- Var s: réel; note: entier
- Début
- s←0;
- i←1;
- Répéter
- Ecrire (''donner la note n°:', i);
- Lire (note);
- s←s+note;
- i←i+1;
- Jusqu'à i>=10
- Ecrire (''la moyenne des 10 notes est:', s/10);
- Fin



ACTIVITE D'INTEGRATION

On souhaite écrire un algorithme qui demande un nombre de départ, et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 7) : Table de 7 : 7 x 1 = 7 ; 7 x 2 = 14 ; 7 x 3 = 21 ; …. ; 7 x 12 = 84 **TAF**

1) Identifier la structure algorithmique correspondante
2) Ecrire l'algorithme complète resolvant ce problème.

Solution

1) Structure itérative (plusieurs répétitions de la même action). Comme le nombre de répétition est connu d'avance, on utilisera l'itération complète (boucle Pour)
2) Algorithme

```
Algorithme TableMultiplication
  Variables n, i : Entier
  Debut
    Ecrire ("Entrez un nombre : " ) ;
    Lire (n) ;
    Ecrire( "La table de multiplication de ce nombre est : " ) ;
    Pour i ← 1 à 10
        Ecrire (n, " x ", i, " = ", n*i );
        i← i+1
  Fin
```

Reinvestissement

Exercice 1. Calcul du $n^{ième}$ nombre d'une suite

Écrire un algorithme permettant de calculer la $n^{ième}$ valeur d'une suite de la forme $U_n=aU_{n-1}+b$, $U_0 = c$ (a, b et c sont des entiers naturels entrés au clavier).

Exercice 2. Calcul d'antécédent par une fonction affine

Écrire un algorithme qui calcule, s'il existe, l'antécédent d'un nombre y par une fonction affine f(x) = ax + b (a et b donnés).

Exercice 3. Tracé de courbe d'un polynôme de degré 2

Écrire un algorithme permettant de tracer la courbe de la fonction polynôme de degré 2 $f(x)=ax^2+bx+c$ (a, b et c donnés).

Exercice 4. Lancer de pièces

Écrire un algorithme permettant de simuler n lancers de pièces, n donné, et d'afficher la fréquence de l'événement « la pièce tombe sur Pile ».

Unité 3 : EXÉCUTION D'UN ALGORITHME

Contrôle de presrequis :

- Connaitre les différentes structures algorithmes ;
- Connaitre les opérations mathématiques et logiques

Compétences visées :

- Exécuter un algorithme ayant une structure alternative ;
- Exécuter un algorithme simple ;
- Exécuter des algorithmes itératifs

Situation problème

Soit l'algorithme suivant:

```
Variable A, B, C : entier
    Debut
        A ← 5
        C ← 3
        B ← A+C
        A ← 2
      C ← B – A
    Fin
```

Guide d'interactivité

- **Identifier tous les éléments présents dans cet algorithme.**

Réponse attendue : les variables (A, B, C), l'instruction d'affectation, l'operateur de soustraction

- **Déterminer les valeurs de A, B et C**

Réponse attendue : A=5 au début et A=2 à la fin ; B= A+C=5+3=8 ; C=3 au début et C=B-A=8-2=6.

Résumé

Pour exécuter un algorithme, il suffit de conserver une trace des valeurs en cours des différentes variables et d'exécuter une à une les opérations qui composent l'algorithme (en respectant la sémantique des structures de contrôle) en reportant leur éventuel impact sur les valeurs des différentes variables.

1- ALGORITHMES ALTERNATIFS

Exemple 1 : Exécuter l'algorithme suivant avec a=-4 et b=-2

(1). Algorithme Plusgrand
(2). **Variable** a, b, pg : réel ;
(3). Début
(4). Ecrire ('entrer les deux valeurs') ;
(5). Lire (a, b) ;
(6). Si a>b alors
(7). Pg← a ;
(8). Sinon
(9). Pg← b ;
(10). FinSi
(11). Ecrire ('le plus grand nombre est :', pg) ;
(12). Fin

Solution

Ligne	Instruction	Variables			Valeur de la condition	Ligne suivante
		A	B	pg		
5	Lire (a,b)	-4	-2	x	……	6
6	Si a>b	-4	-2	x	faux	7
7	pg←a	-4	-2	x	….	8
8	Sinon	-4	-2	x	vrai	9
9	pg←b	-4	-2	-2	vrai	11
11	Ecrire (pg)	-4	-2	-2	vrai	Fin

2- ALGORITHMES ITÉRATIFS

Exemple 1 : Algorithme tant que

(1).Algorithme BABATA
(2).Var a, b, i : entier ;
(3).S, p : réel ;
(4).Début
(5).Ecrire (entrer deux nombres)
(6).Lire (a, b) ;
(7).S ←0 ; p←0 ; i←1 ;
(8).Tantque (i <=b) faire
(9).s←s+i ;p←p+a ;
(10). i←i+1 ;
(11). Fintanque
(12). Ecrire (S; P);
(13). Fin

Exécuter cet algorithme pour a=2, b=3

	test	a	b	S	p	i
6-7	i<=b	2	3	0	0	1
8-10	1<3	2	3	1+0=1	0+2=2	1+1=2
8-10	2<3	2	3	1+2= 3	2+2=4	2+1=3
8-10	3=3	2	3	3 +3=6	4+2=6	3+1=4
8-10	4> 3	/	/			
12	/	/	/	6	6	/

Exemple 2 : Exécution de l'algorithme Repeter

(1). Algorithme moyenne_age;
(2). Var i ,age,som : entier ;
(3). Debut
(4). i← 0 ; age← 0 ;
(5). Repeter
(6). Ecrire(' entrer l'âge du personne ') ;
(7). Lire('age') ;
(8). i← i+1 ;
(9). Som←som+ age ;
(10). Ecrire('Encore une autre personne (O/N) ?')
(11). Lire(rep) ;
(12). Jusqu'à (rep='N')
(13). Ecrire('la moyenne d'âges des personnes est : ',som/i) ;
(14). fin

Prendre les âges : 17, 18, 19 et 22 ans.

	I	Age	Som	test
4	0	0	0	
6-12	1	17	0+17=17	O
6-12	2	18	17+18=35	O
6-12	3	19	35+19=54	O
6-12	4	22	54+22=76	N
13			76/4=19	

Exemple 3 : Exécution de l'algorithme Pour

Tester cet algorithme avec n=4

```
Algorithme Sommenombre
Var n, s, i : entier
Début
   i←0 ; S←0 ;
   Ecrire ('entrer la valeur de n') ;
   Lire(n) ;
  Pour i←1 à n
        s←s+i ;
        i←i+1 ;
  FinPour
  Ecrire (n, i, s)
Fin
```

Tableau d'exécution de l'algorithme

i	n	s
0	4	0
1	4	0+1=1
2	4	1+2=3
3	4	3+3=6
4	4	6+4=10
4	4	10

ACTIVITE D'INTEGRATION

```
Algorithme Merci
Var   n, i, P : entier ;
Debut
 (1) Ecrire('Entrer la valeur de n ') ;
 (2) Lire(n ) ;
 (3)P ← 10;
     Pour i ← 1 à n Faire
 (4)       P ←P*i ;
 (5)        i ← i+1 ;
FinPour
(6) Ecrire(s) ;
 Fin
```

Pour n=4. Exécuter clairement cet algorithme en utilisant le tableau suivant.

Instructions	n	i	S
Ecrire, lire (n), S← 0	4	/	10
i← 1, S←S*i	4	1	10x1=10
i← 2, S←S*i	4	2	10x2=20
i← 3, S←S*i	4	3	20x3=60
i← 4, S←S*i	4	4	60x4=240
Ecrire (s)	/	/	240

Réinvestissement

Soit les algorithmes suivants :

```
Algorithme : Square
a, b, c, out : Entier
Début
  1) Lire (a)
  2) b ← a - 1
  3) c ← b x b
  4) b ← 2 x a - 1
  5) out ← c + b
  6) Afficher (out)
Fin
```

```
Algorithme Calcul
Var  R, i, n : entier
Debut
    Ecrire ('Donner un entier >0) ;
    Lire (n) ;
    R ← 1
    Pour i ← 1 à n faire
       R ← R*i
    Finpour ;
    Ecrire (R) ;
Fin
```

1) On affecte 4 à n dans lire(n) de l'algorithme calcul. combien de fois que la boucle POUR s'exécutera-t-elle ? quel sera alors le résultat R ?
2) Exécuter clairement cet algorithme Square avec a=3 et a=-3.

MODULE2 : ALGORITHMIQUE ET PROGRAMMATION WEB

CHAPITRE 8: UTILISATION DES FONCTIONS ET PROCEDURES

UNITÉ D'ENSEIGNEMENT: LES FONCTIONS ET LES PROCÉDURES

Pré requis :

- Définition algorithme
- Parties d'un algorithme
- Notion de variable
- Utilisation des structures algorithmiques
- Exécution d'un algorithme simple

Compétences attendues :

- Déclarer une fonction ou procédure
- Identifier le mode de passage d'un paramètre
- Distinguer une variable locale d'une variable globale
- Identifier une fonction ou procédure
- Exécuter un algorithme intégrant une fonction ou une procédure

Situation problème : on désire écrire un algorithme qui va déterminer pour deux nombres entiers s'ils sont positifs ou négatifs, puis afficher la plus petite valeur entre les deux et calculer la somme et la moyenne des deux nombres.

Comment procéder pour écrire cet algorithme ?

- Décomposer le problème en sous problèmes et trouver des solutions pour chacun
- Ecrire les sous algorithmes correspondants
- Réunir le tout dans un algorithme.

INTRODUCTION

Lors de l'écriture des algorithmes importants, il peut arriver qu'une suite d'actions revienne plusieurs fois ou encore on a des suites d'actions qui effectuent des traitements spécifiques différents ce qui entraine l'obtention de longs algorithmes. Il devient difficile ainsi d'avoir une vision globale de son fonctionnement et de détecter d'éventuelles erreurs. On peut alors définir ces suites d'actions sous forme de sous algorithmes (bloc faisant parti d'un algorithme). Il en existe deux types : les fonctions et les procédures.

I. DÉFINITION ET AVANTAGES

Une fonction est un sous algorithme qui effectue des traitements sur des données en entrée et renvoie un résultat unique à l'algorithme. La fonction n'affiche jamais de résultat à l'écran.

Une procédure est un sous algorithme qui effectue les traitements sur des données en entrée et affiche le ou les résultats obtenus.

Comme avantages des sous algorithmes on a :

- L'amélioration de la lisibilité des algorithmes
- La facilitation de la correction des erreurs
- L'optimisation du nombre d'instructions dans un algorithme

II. DÉCLARATION D'UNE FONCTION /PROCÉDURE

Pour pouvoir utiliser une fonction ou une procédure il faut au préalable la déclarer.
On identifie une fonction grâce au mot clé Fonction et une procédure grâce au mot clé Procédure.

Les procédures tout comme les fonctions ont trois parties qui sont :

- L'entête : on y retrouve le mot clé Fonction ou Procédure suivi du nom de la fonction ou de la procédure et ensuite les différents paramètres
- La partie déclaration : on y déclare les variables locales et les constantes
- Le corps : on y trouve les différentes instructions du sous algorithme

1. Déclaration d'une fonction

Entete : **Fonction** *nom_fonction* (Nom_parametre 1: type ;… ; Nom_parametre n : type) :**type_fonction**

Variablelocale :
Var // on déclare les variables et constantes
Nom_variable : type
...

Corps de la fonction :
Début
Instruction1 ;
Instruction 2 ;
….
Instruction n ;
Retourner résultat ;
Fin

Remarque :
- **Type_fonction** correspond au type du résultat renvoyé par la fonction
- **Retourner résultat** est l'instruction qui permet de récupérer le résultat de la fonction

Exemple : Déclaration d'une fonction qui permet de calculer la somme de deux nombres entiers.

Fonction *Somme* (Nbre1 : Entier ; Nbre2 : Entier) : **Entier** ;
Var Somme : Entier
Début
Somme= Nbre1+Nbre2 ;
Retourner Somme ;

Fin

2. Déclaration d'une procédure

Entete
Procédure *nom_procédure*(nom_parametre 1 :type ;….. ;Nom_parametre n :type)

Variablelocale
Var
Nom_variable : Type
….

Corps de la fonction
Début
Instruction1 ;
Instruction2 ;
…..
Instruction n ;
Fin

Exemple : Déclaration d'une procédure qui permet de calculer la somme de deux nombres entiers.

Procédure *Somme* (Nbre1 : Entier ; Nbre2 : Entier) ;
Var Somme : Entier
Début
Somme= Nbre1+Nbre2 ;
Ecrire (‘’ la somme est’’, Somme) ;
Fin

Remarque :

- La déclaration d'un sous algorithme ne suffit pas pour qu'elle soit utilisée, il faut l'appeler dans l'algorithme pour qu'elle soit effectivement utilisée.
- L'appel d'un sous algorithme utilise la syntaxe suivante : Nom_sous_algorithme (liste des paramètres)

Exemple :
Algorithme CalculSomme

Procédure *Somme* (Nbre1 : Entier ; Nbre2 : Entier) ;
Var Somme : Entier
Début
Somme= Nbre1+Nbre2 ;
Ecrire (‘’ la somme est’’, Somme) ;
Fin
Var N1, N2 : Entier ;
Début
Ecrire (‘’Entrez un premier entier’’) ;
Lire(N1) ;
Ecrire (‘’Entrez un deuxième entier’’) ;
Lire(N2) ;
Somme (N1 ;N2) ; // **appel de la procédure Somme**
FinAlgo

III. Portée des variables (Variable locale et variable globale)

L'endroit où est déclarée une variable est très important car il détermine dans quels sous programmes elle va pouvoir être utilisée.

Une variable locale est une variable qui est déclarée dans un sous algorithme et n'est visible que dans ce sous algorithme

Une variable globale est une variable qui est déclarée au niveau de la partie déclaration de l'algorithme et est visible dans tout l'algorithme et dans ses différents sous algorithmes.

Exemple :

Algorithme CalculSomme

```
Procédure  Somme (Nbre1 : Entier ; Nbre2 : Entier) ;
    Var Somme : Entier   // variable locale
    Début
      Somme= Nbre1+Nbre2 ;
      Ecrire (‘’ la somme est’’,  Somme) ;
    Fin
Var N1, N2 : Entier ;    // variable globale
Début
  Ecrire (‘’Entrez un premier entier’’) ;
  Lire(N1) ;
  Ecrire (‘’Entrez un deuxième entier’’) ;
  Lire(N2) ;
  Somme (N1 ; N2) ;
FinAlgo
```

IV. Mode de passage de paramètres

Il existe deux types de paramètres : **paramètre effectif** et **paramètre formel**

Un paramètre effectif est une valeur réelle reçue par le sous algorithme au cours de l'exécution de l'algorithme.

Exemple : Somme (N1 ; N2) ; N1 et N2 sont des paramètres effectifs.

Un paramètre formel est la définition du nombre et du type de valeurs que devra recevoir le sous algorithme pour s'exécuter correctement. On déclare les paramètres formels pendant la déclaration du sous algorithme.

Exemple :
Procédure Somme (Nbre1 : Entier ; Nbre2 : Entier) ;
Nbre1 et Nbre2 sont des paramètres formels

Il existe deux modes de passage de paramètres : par valeur et par référence (adresse ou variable)

1. **Passage de paramètres par valeur** : il y'a copie de la valeur des paramètres effectifs dans les variables locales issues des paramètres formels du sous algorithme appelé. Le contenu des paramètres effectifs n'est pas modifié par les instructions du sous algorithme car on travaille avec une copie de la variable.

Exemple :

Soit l'algorithme suivant :

```
Algorithme Param_valeur
 Procédure Val_Abs (n1 : Entier)
   Début
    Si (n1<0) Alors
      n1=-n1 ;
   Finsi
   Ecrire (n1) ;
   Fin
Var nbre : Entier ;
Début
  Lire (nbre) ;
  Val_Abs(nbre) ;
  Ecrire (nbre) ;
FinAlgo
```

Lors de l'exécution de cet algorithme avec la valeur -15, on a :

nbre	n1	Test	Ecran	
-15	Non déclaré			Avant l'appel de la procédure, on n'a que la variable nbre qui existe
-15	-15			A l'appel de la procédure, n1 est déclaré et reçoit une copie de nbre
-15	-15	vrai		On vérifie la condition
-15	15			
-15	15		15	
-15			-15	Retour à l'algo, il ne reste que la variable nbre et sa valeur initiale

2. **Passage de paramètres par référence** : ici on utilise l'emplacement mémoire de la variable c'est-à-dire que le paramètre formel se substitue au paramètre effectif durant le temps d'exécution du sous programme et lui transmet sa nouvelle valeur a la sortie. On se sert du mot clé **var**.

Exemple :

Soit l'algorithme suivant :

```
Algorithme Param_ref
   Procédure Val_Abs (Var n1 : Entier)
   Début
    Si (n1<0) Alors
      n1=-n1 ;
   Finsi
    Ecrire (n1) ;
  Fin
Var nbre : Entier ;
```

Début
Lire (nbre) ;
Val_Abs(nbre) ;
Ecrire (nbre) ;
FinAlgo

Lors de l'exécution de cet algorithme avec la valeur -15, on a :

nbre	N1	Test	Ecran	
-15	Non déclaré			Avant l'appel de la procédure, on n'a que la variable nbre qui existe
	-15			A l'appel de la procédure, n1 se substitue à la variable nbre
	-15	vrai		On vérifie la condition
	15			
	15		15	
15			15	Retour à l'algo, il ne reste que la variable nbre et sa valeur a changé

V. Exécution d'un algorithme intégrant un sous algorithme

Algorithme Exe_Moyenne
Fonction *Moyenne* (Var x1, x2 : Entier) : Réel ;
Var Moy : Réel ;
Début
Moy=(x1+x2)/2 ;
Retourner Moy ;
Fin
Var result : Réel
n1, n2 : Entier
Début
1. Ecrire (''Entrez le premier nombre'') ;
2. Lire (n1) ;
3. Ecrire (''Entrez le deuxième nombre'') ;
4. Lire (n2) ;
5. result= Moyenne (n1, n2) ;
6. Ecrire ('' la moyenne des deux nombres est '', result) ;
FinAlgo

	n1	n2	result	x1	x2	Moy	Ecran
2	14						
4	14	25					
5	14	25	19.5	14	25	19.5	
6							19.5

CONCLUSION :

Les fonctions et les procédures sont des éléments qui permettent d'améliorer la qualité des algorithmes. Cependant il faut veiller a les utiliser correctement afin de profiter de tous les avantages qu'elles offrent.

ACTIVITES DE CONSOLIDATION

1. Donner la syntaxe de déclaration d'une fonction et celle d'une procédure.
2. Ecrire une fonction qui calcule le PGDC de deux nombres entiers.
   a. Ecrire un algorithme qui va utiliser cette fonction
   b. Faire une exécution de cet algorithme avec les valeurs 15 et 9.
3. Ecrire une onction qui permet de calculer la $n^{ième}$ valeur d'une suite de la forme $U_n=aU_{n-1}+b$, $U_0=c$(c, a, b sont des entiers naturels passés en paramètres)
4. Ecrire une procédure qui permet de lire 20 nombres réels et les stockent dans un tableau.
5. On désire produire une calculatrice qui va effectuer les taches suivantes : addition, soustraction, multiplication, division, racine carrée, valeur absolue, le calcul de la puissance d'un nombre.
   a. Comment peut-on procéder pour obtenir un algorithme simple ?
   b. De combien de sous programmes a-t-on besoin ?
   c. Ecrire les codes correspondant à chacun des sous programmes (fonctions et/ou procédures)
   d. Ecrire l'algorithme qui va utiliser ces sous programmes.
      NB : on va effectuer un passage de paramètres par référence pour le sous algorithme qui calculer la puissance d'un nombre et pour les autres algorithmes on va utilise le passage de paramètres par valeur

MODULE 2 : ALGORITHMIQUE ET PROGRAMMATION

CHAPITRE 3 : PROGRAMMER EN HTML

PREREQUIS :

- Savoir utiliser un éditeur de texte et un navigateur
- Connaître la structure d'un code HTML
- Connaître les notions de balise, d'attribut et de valeur

COMPETENCES ATTENDUES :

- Insérer un tableau dans une page web
- Insérer des fichiers multimédia (image/son/vidéo) dans une page web
- Créer un formulaire
- Organiser le dossier du site
- Décrire la contribution du CSS.

SITUATION PROBLEME :

Votre papa président d'une association nationale, veut créer un site Web dans lequel il présentera dans une page Web statique, le bilan de son association sous forme d'un tableau afin que ce bilan soit vu par tous les membres dans les quatre coins du pays. Votre papa ne s'y connaissant pas en programmation Web, vous demande de l'aide.

De quels outils aurez-vous besoin ? Quel langage de programmation utiliserez-vous ? Comment procéder pour réaliser cette page Web ?

3.1. LES TABLEAUX

3.1.1. UTILISATION DE TABLEAUX

Les tableaux permettent de présenter les informations en ligne et en colonnes. Ils sont définis comme étant des suites de lignes. Un tableau doit respecter les règles suivantes :

- Le tableau est encadré par la balise <TABLE> et </TABLE>

- Le titre du tableau est encadré par <CAPTION> et </CAPTION>
- Chaque ligne est encadrée par <TR> et >/TR> (Table Row : ligne du tableau)
- Les cellules d'en-tête sont encadrées par <TH> et </TH> (Table Header : En-tête de tableau)
- Les cellules de valeur sont encadrées par <TD> et </TD> (Table Data)

Exemple de tableau HTML :

Soit à réaliser le tableau ci-dessous

Produits vivriers

Fruits	Tubercules	Légumes
Ananas	Manioc	Carotte
Banane	Tarot	Haricot

Voici le code HTML qui permet d'afficher ce tableau :

```
<table border="1">
    <caption>Produits vivriers</caption>
    <tr>
        <th>Fruits</th>
        <th>Tubercules</th>
        <th>Légumes</th>
    </tr>
    <tr>
        <td>Ananas</td>
        <td>Manioc</td>
        <td>Carotte</td>
    </tr>
    <tr>
        <td>Banane</td>
        <td>Tarot</td>
        <td>Haricot</td>
    </tr>
</table>
```

3.1.2. LES ATTRIBUTS

Attribut	Balises auxquelles Il s'applique	Valeur	Effet visuel
ALIGN	THEAD TBODY TH TR TD	CENTER LEFT RIGHT JUSTIFY	Centré Gauche Droite Justifié

	CAPTION	TOP BOTTOM	Au-dessus En-dessous
VALIGN (alignement vertical)	THEAD TBODY TH TR TD	TOP MIDDLE BOTTOM	En haut Au milieu En bas
BORDER	TABLE	n (c'est un nombre)	Taille de la bordure
CELLPADDING	TABLE	n (c'est un nombre)	Espacement de n pixels entre le contenu des cellules et la bordure
CELLSPACING	TABLE	n (c'est un nombre)	Epaisseur de la grille intérieure
FLOAT	TABLE	RIGHT LEFT	Position du texte qui suivra
COLS	TABLE	n (c'est un nombre)	Nombre de colonnes
FRAME (contrôle les éléments individuels d'encadrement du tableau	TABLE	NONE TOP BOTTOM TOPBOT SIDES ALL	Aucun Au-dessus En bas Tout en haut Sur les côtés Tous
RULES (contrôle les éléments de la grille des cellules)	TABLE	NONE BASIC ROWS COLS ALL	Aucun Basique Ligne Colonne Tous
COLSPAN	THEAD ; TBODY ; TH ; TR ; RD		Débordement des cellules sur les colonnes adjacentes

ROWSPAN	THEAD ; TBODY ; TH ; TR ; RD		Débordement des cellules sur les lignes adjacentes

Exercice d'application :

Ecrire le code html du tableau ci-dessous :

Habitants du Cameroun

Nom	Prénom	Age	Ville
ILOGA	David	32	Edéa
DONFACK	Flaubert	24	DSCHANG
HAMIDOU	Issa	53	MEIGANGA

3.2. LES FICHIERS MULTIMEDIA

3.2.1. LES IMAGES

Les 3 principaux formats d'images reconnus par les navigateurs sont le JPEG, le PNG et le GIF.

- Le **JPEG** (ou **JPG**) n'est limité qu'à 16 millions de couleurs mais ne gère pas la transparence.
- Le **PNG** quant à lui est plus récent et combine les avantages du **JPG** avec une gestion fine de la transparence (translucidité) mais cette dernière propriété n'est pas reconnue par tous les navigateurs.
- Le **GIF** est limité à 256 couleurs et permet de gérer la transparence.

Remarque : Il est très important de prendre en compte le poids des fichiers images. Cela influe sur le temps de téléchargement de la page. Il faut donc :

- Proposer des dimensions respectables d'une part ;
- Veiller d'autre part lors de la compression, à ne pas dépasser un certain seuil (30 à 100 Ko selon l'objectif en image)

La balise d'image : **<img>** c'est une des rares balises à être non fermante. Cette balise prend au moins 2 attributs :

- L'attribut **src** : qui prend pour valeur la source c'est-à-dire l'adresse vers le fichier image

- L'attribut indispensable est **alt** : il définit un texte alternatif, qui s'affiche lorsque l'image n'est pas encore téléchargée ou non visible par le visiteur ou le navigateur.

Exemple d'insertion d'une image :

<img src= "moi.jpg" alt="Ma photo à la piscine olympique de la cité-sic">

Exercice d'application :

Soit une image avec pour nom « bobo.gif » se trouvant dans un ordinateur à l'adresse « C:/Document/image »

a) Quel est le format et le nom de cette image ?
b) Donner la syntaxe permettant d'insérer cette image dans un document Web nommé « accueil.html »
c) Modifier cette syntaxe de sorte que le message « non accessible » s'affiche lorsque l'image n'est pas chargée dans la page
d) Ecrire la syntaxe permettant d'établir un lien entre cette image et le document nommé « page2.html »

3.2.2. L'AUDIO ET LA VIDEO

En HTML5, les balises utilisées pour insérer l'audio et la vidéo sont respectivement et <audio> <video>. Les principaux attributs utilisés sont :

- **src** : qui prend pour valeur la source c'est-à-dire l'adresse vers le fichier image
- **controls** : pour ajouter les boutons « Lecture », « Pause » et la barre de défilement. Cela peut sembler indispensable.

Exemple d'insertion d'un élément audio et d'une vidéo :

<audio src="x-maleya_hola-me.mp3" controls> </audio>

<video src="salatiel_anita.mp4" controls> </video>

3.3. LES FORMULAIRES

3.3.1. LA BALISE FORM

Les formulaires interactifs permettent aux auteurs de pages web de doter leur page web d'éléments interactifs permettant par exemple un dialogue avec les internautes, à la recherche des coupons réponses présents dans certains magazines.

Les formulaires sont délimités par la balise <**FORM**>…</**FORM**>, une balise qui permet de regrouper plusieurs éléments de formulaire (boutons, champ de saisie, …) et qui possède les attributs obligatoires suivants :

- **METHOD** : indique sous quelle forme seront envoyées les réponses. « **POST** » est la valeur qui correspond à un envoi de données stockées dans le corps de la requête, tandis que « **GET** » correspond à un envoi des données codées dans **l'URL** et séparées de l'adresse du script par un point d'interrogation
- **ACTION** : indique l'adresse d'envoi (script CGI ou adresse email)

Syntaxe de la balise FORM :

<form method="post" action="url"

…

</form>

3.3.2. A L'INTERIEUR DE LA BALISE FORM

La balise FORM constitue en quelque sorte un conteneur permettant de regrouper des éléments qui vont permettre à l'utilisateur de choisir ou de saisir des données, ensemble de données qui seront envoyées à l'URL indiquée dans l'attribut ACTION de la balise FORM par la méthode indiquée par l'attribut METHOD.

Il est possible d'insérer n'importe quel élément HTML de base dans une balise FORM (textes, boutons, tableaux, liens, …) mais il est surtout intéressant d'insérer des éléments interactifs. Ces éléments interactifs sont :

- La balise INPUT : un ensemble de boutons et de champ de saisie
- La balise TEXTAREA : une zone de saisie multi lignes
- La balise SELECT : une liste à choix multiples

3.3.2.1. LA BALISE INPUT

La balise INPUT est la balise essentielle des formulaires, car elle permet de créer un bon nombre d'éléments "interactifs". La syntaxe de cette balise est la suivante :

<input type="nom du champ" value="valeur par défaut" name="nom de l'élément" />

L'attribut name est essentiel car il permettra au script CGI de connaître le champ associé à la paire nom/valeur c'est-à-dire que le nom du champ sera suivi du caractère "=" puis de la valeur entrée par l'utilisateur, ou dans le cas contraire de la valeur par défaut repérée par l'attribut value.

L'attribut type permet de préciser le type d'éléments que représente la balise INPUT, voici les valeurs que ce champ peut prendre :

- Checkbox : il s'agit de cases à cocher pouvant admettre 2 états : checked (coché) et unchecked (non coché). Lorsque la case est cochée la paire nom/valeur est envoyée au CGI
- Hidden : il s'agit d'un champ caché. Ce champ non visible sur le formulaire permet de préciser un paramètre fixe qui sera envoyé au CGI sous forme de paire nom/valeur
- File : il s'agit d'un champ permettant à l'utilisateur de préciser l'emplacement d'un fichier qui sera envoyé avec le formulaire. Il faut dans ce cas préciser le type de données pouvant être envoyées grâce à l'attribut ACCEPT de la balise FORM
- Image : il s'agit d'un bouton de soumission personnalisé, dont l'apparence est l'image située à l'emplacement précisé par son attribut src
- Password : il s'agit d'un champ de saisie, dans lequel les caractères saisis apparaissent sous forme d'astérisques afin de camoufler la saisie de l'utilisateur
- Radio : il s'agit d'un bouton permettant un choix parmi plusieurs proposés (l'ensemble des boutons radios devant porter le même attribut name. La paire nom/valeur du bouton radio sélectionné sera envoyée au CGI. Un attribut checked pour un des boutons permet de préciser le bouton sélectionné par défaut
- Reset : il s'agit d'un bouton de remise à zéro permettant uniquement de rétablir l'ensemble des éléments du formulaire à leurs valeurs par défaut
- Submit : il s'agit du bouton de soumission permettant l'envoi du formulaire. Le texte du bouton peut être précisé grâce à l'attribut value
- Text : il s'agit d'un champ de saisie permettant la saisie d'une ligne de texte. La taille du champ peut être définie à l'aide de l'attribut size et la taille maximale du texte saisie grâce à l'attribut maxlength.

3.3.2.2. LA BALISE TEXTAREA

La balise TEXTAREA permet de définir une zone de saisie plus vaste par rapport à la simple ligne de saisie que propose la balise INPUT. Cette balise possède les attributs suivants :

- cols : représente le nombre de caractères que peut contenir une ligne
- rows : représente le nombre de lignes
- name : représente le nom associé au champ, c'est le nom qui permettra d'identifier le champ dans la paire nom/valeur
- readonly : permet d'empêcher l'utilisateur de modifier le texte entré par défaut dans le champ
- value : représente la valeur qui sera envoyée par défaut au script si le champ de saisie n'est pas modifié par une frappe de l'utilisateur

3.3.2.3. LA BALISE SELECT

La balise SELECT permet de créer une liste déroulante d'éléments (précisée par des balises OPTION à l'intérieur de celle-ci). Les attributs de cette balise sont :

- name : représente le nom associé au champ, c'est le nom qui permettra d'identifier le champ dans la paire nom/valeur
- disabled : permet de créer une liste désactivée, c'est-à-dire affichée en grisée
- size : représente le nombre de lignes dans la liste (cette valeur peut être plus grande que le nombre d'éléments effectifs dans la liste)
- multiple : marque la possibilité pour l'utilisateur de choisir plusieurs champs sur la liste.

Exemple de formulaire

Les formulaires peuvent être mis en page à l'aide de tableaux (cela est même conseillé pour avoir une mise en page soignée). Voici un exemple récapitulant les points ci-dessus et montrant comment mettre en page un formulaire à l'aide d'un tableau :

```
<form method="post" action="traitement.php">

    Enregistrement d'un utilisateur

    <table border="0">

        <tr>

            <td>Nom</td>
```

```
        <td> <input type="text" name="nom"> </td>
</tr>
<tr>
        <td>Prénom</td>
        <td> <input type="text" name="prenom"> </td>
</tr>
<tr>
        <td>Sexe</td>
        <td>
                Homme : <input type="radio" name="sexe" value="M"> <br>
                Femme : <input type="radio" name="sexe" value="F">
        </td>
</tr>
<tr>
        <td>Fonction</td>
        <td>
                <select name="Fonction">
                        <option value="enseignant">Enseignant</option>
                        <option value="etudiant">Etudiant</option>
                        <option value="ingenieur">Ingénieur</option>
                        <option value="retraite">Retraité</option>
                        <option value="autre">Autre</option>
                </select>
        </td>
```

```
            </tr>
            <tr>
                <td>Commentaires</td>
                <td>
                    <textarea rows="3" name="commentaires">Tapez ici vos commentaires</textarea>
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    <input type="submit" value="Envoyer">
                </td>
            </tr>
        </table>
</form>
```

Voici le résultat de ce code HTML

Enregistrement d'un utilisateur

Nom	
Prénom	
Sexe	Homme : ○ Femme : ○
Fonction	Enseignant
Commentaires	Tapez ici vos commentaires

Envoyer

Exercice d'application

Nom :	
Téléphone :	
Adresse :	

1. Ecrire le code HTML permettant d'obtenir le code ci-dessus
2. Puis, écrire le code pour insérer un bouton avec pour nom « Envoyer » permettant de soumettre ce formulaire
3. Et enfin, écrire le code pour insérer un bouton avec pour nom « Annuler » permettant d'initialiser le formulaire

3.4. ORGANISER LE DOSSIER DU SITE

Il est conseillé dans la mesure du possible, de mettre le dossier principal du site 3 éléments :

- **Le fichier index.html :** le nom à donner à votre page d'accueil sera imposé par votre fournisseur d'accès. Le plus souvent ce sera « index.htm » ou « index.html ». Il s'agit de la première page de votre site que l'utilisateur verra. C'est la seule page dont le fournisseur d'accès a besoin de connaître le nom. Les autres pages ainsi que les images sont organisées selon votre souhait.
- **Un sous-dossier nommé par exemple « pages » :** qui contiendra les autres pages de votre site
- **Un sous-dossier nommé par exemple images :** qui contiendra les images de votre site

3.5. DECRIRE LA CONTRIBUTION DU CSS

Le CSS a été mis au point afin de compenser les manquements du langage HTML en ce qui concerne la mise en page et la présentation. En effet, le HTML offre un certain nombre de balises permettant de mettre en page et de définir le style d'un texte, toutefois chaque élément possède son propre style, indépendamment des éléments qui l'entourent. Grâce aux feuilles de style, lorsque la charte graphique d'un site composé de plusieurs centaines de pages Web doit être changée, il suffit de modifier la définition des feuilles de style en un seul endroit pour changer l'apparence du site tout entier ! Elles sont appelées « feuilles de style en cascade » (en anglais : Cascading Style Sheets » car il est possible d'en définir plusieurs et que les styles peuvent être hérités en cascade.

Les feuilles de style permettent notamment :

- D'obtenir une présentation homogène sur tout un site en faisant appel sur toutes les pages à une même définition de style ;

- De changer l’aspect d’un site complet entier par la seule modification de quelques lignes ;
- Une plus grande lisibilité du HTML, car les styles sont définis à part ;
- Des chargements de page plus rapides, pour les mêmes raisons que précédemment ;
- Un positionnement plus rigoureux des éléments.

CHAPITRE 10: PROGRAMMATION EN JAVASCRIPT

LEÇON 1 : INITIATION AU LANGAGE JAVASCRIPT

Compétences visées : Au terme de cette leçon, l'apprenant sera capable de :

- Énoncer et expliquer les limites du HTML liées à l'interactivité ;
- Énoncer et décrire les avantages et les limites de Javascript ;
- Utiliser la balise <script>…</script> ;
- Déclarer les variables en Javascript ;
- Effectuer les conversions de type.

Situations problème : En classe de seconde, nous avons dit que la seule interaction dans une page web était le click sur les liens hypertextes. Cependant, comment faire pour récupérer des valeurs saisies par l'utilisateur ? Ou bien, comment faire pour afficher des messages d'alerte, soumettre les données d'un formulaire, réaliser les opérations et bien d'autres. Ces tâches ne sont pas possibles en programmation HTML. Quel est le langage de programmation qui peut supporter ces tâches ?

Introduction

JavaScript est un langage de programmation de scripts orienté-objet qui s'exécutent côté client. Cela signifie que le programme javascript est interprété et exécuté grâce au navigateur web de la machine du programmeur (machine cliente). Le javascript est un langage de scripts qui peut être inclus dans des pages HTML pour les rendre dynamique. Le logiciel pour interpréter et afficher le résultat d'un programme écrit en est appelé le navigateur web. Grâce à ce langage, il est possible d'écrire des pages web interactives.

I. HISTORIQUE, DEFINITIONS ET LIMITES DU HTML LIEES A L'INTERACTIVITE

1. Historique

Javascript a été mis au point par Netscape en 1995 pour enrichir ou apporter des améliorations au html et donner plus d'interactivité tels que la création des boutons, l'affichage des boîtes de dialogue, la récupération des informations du formulaire, la réalisation des opérations, la gestion des évènements tels que le clic ou le déplacement de la souris, etc.

2. Définitions

Javascript est un langage de scripts orienté-objet qui peut être incorporé dans un document html pour apporter plus d'interactivité aux pages web.

Un **script** est un ensemble d'instructions intégrer dans le code d'une page html et interpréter par le navigateur web du client.

Un navigateur web est un logiciel d'application permettant d'interpréter et d'afficher le résultat d'un programme javascript.

3. Limites du html liées à l'interactivité

L'interactivité correspond au changement observable suite à une action (le click ou le déplacement de la souris, la saisie de données dans une zone de texte, etc.) effectuée par l'utilisateur dans la page web. Or le seul

changement observable dans une page web apparait lorsque l'utilisateur click sur le lien hypertexte. Ce qui ne satisfait pas toujours l'utilisateur. Ainsi, les limites du html liées à l'interactivité peuvent être :

- Le html ne permet pas d'afficher les boîtes de dialogue ;
- Le html ne permet pas de soumettre les informations saisies dans le formulaire ;
- Le html ne permet pas le contrôle des données saisies dans le formulaire ;
- Le html ne permet pas d'effectuer les calculs ;
- Le html ne prend pas en compte la gestion des évènements tels que le click ou le déplacement de la souris ;

II. AVANTAGES ET LIMITES DU JAVASCRIPT

1. Avantages du javascript

L'avantage principal du javascript est qu'il permet de rendre les pages web interactives ou dynamiques. En effet, le javascript est important car il permet :

- La vérification des champs du formulaire ;
- La validation du formulaire ;
- L'affichage des messages d'alerte ;
- La gestion des évènements tels que le click ou le déplacement de la souris ;
- La récupération des valeurs entrées par l'utilisateur pour exécuter des commandes ;
- Réaliser des opérations ;
- Les fonctions javascripts s'exécutent directement grâce au navigateur présent sur votre machine. Etc.

Par ailleurs, le javascript contient aussi des limites.

2. Limites du javascript

Comme limites, nous avons :

- Javascript n'est pas compatible à tous les navigateurs web ;
- Le code javascript n'est sécurisé. Il est visible par tout le monde ;
- Javascript ne permet d'écrire ou de lire sur le disque dur ;
- Les erreurs en javascript sont difficiles à détecter à cause de l'absence d'un débuggeur ;
- Javascript ne permet pas de faire des bases de données ;
- Javascript ne permet pas de réaliser des forums de discussion.

3. Différences entre le langage javascript et le langage java

Le tableau ci-après présente quelques différences entre les langages javascript et java :

Javascript	**Java**
Langage interprété	Langage pseudo-compilé
S'intègre toujours au code html	Ne s'intègre pas toujours dans le code html
Accessibilité du code	Confidentialité du code
Langage peu évolué et peu typé	Langage très évolué et fortement typé

III. INSERTION DU CODE JAVASCRIPT DANS LA PAGE HTML

Tout script javascript est délimité par les balises <script> et fin </script>. La syntaxe générale du code javascript est définie par l'une des deux méthodes suivantes :

Méthode 1 :
```
<script language="javascript">
    Ici le code javascript ;
</script>
```

Méthode 2 :
```
<script type="text/javascript">
    Ici le code javascript ;
</script>
```

Le code javascript peut être inséré à trois endroits par exemple :

- Dans l'en-tête du document html entre les balises <head>…</head>. **Exemple** :

```
<html>
  <head>
    <title>Titre de la page</title>
    <script language="javascript">
      Placer le code javascript ici ;
    </script>
  </head>
  <body>
  </body>
</html>
```

- Dans le corps du document html entre les balises <body>…</body>. **Exemple** :

```
<html>
  <head>
     <title>Titre de la page</title>
  </head>
  <body>
    <script type="text/javascript">
      Placer le code javascript ici ;
    </script>
  </body>
</html>
```

- Dans un fichier externe portant l'extension .js. Dans ce cas, il faut écrire dans l'en-tête du code html le lien vers le fichier. Par exemple : **<script type="text/javascript" src="MonFichier.js"></script>**. Le principal avantage d'écrire le code javascript dans un fichier externe est que ce code peut être réutiliser pour d'autres pages web. C'est la méthode la plus recommandée.

NB : Pour insérer les commentaires sur une seule ligne dans le code javascript, il faut utiliser le double slash //. Lorsque le commentaire dépasse une ligne, il faut utiliser /* et */.

Exemple de code javascript :

/* Le code javascript suivant permet d'afficher le message « Bonne et heureuse année académique 2019-2020 à tous les élèves. » en utilisant la méthode **document.write()**. */

```
<html>
  <head>
    <title> Voici une page contenant du Javascript</title>
  </head>
  <body>
  <script language="Javascript">
     document.write ("Bonne et heureuse année académique 2019-2020 à tous les élève"); //Permet d'afficher le
message.
  </script>
  </body>
</html>
```

IV. Opérateurs

i. Operateurs arithmétiques

Comme en mathématique, en JavaScript nous avons les operateurs : addition (+), soustraction (-), multiplication(*), le reste de la division modulo (%).

ii. Operateurs de comparaison

Les operateurs de comparaisons en JavaScript sont :

traduisant l'égalité	==
traduisant l'inégalité	!=
supérieur ou égal	>=
inferieur ou égal	<=
inferieur	<
supérieur	>
pour traduire l'affectation	=

iii. Operateurs logique

Les opérateurs logiques en JavaScript sont :

traduisant le ET logique	&&
traduisant le OU logique	||
traduisant le NON logique	!
traduisant	===

V. DECLARATION DES VARIABLES EN JAVASCRIPT

Une **variable** est un objet mémoire dont le contenu peut être modifié au cours de l'exécution du script. En javascript, le nom d'une variable doit respecter les règles suivantes :

- Il doit commencer par une lettre majuscule, minuscule ou par le caractère underscore ;
- Il ne doit pas commencer par un chiffre ;
- Il ne doit pas contenir des espaces ;
- Il ne doit pas être un mot réservé comme : **Boolean, char, int, public, if, for, return, while, float, private, function,** etc.

En outre, javascript est sensible à la casse ! C'est-à-dire que javascript fait la différence entre les majuscules et les minuscules, les caractères accentuées et les caractères non-accentués. **Exemples** : « **Mavariable** » est différente de « **mavariable** » de même que « **élève** » est différent de « **eleve** ».

En javascript, il existe deux types de déclarations de variables : **la déclaration explicite** et la **déclaration implicite.**

IV.1 Déclaration explicite et déclaration implicite

Pour déclarer explicitement une variable, il faut utiliser le mot clé « **var** ». Il permet d'indiquer de façon rigoureuse que c'est une variable. Le tableau suivant présente quelques exemples de déclarations explicites.

Var x ; // déclaration explicite

Var x=18 ; //déclaration explicite avec affectation de valeur.

Pour déclarer une variable de manière implicite, le mot-clé « var » n'est pas utilisé. Le tableau suivant présente quelques exemples de déclarations implicites.

Pour déclarer implicitement une variable, il suffit d'écrire le nom de la variable suivie de caractère = (égal) et de la valeur à affecter. **Exemple** : **chaine="bonjour"**. Dans la déclaration implicite, le soin est laissé au navigateur de déterminer qu'il s'agit d'une déclaration de variable.

NB : Même si une déclaration implicite est tout à fait reconnue par le navigateur, il est plus rigoureux de déclarer les variables de façon explicite avec le mot-clé « **var** ».

En javascript, nous pouvons aussi déclarer les variables en utilisant le constructeur « new ». Ce dernier permet de déclarer les variables de même type que les types de base de javascript. Les types de base en javascript sont : Nombre (entier ou à virgule), Booléen et Chaine de caractères.

Exemples: var x = new Number (); var y = new String ("Bonjour");

IV.2 Portée (visibilité) des variables

Selon l'endroit où une variable est déclarée, celle-ci pourra être accessible (visible) de partout dans le script ou bien uniquement dans une portion confinée du code. On parle de « **portée** » d'une variable. La **portée d'une variable** représente la portion du code où la variable est visible.

Lorsqu'une variable est déclarée sans le mot-clé ***var***, c'est-à-dire **de façon implicite**, elle est accessible de partout dans le script (n'importe quelle fonction du script peut faire appel à cette variable). On parle alors de **variable globale.**

La portée d'une variable déclarée **de façon explicite** (précédée du mot-clé *var*), dépend de l'endroit où elle est déclarée :

- Une variable déclarée au début du script, avant toute fonction, sera **globale**. Elle peut être utilisée n'importe où dans le script ;
- Une variable déclarée explicitement dans une fonction aura une portée limitée à cette seule fonction, c'est-à-dire qu'elle est inutilisable ailleurs. On parle alors de **variable locale**.

Voici deux exemples permettant de l'illustrer :

```
<SCRIPT language="Javascript">
   <!--
            var a = 12;
            var b = 4;
    function MultipliePar2(b) {
              var a = b * 2;
              return a;
    }
   document.write("Le double de ",b," est ",MultipliePar2(b));
   document.write("La valeur de a est ",a);
   // -->
</SCRIPT>
```

Dans l'exemple ci-dessus, la variable *a* est déclarée explicitement en début de script, ainsi que dans la fonction. Voici ce qu'affiche ce script :

Le double de 4 est 8
La valeur de a est 12

Voici un autre exemple dans lequel *a* est déclarée implicitement dans la fonction :

```
<SCRIPT language="Javascript">
   <!—
          var a = 12;
          var b = 4;
      function MultipliePar2(b) {
          a = b * 2;
          return a;
      }
      document.write("Le double de ",b," est ",MultipliePar2(b));
      document.write("La valeur de a est ",a);
      // -->
</SCRIPT>
```

Voici ce qu'affiche ce script :

Le double de 4 est 8
La valeur de a est 8

VI. TYPES DE DONNÉES DES VARIABLES JAVASCRIPT

Un **type de données** correspond à un ensemble de valeurs qu'une variable peut prendre. En javascript, il n'est pas nécessaire de déclarer le type de variable contrairement à des langages évolués tels que le C ou le java pour lesquels il faut toujours indiquer le type de variables. Javascript utilise quatre types de données :

- **Number** : il s'agit des nombres entiers ou à virgule tels que 22 ou 3.14 ;
- **String** : il s'agit des chaines de caractères tels que "bonjour" ;
- **Boolean** : il s'agit des variables qui peuvent prendre deux valeurs : "**True**" si le résultat est vrai ou "**False**" si le résultat est faux. Elles permettent de vérifier une condition ;
- **Null** : il s'agit des variables qui ne contiennent aucune donnée.

VII. CONVERSION DE TYPES

Par défaut, les variables en javascript sont de types chaine de caractères (String). Il sera souvent nécessaire de convertir ces variables en types entier ou en type réel.
Pour convertir une variable en type entier, il faut utiliser la fonction **parseInt**(). Il faut utiliser la fonction **parseFloat**() pour convertir une chaine en type réel.

VI.1 FONCTION PARSEINT()

La fonction ***parseInt ()*** permet de convertir une variable passée en paramètre (soit en tant que chaîne de caractère, soit en tant que nombre dans la base précisée en second paramètre) en nombre entier (en base décimale). La syntaxe de la fonction *parseInt()* est la suivante : **parseInt(chaine[, base])** ;

Pour que la fonction *parseInt()* retourne un entier, la chaine passée en paramètre doit commencer par des caractères valides : c'est-à-dire les chiffres **[0-9]** ou le préfixe hexadécimal ***0x***, et/ou les caractères **+, -, *E* et *e***. Dans le cas contraire la fonction *parseInt()* retournera la valeur ***NaN*** (***Not a Number***).

Si les caractères suivants ne sont pas valides, ils seront ignorés par la fonction *parseInt()*. Si la chaîne passée en paramètre représente un nombre possédant une partie littérale, celle-ci sera tronquée.

Le paramètre "***base***" est un entier facultatif permettant de préciser la **base** devant être utilisée pour interpréter la chaîne. Il vaut **10** par défaut. Si le paramètre ***base*** n'est pas précisé (ou s'il est fixé à la valeur 10), la base utilisée sera la base décimale ; la base sera **16** si la chaîne commence par ***0x***, elle sera **8** si la chaîne commence par ***0***. Le tableau suivant donne des exemples d'utilisation de la fonction parseInt()

Exemples	**Résultats**
parseInt("128.34 ")	128
parseInt("Bonjour ")	NaN
parseInt("0284,8 ")	284
parseInt("12.3E-6")	12

parseInt("AF8BEF")	NaN

VI.2 La fonction parseFloat()

La fonction ***parseFloat()*** permet de convertir une variable passée en paramètre en nombre en virgule flottante (**nombre avec une partie décimale**). Sa syntaxe est la suivante : **parseFloat(chaine)** ;

Pour que la fonction ***parseFloat()*** retourne un flottant, la chaine passée en paramètre doit commencer par des caractères valides. C'est-à-dire les chiffres [**0-9**] et/ou les caractères **+, -, *E* et *e***. Dans le cas contraire la fonction *parseFloat()* retournera la valeur ***NaN* (*Not a Number*)**.

Si les caractères suivants ne sont pas valides, ils seront ignorés par la fonction *parseFloat()*. Si la chaîne passée en paramètre représente un nombre possédant une partie littérale, celle-ci sera tronquée. Le tableau suivant donne des exemples d'utilisation de la fonction ***parseFloat()*** :

Exemples	**Résultats**
parseFloat("128.34") ;	128.34
parseFloat("128,34") ;	128
parseFloat("12.3E-6") ;	0.0000123
parseFloat("Bonjour") ;	NaN
parseFloat("24.568Bonjour38") ;	24.568
parseFloat("Bonjour28.34") ;	NaN
parseFloat("AB8BEF") ;	NaN
parseFloat("0234") ;	234

Conclusion

Javascript est un langage de programmation de script qui s'exécute grâce au navigateur web. Il permet d'apporter des améliorations au html tels que l'affichage des boites de dialogue. Toutefois, le javascript est un langage de programmation basique dont la prise en main est nécessaire. $

O.P.O : A la fin de cette leçon l'apprenant devra être capable de :

- ✓ Définir et utiliser la fonction de lecture
- ✓ Définir et d'utiliser les fonctions d'écritures
- ✓ Définir et utiliser les structures qu'offrent JavaScript

I. Fonctions du langage

L'affichage et la lecture des données se font via les fonctions JavaScript **alert()** et **prompt()**.

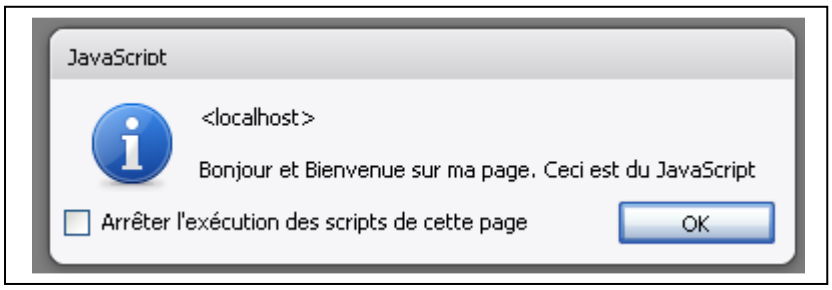
La fonction **alert ()** sert à afficher une valeur a l'écran et la fonction **prompt()** sert à lire une données au clavier.

Exemple :

➢ **Syntaxes d'Affichage des données**

- alert (Nom_Variable) : pour afficher le contenu d'une variable.
- alert ('Mon Texte') ou alert ("Mon Texte") : pour afficher un texte à l'écran.

Exemple d'affichage : alert ("Bonjour et Bienvenue sur ma page. Ceci est du JavaScript");



Remarque :

L'affichage peut aussi se faire grâce à la fonction **document.write**('Votre Texte') ou **document.write**(Nom_Variable). Cette fonction génère du code HTML puis l'affiche.

Exemple : document.write("Bienvenue, ceci est du HTML généré par JavaScript");

Résultat :

➢ **Lecture d'une donnée**

Syntaxe : X = prompt ('Texte à afficher') ou X = prompt ("Mon Texte") : lire une donnée et la ranger dans la mémoire après validation de la saisie.

Exemple : X = prompt ("Entrez votre nom");

Résultat :

II. Structure en JavaScript

1- Structure Séquentielle

Pour exécuter des instructions en séquence, il suffit de les écrire chaque instruction les unes après les autres. Au cas où il y a plusieurs instructions sur une même ligne, il faut les faires suivre obligatoirement d'un point virgule.

Syntaxe :
```
Instruction 1
Instruction 2
Instruction3; Instruction4; Instruction5
Instruction 4 ;
```

Exemple :
```
a = 15 ;
b =52 ;
c = 2*a – b ; d = b/a ;
```

2- Structures conditionnelles (alternatives)

i. Structure avec une alternative

Syntaxe : Formulation générale
```
if (<condition>) {
        <action>;
        …
}
```

Exemple :
```
if (i == 5) {
  Somme = somme *5 ;
}
```

ii. Structure avec deux alternatives

Syntaxe : Formulation générale
```
if (<condition>) {
        <action>;
        …
}
else {
        <action>;
        …
}
```

Exemple :
```
if (i == 5) {
        Somme = somme *5;
}
else {
        Somme = somme /5 ;
}
```

iii. Structure avec alternatives reliées

Syntaxe : Formulation générale
```
if (<cond1> <operateur> <cond2> ) {
        <action> ;
        <action> ;
         …
}else {
          <action> ;
         …

}
```

Exemple:
```
if (( i>=10 ) && (i <= 50) ) {
        somme = somme /5;
}
else {
        somme = somme +10;
}
```

iv. Structure avec conditions imbriquées

Syntaxe : Formulation générale
```
if (<condition1>) {
        <action>
} else {
 if (<condition2>) {
        <action>
        …
 } else {
          <action>
          …
        }
}
```

Exemple :

```
if ( reponse ==1 ) {
        cadeau = “cigarettes”;
} else {
  if ( reponse==3 ) {
        cadeau = “fleurs” ;
  } else {
        cadeau = “chaussettes” ;
  }
}
```

3- Structures répétitives (ou boucles ou itératives)

Il est très utile de disposer d’une instruction permettant d’effectuer de manière répétitive une série d’opérations. JavaScript propose plusieurs types de boucles itératives.

i. La boucle while

L’instruction **while** permet d’exécuter une instruction (ou un groupe d’instructions) un certain nombre de fois.

Syntaxe : Formulation générale

```
while ( <condition> ) {
        <action>;
        ...
}
```

Exemple :

```
a = 0 ; somme = 0 ;
while (a <25) {
        a = a+1 ;
        somme = somme + a ;    }
```

Signification : Exécuter les <actions> aussi longtemps que la <condition> est vraie. Si la condition est fausse au début, aucune instruction n’est exécutée.

ii. La boucle do .... while

L’instruction **do … while** permet d’exécuter une instruction (ou un groupe d’instructions) un certain nombre de fois.

Syntaxe : Formulation générale

```
do {
        <action> ;
        ...
}
while ( <condition> ) ;
```

Exemple :

```
a = 0 ; somme = 0 ;
do {
        a = a+1 ;
        somme = somme + a ;
}
while(a<25) ;
```

Signification Exécute les <actions> aussi longtemps que la <condition> est vraie. Si la condition est fausse au début, les instructions seront quand même exécutées une seule fois.

iii. La boucle for

Syntaxe : Formulation générale

```
for ( <expression de départ> ; <condition de continuation> ; <incrémentation> ) {
        <action>
        ...
}
```

Exemple :

```
somme = 0 ;
for (a=0; a < 25; a = a+1 ){
        somme = somme + a;
}
```

Signification : La boucle for se déroule de la manière suivante :

Tant que la condition de continuation est vraie c'est-à-dire (***a<25***):

- en partant de l'expression de départ (***a = 0***) on exécute le contenu des accolades;
- la variable (***a***) est incrémentée (***a = a+ 1***) et on exécute le contenu des accolades autant de fois que nécessaire;
- la boucle ne prend fin que lorsque la condition de continuation devient fausse c'est-à-dire lorsque ***a*** devient supérieur à **25**.

Introduction

On appelle fonction un sous-programme qui permet d'effectuer un ensemble d'instruction par simple appel de la fonction dans le corps du programme principal. Cette notion de sous-programme est généralement appelée fonction dans les langages autres que le JavaScript (toutefois leur syntaxe est généralement différente...).

O.P.O : A la fin de cette leçon l'apprenant devra être capable de :

- ✓ Définir et utiliser une fonction dans un exercice
- ✓ Définir et utiliser une procédure dans un exerce
- ✓ Donner la différence entre une fonction et une procédure

I. Présentation générale

Les fonctions et les procédures permettent d'exécuter dans plusieurs parties du programme une série d'instructions, cela permet une simplicité du code et donc une taille de programme minimale. Dans JavaScript, les fonctions et les procédures sont définies par le mot clé ***function***.

La différence entre une fonction et une procédure est que la fonction retourne une valeur (numérique, booléen ...) à la fin de son exécution et la procédure ne retourne aucune valeur. Ce retour de valeur se fait par le mot clé ***return*** qui est la dernière instruction de la fonction.

NB : Avant d'être utilisée, une fonction ou procédure doit être définie car pour l'appeler dans le corps du programme il faut que le navigateur la connaisse, c'est-à-dire son nom, ces arguments (paramètres) et les instructions qu'elle contient.

II. Syntaxe

La définition d'une fonction ou procédure s'appelle déclaration et se fait suivant la syntaxe suivante :

Syntaxe :

```
function nom_fonction(parametre 1, parametre 2, ...) {
    <instructions >
    <instructions >
        …..
    [ return    resultat; ]  /* présent si on une fonction */
}
```

Exemple : fonction calculant le carrée d'un nombre qu'on lui passe en paramètre.

```
function carre(nombre){
        res = nombre*nombre ;
        return   res;
}
```

Pour utiliser une procédure , il suffit de faire appel à elle en écrivant son nom (tout en respectant la casse) suivie d'une parenthèse ouvrante (éventuellement ces arguments) puis d'une parenthèse fermente.

Syntaxe : Nom_Procedure () ;

Pour utiliser une fonction, il suffit de faire appel à elle en écrivant son nom (tout en respectant la case) suivie d'une parenthèse ouvrante (éventuellement ces arguments) puis d'une parenthèse fermente. Et la faire précéder d'une variable qui récupérera la valeur retournée.

Syntaxe : Nom_Variable = Nom_Fonction() ;

Exemple : a = carre(5) ;

III. Exemple de programme JavaScript

Pour illustrer tout ce qui a été fait jusqu'à ce niveau nous allons écrire un programme JavaScript qui permet de résoudre une équation du second degré. Nous allons écrire deux versions : l'une utilisant les procédures et fonctions et l'autre pas.

i. Programme sans fonctions ou procédures

```
<html>
<head> <title> résolution d'une équation du second degré </title>
</head
<body>
        <script language="JavaScript" >
                alert('ce programme nous permettra de résoudre une  équation du second degré de la forme ax² + bx +c = 0 ');

                a=prompt ("entrez la valeur de la variable a ");
                b=prompt ("entrez la valeur de la variable b ");
                c=prompt ("entrez la valeur de la variable c ");

                alert("l'équation à résoudre est : ("+ a + ")x² + (" + b + ")x + (" + c + ") = 0 ");

if((a==0) && (b==0)){
                alert('c\'est pas une équation du second degré');
        }
      else {
                if((a==0) && (b!=0)){ alert('s = { ' + -c/b + ' }' ); }
                else {
                        if((a!=0) && (b==0)) {
                                if ((-c/a)<0) {
                                        alert('impossible');
                                }
    else {
                                x1=Math.sqrt(-c/a);
                                alert('S = { ' + x1 + ' }' );
                                        }
                                }
else {
                        delta = b*b - 4*a*c;
                                if(delta<0) {
                                        alert('ensemble vide');   }
                                else {
                                        if(delta==0) { alert('S = { ' + -b/(2*a) + ' }' ); }
                                                else {
                                                        x1 = (-b - Math.sqrt(delta))/(2*a);
                                                        x2 = (-b + Math.sqrt(delta))/(2*a);
                                                        alert('S = { ( ' + x1 + ' ; ' + x2 + ' ) }' );
                                                                }
                                                        }
                                                }
                                        }
  }
alert("  Fin d\'exécution. Merci pour avoir testé ce programme !!! Appuyer sur OK puis sur F5 pour recommencer ");
        </script>
</body>
</html
```

ii. Programme avec fonctions ou procédures

```
<html>
<head>
<title> Résolution d'une Equation du second degré </title>

    <script language="JavaScript" >
        function debut_programme(){
            alert('Ce programme nous permettra de résoudre une équation du second degré de la forme ax² + bx +c = 0 '); }
        function saisie_equation( ){
            a=prompt ("Entrez la valeur de la variable a ");
            b=prompt ("Entrez la valeur de la variable b ");
            c=prompt ("Entrez la valeur de la variable c "); }
        function affiche_equation(a, b, c){
            alert("L'équation à résoudre est : ("+ a + ")x² + (" + b + ")x  +  (" + c + ") = 0 "); }
        function  determinant(a, b, c){
            delta = b*b - 4*a*c;
            return delta;            }
        function  determinant_positif(a, b, delta){
            x1 = (-b - Math.sqrt(delta))/(2*a);
            x2 = (-b + Math.sqrt(delta))/(2*a);
            alert('S = { ( ' + x1 + ' ; ' + x2 + ' ) }' );   }
        function  fin_programme( ){
            alert("  Fin d\'exécution. Merci pour avoir utilisé ce programme !!! Appuyer sur OK puis sur F5 pour recommencer ");
    }
        function  affiche_msg1( ){
            alert('C\'est pas une équation du second degré');    }
        function  affiche_msg2( ){
                alert('S = { ' + -c/b + ' }' );     }
function  affiche_msg3( ){
        alert('Impossible');      }
function  affiche_msg4( ){
        alert('S = { ' + Math.sqrt(-c/a) + ' }' );      }
function  affiche_msg5( ){
        alert('ensemble vide');       }
function  affiche_msg6( ){
        alert('S = { ' + -b/(2*a) + ' }' );       }
</script>

</head
<body>
        <script language="JavaScript">
                debut_programme( );
                saisie_equation( );
                affiche_equation(a, b, c);
                if((a==0) && (b==0)){ affiche_msg1( ); }
                    else { if((a==0) && (b!=0)){ affiche_msg2( );}
                        else{ if((a!=0) && (b==0)){
                                if ((-c/a)<0){ affiche_msg3( ); }
                                else { affiche_msg4( );  }  }
                            else { delta = determinant(a, b, c);
                                if(delta<0) { affiche_msg5(); }
                                else { if(delta==0) { affiche_msg6(); }
                        else {  determinant_positif(a,b,delta); }
                } } } }
        fin_programme();
</script>
</body>
</html>
```

iii. Résultat d'exécution

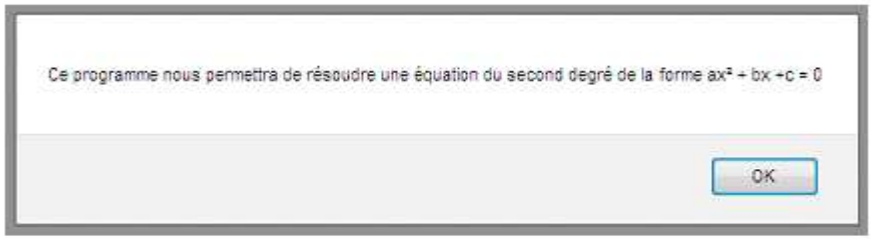


Figure 1 : Lancement de la page HTML

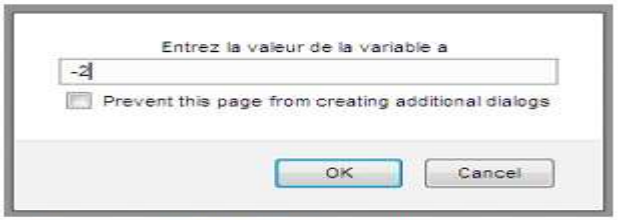


Figure 2 : Saisie de la variable « a »

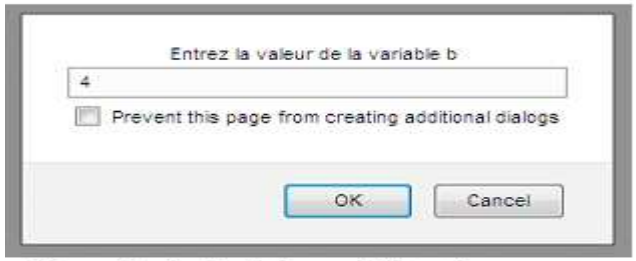


Figure 3 : Saisie de la variable « b »

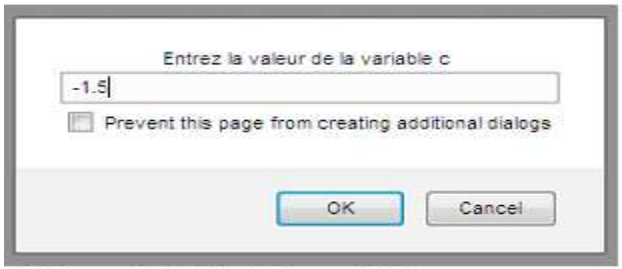


Figure 4 : Saisie de la variable « c »

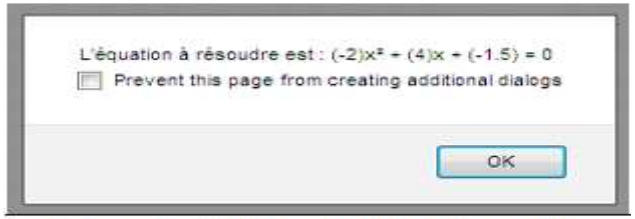


Figure 5 : Affichage de l'équation saisie

Figure 6 : Affichage du résultat d'exécution

Figure 7 : Affichage Message de fin d'exécution

UNITE D'APPRENTISSAGE 12 : PROGRAMMATION EN LANGAGE C

Compétences à développer :

- Programmation en mode console

45

UNITE D'ENSEIGNEMENT 45 : INTRODUCTION AU LANGAGE C

Objectifs pédagogiques :

- Installer un compilateur C ;
- Ecrire la structure d'un programme C ;
- Inclure les bibliothèques stdio.h, stlib.h, math.h et conio.h ;
- Utiliser les fonctions d'entrée/sortie classiques (scanf, printf, get,) ;

Contrôle de presrequis :

1. Donner les éléments essentiels d'un algorithme.
2. Savoir écrire un algorithme pour résoudre de problèmes mathématiques et physiques du niveau.

SITUATION PROBLEME :

Votre ami souhaite utiliser l'ordinateur pour exécuter ses algorithmes. Pour cela, votre petit frère lui propose la traduction de ces algorithmes en langage C avant de les exécutés. Ne connaissant rien sur ce langage, il fait donc appel à vous dans le but de l'expliquer quelques notions sur langage C.

Consignes :

1. Définir langage de programmation (**Réponse :** ensemble des mots et symboles permettant d'écrire un programme).
2. A part le langage C, quel autre langage de programmation connaissez-vous ? (**Réponse :** java, C++, pascal, python, javascript, PHP)
3. Comment appelle-t-on un algorithme déjà traduit en langage de programmation ? (**Réponse :** programme)
4. Comment appelle-t-on l'application qui permet d'exécuter un programme sur un ordinateur ? (**Réponse :** compilateur)
5. Donner la structure du programme écrit en langage C.
   Réponse :

```
[Directives au préprocesseur]
[Déclarations de variables externes]
[Fonctions secondaires]

int main ()
{
```

```
        Déclarations de variables internes
        instructions
    }
```

6. Donner le rôle des bibliothèques en langage C. Puis énumérer quelques exemples (**Réponse :**

7. Donner une fonction utilisée en langage C pour :
   - Afficher un message : **printf()**
   - Lire une variable : **scanf()**

RESUME

Définition :

Programmation : c'est la traduction d'un algorithme en un langage de programmation.
Langage de programmation : ensemble des mots et symboles permettant d'écrire un programme.
Programme : Suite d'instructions écrite dans un langage de programmation quelconque et permettant de réaliser une ou plusieurs tâches.

Installation d'un compilateur

Un **compilateur** : est une application qui transforme le code source d'un programme en un fichier binaire exécutable par la machine.
Exemple de compilateur : GNU Assembler, Turbo Pascal, Turbo C, GNU Pascal, Delphi, javac, GCJ (Gnu Compiler for Java), Jikes, Visual Basic, FreeBasic….

NB : chaque langage de programmation a son compilateur approprié. Par exemple Turbo Pascal, Delphi pour le langage **Pascal** et Javac , GCJ pour le langage **Java.**

Pour écrire des programmes, il est nécessaire d'installer différents outils. Le premier et le plus important est **l'installation d'un compilateur**.

Certains compilateurs sont déjà disponibles sur plusieurs SE, alors que d'autres sont spécifiques d'un système. Pour le plus connus sur Desktop : GCC (pour Windows, Android, Linux), Microsoft Visual Studio (MSVC).

D'autres compilateurs sont intégrés dans les environnements de développement (IDE). Il suffit de les cocher lors de l'installation de ces IDE.

Exemples des IDE: Code:: Blocks, Visual Studio, Qt Creator, Dev C++, Dev Pascal, Eclipse…

Structure générale d'un programme C

Définition :

Une expression : est une suite de composants élémentaires syntaxiquement correcte. Elle est toujours suivie d'un point-virgule (;)

En C on n'a pas une structure syntaxique englobant tout, comme la construction « **Algorithme ... Fin.** ». Un programme n'est qu'une collection de fonctions assortie d'un ensemble de variables globales. Ainsi un programme C est structuré comme suit :

```
[Directives au préprocesseur]
[Déclarations de variables externes]
[Fonctions secondaires]

int main ()
{
  Déclarations de variables internes
  instructions

}
```

La ligne : **int main()** se nomme un "**en-tête**". Elle précise que ce qui sera décrit à sa suite est en fait le "programme principal". Elle peut avoir des paramètres formels.

Le **programme (principal)** proprement dit est constitué des variables internes et des instructions et vient à la suite de cet en-tête. Il est délimité par les accolades "{" et "}".

Exemple de programme C affichant "Bonjour".

```c
#include <stdio.h>


int main(void)
{
    printf("Bonjour tout le monde !\n") ;
    return 0;
}
```

La première ligne de notre programme : #include <stdio.h> est en fait un peu particulière. Il s'agit d'une "directive" qui est prise en compte avant la traduction (compilation) du programme.

Ces directives, contrairement au reste du programme, doivent être écrites à raison d'une par ligne et elles doivent obligatoirement commencer en début de ligne. Leur emplacement au sein du programme n'est soumis à aucune contrainte (mais une directive ne s'applique qu'à la partie du programme qui lui succède). D'une manière générale, il est préférable de les placer au début.

La directive demande en fait d'introduire (avant compilation) des instructions (en langage C) situées dans le fichier stdio.h. Notez qu'un même fichier en-tête contient des déclarations relatives à plusieurs fonctions. En général, il est indispensable d'incorporer stdio.h.

Quelques bibliothèques (ou directives de préprocesseur) utilisés en langage C sont :

Bibliothèques	Rôle
<stdio.h>	Fournit la capacités centrales d'entrées/sorties

<math.h>	Pour calculer des fonctions mathématiques
<stdlib.h>	Pour exécuter les opérations de conversion, l'allocation des mémoires, le contrôle de processus, le tri, la recherche, …
<string.h>	Pour manipuler les chaines de caractères

NB : cette liste des bibliothèques standard C n'est pas exhaustive.

Notion des variables et operateurs

✓ Variable

Une variable est un objet manipulé dans l'exécution d'un programme. Elle est caractérisée par son **identificateur** (nom), son **type** et parfois sa **valeur** (le cas de constante).

Quelques types prédéfinis en langages C sont :

- ***int*** *(entier naturel)*
- ***Char*** *(caractères)*
- ***float , double*** *(nombre à virgule)*

Une variable doit être déclarée avant son utilisation. En C, toute instruction composé d'un spécificateur de type et d'une liste d'identificateurs séparés par ne virgule est une **déclaration**. Ainsi en C, une variable est déclarée de la manière suivante : *type identificateur ;*

Exemples : ***int*** *a ; et* ***char*** *a ;* sont de déclarations

✓ Operateur

Le tableau ci-dessous résume quelques operateurs utilisés en langage C

Operateurs	Symboles	Operateurs	Symboles
Affectation	=	égal	==
Reste de la division(modulo)	%	diffèrent	!=
Multiplication	*	ET logique	&&
Inférieur ou égal	<=	OU logique	||

Les entrées/sorties en C

Durant l'exécution d'un programme, le processeur, qui est le cerveau de l'ordinateur, a besoin de communiquer avec le reste du matériel. Il doit en effet recevoir des informations pour réaliser des actions et il doit aussi en transmettre. Ces échanges d'informations sont les entrées et les sorties (ou input / output en anglais).

- **Les sorties**

Le tableau ci-dessous décrit les trois fonctions d'affichage de données

Fonctions	Rôle
Printf()	Pour écrire une chaîne de caractères formatée
puts ()	Pour écrire une chaîne de caractères toute simple
Putchar()	Pour écrire un caractère

La syntaxe de l'utilisation de ces fonctions est :

```
fonction ("texte_ a_afficher… ") ;
```

Exemples :

```
printf("Bonjour le monde");
puts("Hello world !") ;
put("c");
```

La fonction **Printf** permet non seulement d'afficher des chaînes de caractères simples, mais également la valeur d'une variable passée en paramètre. Pour ce faire, il suffit d'utiliser un indicateur de conversion : il s'agit du caractère spécial % suivi d'une lettre qui varie en fonction du type de la variable.
Le tableau ci-dessous donne quelques indicateurs de conversion :

Type	Indicateurs de conversions
char	%c
int	%d
long	%ld
short	%hd
float	%f
double	%f
long double	%Lf
unsigned int	%u
unsigned short	%hu
unsigned long	%lu

Après avoir inscrit un indicateur de conversion dans la chaîne de caractère (dans les guillemets ""), il faut indiquer de quelle variable il faut afficher la valeur. Il suffit de rajouter une virgule après les ces derniers, suivis du nom de la variable, comme ceci :

```
printf ("% [lettre]", variable_ a_afficher);
```

Exemples :

```c
#include <stdio.h>


int main(void)
{
    int variable = 20;

    printf("%d\n", variable);
    return 0;
}
```

Remarque : Plutôt qu'appeler plusieurs fois la fonction printf pour écrire du texte, on peut ne l'appeler qu'une fois et écrire plusieurs lignes. Pour cela, on utilise le signe \ à chaque fin de ligne.

- **Les entrées**

Pour récupérer les valeurs saisies par l'utilisateur, nous utilisons les fonctions suivantes : **scanf ()** et **get ()**
La syntaxe de l'utilisation de la fonction Scanf() est :

```
scanf("%[lettre]", &variable qui_contiendra_notre_valeur);
```

NB : La fonction scanf a besoin de connaitre l'emplacement en mémoire de nos variables afin de les modifier. Afin d'effectuer cette opération, on utilise le symbole &. Donc, si vous oubliez le &, le programme plantera quand vous le lancerez, car vous tenterez d'accéder à une adresse mémoire inexistante !
Exemple :

```c
#include <stdio.h>


int main(void)
{
    int age;

    printf("Quel âge avez-vous ? ");
    scanf("%d", &age);
    printf("Vous avez %d an(s)\n", age);
    return 0;
}
```

SITUATION D'INTEGRATION

Vous souhaitez écrire un programme C qui demande à l'utilisateur de saisir un nombre puis affiche ce nombre.

1. Donner la structure d'un programme C
2. Quelles sont les bibliothèques standard que vous avez besoin dans votre programme C
3. Ecrire l'algorithme qui permet de réaliser cette tâche.
4. Traduire votre algorithme en langage C

REINVESTISSEMENT

Dans le cadre d'un TP sur la programmation, on vous demande d'écrire un programme C qui calcule la somme de deux nombres.

1. Démarrez votre éditeur de texte puis écrire le programme demandé.
2. Enregistrez votre programme sous le nom « **addition.c** »

46 UNITE D'ENSEIGNEMENT 46 : LES STRUCTURES DE CONTROLE

Objectifs pédagogiques :

- Ecrire la syntaxe des structures for, while et do…while en C;
- Utiliser l'opérateur conditionnelle (?) et virgule ;

Contrôle des prérequis :

1. Donner la structure d'un programme C
2. Donner les fonctions C utilisés pour afficher un texte à l'écran et pour lire une donnée fournie par l'utilisateur.

SITUATION PROBLEME :

Votre amie souhaite écrire un programme C qui calcule et affiche la table de multiplication d'un nombre positif saisi par l'utilisateur. Elle souhaite donc que ce programme affiche à chaque fois un message à l'utilisateur s'il saisi un nombre négatif. Rencontrant des problèmes sur l'écriture de ce programme, elle donc appel à vous dans le but de l'aider.

Consignes :

1. Qu'entend-on par structure de contrôle ? (**Réponse :** est une instruction particulière d'un langage de programmation impératif pouvant dévier le flot de contrôle du programme la contenant lorsqu'elle est exécutée).
2. Enumérer les différentes structures de contrôle utilisées en programmation. (**Réponse :** structure alternative, structure itérative)
3. Quelle structure utilise-t-on dans le cas où la condition se traite en deux cas possibles ? (**Réponse :** structure alternative).
4. Quelle boucle utilise-t-on dans chacun des cas suivants :

- La répétition connue d'un bloc instructions d'un programme ? (**Réponse :** le boucle pour...)
- La répétition non connue d'un bloc instructions d'un programme ? (**Réponse :** boucle tant que…).

RESUME

En programmation informatique, une **structure de contrôle** est une instruction particulière d'un langage de programmation impératif pouvant dévier le flot de contrôle du programme la contenant lorsqu'elle est exécutée. Le langage C propose plusieurs structures de contrôle différentes, qui nous permettent de nous adapter à tous les cas possibles. Il permet d'utiliser les conditions (structures alternatives), des boucles (structures itératives).

- **Structure alternative (if...else...)**

Il s'agit de tester une condition et d'effectuer une série d'instructions si la condition est vraie et si elle est fausse, effectuer une autre série d'instructions.

Il existe deux formes de structures conditionnelles :

- **La forme réduite**

```
if (condition) {

 bloc instructions 1

}
```

Exemple : le programme C qui prend en entrée un nombre puis affiche le message « positif » si c'est ce nombre est supérieur à 0 est :

```
#include <Stdio.h>
 int main (void) {
 int n ;
 printf ("saisir un nombre");
 scanf("%d",&n);
 if (n>0) {
 printf("Positif");
 }
 Return 0;
}
```

- **La forme complète**

```
if (condition) {

 bloc instructions 1

} else {

bloc instructions 2

}
```

Exemple : le programme C qui permet d'étudier la parité d'un nombre saisi par l'utilisateur est :

```
#include <Stdio.h>
 int main (void) {
 int n ;
 printf ("saisir un nombre");
 scanf("%d",&n);
 if (n%2==0) {
 printf("%d",n,"est paire");
 } else {
 printf("%d",n,"est impaire");
 }
 Return 0;
}
```

- **L'opérateur conditionnel ?**

L'opérateur conditionnel ? est un opérateur ternaire. Sa syntaxe est la suivante : ***condition ?*** *expression 1 : expression 2 ;*

Cette expression est égale à ***expression 1*** si ***condition*** est satisfaite, et à ***expression 2*** sinon.

Exemple : le programme C qui calcule la valeur absolue d'un nombre est :

```
#include <Stdio.h>
 int main (void) {
 int x, abs;
 abs=((x>=0)? x : -x) ;
 printf(" abs= %d",abs);
 }
 Return 0;
}
```

- **L'opérateur virgule**

Une expression peut être constitué d'une suite d'expressions séparées par des virgules :

expression 1, expression 2 , …… , expression n ;

Cette expression est alors évaluée de gauche à droite. Sa valeur sera la valeur de l'expression de droite. Par exemple, le programme

```
#include <Stdio.h>
 int main (void) {
 int a, b ;
 b=((a=3),(a+2)) ;
 printf(" b= %d",b);
 }
 Return 0;
}
```

retournera 5 à la sortie.

NB : la virgule séparant les arguments d'une fonction ou les déclarations des variables n'est pas l'opérateur virgule.

- **Structures itératives**

Toutes les structures itératives répètent l'exécution de traitement(s). Deux cas sont cependant à envisager, selon que :

- Le nombre de répétitions est connu à l'avance : c'est le cas des boucles itératives.
- Le nombre de répétitions n'est pas connu ou est variable : c'est le cas des boucles conditionnelles.

  - **Boucle for**

Cette structure est une boucle itérative ; elle consiste à répéter un certain traitement un nombre de fois fixé à l'avance. Cette structure est donnée par :

```
for (initialisation ; condition ;incrémentation ) {

bloc instructions;

}
```

Exemple : Affichage de nombres plus petit ou égal à 5.

```
for (i=0 ;i<=5 ; i++ ) {

printf(" i=%d",i);

}
```

- **Boucle while**

Parfois, pour réaliser une tâche, on doit effectuer plusieurs fois les mêmes instructions, sans que le nombre de fois soit déterminé à l'avance. On utilise alors une boucle conditionnelle. Dans cette structure, le même traitement est effectué tant qu'une condition reste valide ; la boucle s'arrête quand celle-ci n'est plus remplie. Cette structure répétitive est ainsi formulée :

```
While (condition vraie) {

 bloc instructions;

}
```

Exemple : affichage des nombres plus petit que 5

```
i=0 ;
While (i < 5) {
printf("i=%d",i);
i++;
}
```

- **Boucle do…while**

Une variante de structure répétitive avec boucle conditionnelle consiste à répéter un traitement jusqu'à ce qu'une certaine condition soit vérifiée. Dans ce cas, la boucle est exécutée au moins une fois, après quoi on teste la condition. On la traduit par l'instruction :

```
do {

 bloc instructions;

} While (condition);
```

Exemple : affichage des nombres plus petit que 5

```
i=0;
do {
printf("i=%d",i) ;
i++;
} While (i<5) ;
```

SITUATION D'INTEGRATION

Soit le programme suivant :

```
#include <stdio.h>
int main(void)
{       int i,n,som;
        som = 0;
        for(i=0;i<4;i++)
        {       printf(''donnez un entier '');
                scanf('' %d '',&n);
                som+=n;
        }
printf(''somme : %d\n'',som);
return 0 ;
}
```

Ecrire un programme C réalisant exactement la même chose, en utilisant :

a) Une instruction while;
b) Une instruction do ... while

REINVESTISSEMENT :

Soit le programme C ci-dessous :

```
#include <Stdio.h>
 int main (void) {
 int a,b,m;
 m =((a>b)? a : b) ;
 printf("%d",m);
 }
 Return 0;
}
```

a. Donner la valeur de la variable **m** dans chacun des cas suivants :

a	b	m
1	0	
23	24	
15	5	

b. Traduire ce programme en utilisant la structure alternative (if…else).

47 UNITE D'ENSEIGNEMENT 47 : LES TABLEAUX EN LANGAGE C

Objectif pédagogique :

- Déclarer un tableau en C ;

SITUATION PROBLEME :

Votre camarade Ali souhaite écrit un programme C qui calculera la moyenne arithmétique de 50 nombres. Votre petit frère lui conseil d'utiliser le tableau dans son programme pour stocker les 50 nombres. Ne connaissant pas comment utiliser le tableau en C, il fait appel à vous dans le but de l'aider.

Consignes :

1. Définir tableau (**Réponse :** un tableau est un ensemble fini d'éléments de même type stocké en mémoire)
2. Donner l'intérêt de l'utilisation d'un tableau. (**Réponse** : Le tableau permet la sauvegarde des données de manière structurée).
3. Donner les caractéristiques d'un tableau (**Réponse :** l'identificateur du tableau, le type des éléments du tableau et sa taille c'est-à-dire le nombre d'éléments)
4. Donner la syntaxe de déclaration d'un tableau en langage C (**Réponse** : type nom-tableau [taille-tableau]).
5. Expliquer comment, on remplit un tableau avec ses éléments en C (**Réponse** : affecter à chaque case du tableau une valeur)
6. Expliquer comment on accède aux différents éléments d'un tableau en C (**Réponse :** on accède à l'élément du tableau de la façon suivante : nom_tableau [indice_élément])

RESUME

Déclaration d'un tableau

Un tableau est un ensemble fini d'éléments de même type, stockés en mémoire à des adresses contiguës.
La déclaration d'un tableau à une dimension se fait de la façon suivante :

***type nom-du-tableau*[*nombre-éléments*] ;**

où ***nombre-éléments*** est une expression constante entière positive. Par exemple : ***int tab*[10] ;** indique que ***tab*** est un tableau de 10 éléments de type ***int.***
Pour plus de clarté, il est nécessaire de donner un nom à la constante ***nombre-éléments*** par une directive au préprocesseur, par exemple

 # define nombre-éléments 10

De façon similaire, on peut déclarer un tableau à plusieurs dimensions. Par exemple pour un tableau à deux dimensions :

***type nom-du-tableau*[*nombre-lignes*] [*nombre-colonnes*] ;**

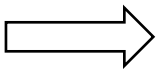
Exemple : ***int tab*[10][8]** est un tableau à deux dimensions de 10 lignes et de 8 colonnes.

Ajout des éléments d'un tableau

Pour remplir les différents éléments du tableau, on pourra affecter à chaque case de ce tableau une valeur.

Exemple :

```
int tab[2] ;
tab[0]=10 ;
tab[1]= 9 ;
```



10	9

Evidemment, à ce rythme-là, l'affectation est longue, surtout si votre tableau est de grande taille. C'est pourquoi on utilise la boucle itérative **for**.

On peut initialiser le tableau lors de sa création par une liste de constantes de la façon suivante :

***type nom-du-tableau*[N]={*constante-1,…,constante-N*} ;**

Par exemple, on peut écrire :

```
#define N 4
 int tab[N]={1,2,3,4} ;
```

1	2	3	4

Accès aux éléments d'un tableau

On accède à un élément du tableau en lui appliquant l'operateur []. Les éléments d'un tableau sont toujours numérotés de **0** à ***nombre-éléments-1***

Par exemple l'instruction **`printf("%d",tab[2]) ;`** affiche le troisième élément du tableau (contenu du tableau à l'indice i=2).

On pourra afficher tous les éléments du tableau en utilisant la boucle **for.** Le programme ci-dessous réalise la tâche.

```
#include <Stdio.h>
#define N 10
 int main () {
 int tab[N];
 int i;
 ……
 for (i=0;i<N;i++) {
 printf("tab[%d]=%d\n",I,tab[i]);
 }
 return 0;
}
```

SITUATION D'INTEGRATION

Ecrire un programme C qui calcule la moyenne de 5 nombres saisi par l'utilisateur.

REINVESTISSEMENT

Ecrire un programme C permettant de créer le tableau ci-dessous et de rechercher dans ce tableau une valeur fournie par l'utilisateur. Si la valeur est trouvée, alors le message suivant est affiché : « Succès ! ». Dans le cas contraire affiché « Echec ! ».

Tableau :

1	12	10	5	3

48 UNITE D'ENSEIGNEMENT 48 : LE SOUS-PROGRAMME

Objectif pédagogique :

- Déclarer et appeler un sous-programme en C ;

Contrôle des prérequis :

1. Définir sous-programme et donner les exemples.
2. Donner l'intérêt de l'utilisation de sous-programme.
3. Donner la syntaxe de déclaration et d'appel d'une fonction en algorithmique.

SITUATION PROBLEME :

Votre ami Isaac a utilisé le langage C pour créer une calculatrice qui effectue les opérations d'addition, de soustraction, de division et de multiplication. Il constate que le code source de son programme est touffu et volumineux ce qui lui rend difficile la compréhension de son code. Il cherche donc un moyen très efficace pour mieux structurer son programme. Pour cela il fait appel à vous dans le but de l'aider.

Consignes :

1. Que doit faire votre ami Isaac pour organiser et mieux comprendre son code ? (**Réponse :** diviser son programme en des sous-programmes)
2. Comment appelle-t-on le sous-programme en C ? (**Réponse :** fonction)
3. Donner la syntaxe de définition d'une fonction en C (**Réponse :**
   type nom-fonction(type-1 argument-1,…,type-n argument-n) {
   [déclarations des variables locales]
   listes d'instructions })
4. Comment fait-on appel à une fonction en langage C ? (**Réponse :** nom-fonction (liste-paramètre))

RESUME

Définition d'une fonction

L'écriture et la compréhension d'un programme peut s'avérer difficile lorsque le nombre d'instructions devient trop important. A ce moment, le programme devient lourd et son exécution par l'ordinateur devient beaucoup plus lente. Pour pallier ce problème, on a mis sur pied la notion de sous-programme informatique dont l'objectif principal est de scinder un programme complexe et touffu en plusieurs sous-programmes plus simples et plus efficaces. En C, les sous programmes sont appelés les **fonctions.**

A part la fonction principale main() qui est obligatoire, les autres fonctions d'un programme C sont dites fonctions secondaires.

Une fonction en C se définit de la manière suivante :

```
type nom-fonction(type-1 argument-1,…,type-n argument-n) {
  [déclarations des variables locales]
  listes d'instructions
}
```

La première ligne de cette définition est **l'en-tête** de la fonction. Dans cet en-tête, ***type*** désigne le type de la fonction, c'est-à dire le type de la valeur qu'elle retourne.

NB :

- Contrairement à d'autres langage de programmation, en C, il n'y a pas la notion de procédures et de sous-programme. Une fonction qui ne renvoie pas de valeur est une fonction dont le type est spécifié par le mot clé **void.** Les arguments de la fonction sont appelés **paramètres formels**, par opposition aux **paramètres effectifs.**

- Si la fonction ne possède pas de paramètres, on remplace la liste de paramètres formels par le mot clé **void.**

- Le corps de la fonction débute par des déclarations de variables locales à cette fonction. Il se termine par l'instruction de retour à la fonction appelante, ***return,*** dont la syntaxe est :

  return(expression) ;

  La valeur ***expression*** est la valeur que retourne la fonction. Son type doit être de le même que celui qui a été spécifié dans l'entête de la fonction. Si la fonction ne retourne pas de valeur (fonction de type void), sa définition s'achève par ***return ;***

Exemples :

- Une fonction calculant le produit de deux nombres est :

```
int produit (int a, int b) {
int p ;
p=a*b;
return p ;
}
```

- Une fonction qui imprime les N éléments d'un tableau est :

```
void imprime-tab (int *tab, int N) {
int i ;
for(i=0; i<N; i++){
printf("%d \t ", tab[i]);
}
return ;
}
```

Appel d'une fonction

L'appel d'une fonction se fait par l'expression

nom-fonction(paramètre-1,…,paramètre -n)

NB : L'ordre et le type des paramètres effectifs doivent concorder avec les ceux donnés dans l'en-tête de la fonction.

Déclaration d'une fonction

La déclaration d'une fonction en langage C peut être faite soit avant, soit après la fonction principale **main().** Toutefois, il est indispensable que le compilateur connaisse la fonction au moment ou elle celle-ci est appelée. Si une fonction est définie après son premier appel (en particulier si sa définition est placée après la fonction main), elle doit être impérativement déclarée au préalable. Une fonction secondaire est déclarée par son prototype, qui donne le type de la fonction et celui de ses paramètres, sous la forme :

type nom-fonction(type-1,…,type-n) ;

Exemple :

```
int produit (int , int) ;

int produit (int a , int b){
   return a*b;
}
int main() {
int c=2,d=5 ;

printf("%d\n",produit(c,d));
}
```

Déclaration de la fonction

Définition de la fonction

Appel de la fonction

Notion de variable locale et globale

Définitions :

Une **variable globale** : est une variable déclarée en dehors de toute fonction. Elle est connue d compilateur dans toute la portion de code qui suit sa déclaration.
Une **variable locale** : est une variable déclarée en l'intérieur d'une fonction (ou d'un bloc d'instructions) du programme.
NB : Les variables locales n'ont en particulier aucun lien avec des variables globales de même nom.

SITUATION D'INTEGRATION

On veut écrire un programme C qui calcule la moyenne de 50 nombres saisis par l'utilisateur.

a. Ecrire une fonction nommée **Lecture** qui permet de lire les 50 nombres puis les stockent dans un tableau de taille 50.
b. Ecrire une autre fonction nommée **somme** qui permet de sommer les 50 nombres stockés dans le tableau.
c. Ecrire une autre fonction nommée **moyenne** qui calcule la moyenne de ces 50 nombres.
d. Ecrire la fonction principale faisant appel à ces fonctions pour résoudre le problème posé.

REINVESTISSEMENT

Démarrez votre éditeur de texte puis saisissez le code source d'un programme C à l'intérieur duquel on y trouve une fonction qui permet de calculer x^n où x et n sont les nombres entiers fournis par l'utilisateur.

49 UNITE D'ENSEIGNEMENT 49 : EXECUTION D'UN PROGRAMME C

Objectif pédagogique :

- Tester un programme dans un compilateur

Contrôle des prérequis :

1. Savoir écrire un programme en langage C

SITUATION PROBLEME :

Votre amie claudine a écrit un programme qui calcule la somme des 10 nombres en utilisant le langage C. Elle souhaite donc tester ce programme pour savoir s'il réalise vraiment la tâche demandée. Rencontrant des difficultés, elle fait appel à vous dans le but de l'aider.

Consignes :

1. De quoi a-t-elle besoin pour tester son programme ? (**Réponse :** compilateur)
2. Donner le rôle d'un compilateur. (**Réponse :** traduire un programme en langage machine)
3. Décrire les étapes qu'elle doit suivre pour résoudre son problème. (**Réponse :** saisir le code source de son programme dans un éditeur de texte puis l'enregistrer sous l'extension **.c**, utiliser ensuite un compilateur pour produire un fichier exécutable. Ou bien utilise un IDE pour la saisie et l'exécution de son programme).

RESUME

Tester un programme C en utilisant le compilateur *gcc*

Le C est un langage compilé (par opposition aux langages interprétés). Cela signifie qu'un programme C est décrit par un fichier texte, appelé fichier source. Ce fichier n'étant évidemment pas exécutable par le microprocesseur, il faut le traduire en langage machine. Cette opération est appelée la **compilation** et est effectuée par un programme appelé compilateur. Elle se décompose en fait en 4 phases successives :

1. **Le traitement par le préprocesseur** : le fichier source est analysé par le préprocesseur qui effectue des transformations purement textuelles (remplacement de chaînes de caractères, inclusion d'autres fichiers source …).
2. La **compilation** : la compilation proprement dite traduit le fichier généré par le préprocesseur en assembleur.
3. **L'assemblage** : cette opération transforme le code assembleur en un fichier binaire, c'est-`a-dire en instructions directement compréhensibles par le processeur.
4. **L'édition de liens** : un programme est souvent séparé en plusieurs fichiers source, pour des raisons de clarté mais aussi parce qu'il fait généralement appel à des librairies des fonctions standard déjà écrites. Une fois chaque code source assemble, il faut donc lier entre eux les différents fichiers objets. L'édition de liens produit alors un fichier dit exécutable.

Pour compiler votre code source C en utilisant le compilateur ***gcc***, on utilise la commande suivante :

```
gcc [options] fichier.c [-llibrairies]
```

Par défaut, le fichier exécutable s'appelle **`a.out`**. Le nom de l'exécutable peut être modifié à l'aide de l'option **`-o`** ***nom-de-fichier***

fichier.c est le nom de votre programme C que vous avez tapé dans votre éditeur texte. Un programme C est toujours enregistré sous l'extension ***.c***

Les éventuelles librairies sont déclarées par la chaîne ***-llibrairie***. Dans ce cas, le système recherche le fichier ***liblibrairie.a*** dans le répertoire contenant les librairies pré-compilées (généralement /usr/lib/). Par exemple, pour lier le programme avec la librairie mathématique, on spécifie **-lm**. Le fichier objet correspondant est **libm.a**. Lorsque les librairies pré-compilées ne se trouvent pas dans le répertoire usuel, on spécifie leur chemin d'accès par l'option **-L**

Exemple : la commande ***gcc -o programme1 programme1.c*** compile le code source du programme nommé ***programme1.c*** pour produire un fichier exécutable nommé ***programme1***

Apres compilation de votre programme C, il faut donc ouvrir le fichier exécutable pour tester votre programme.

Tester un programme C en utilisant un IDE : cas de Code::Blocks

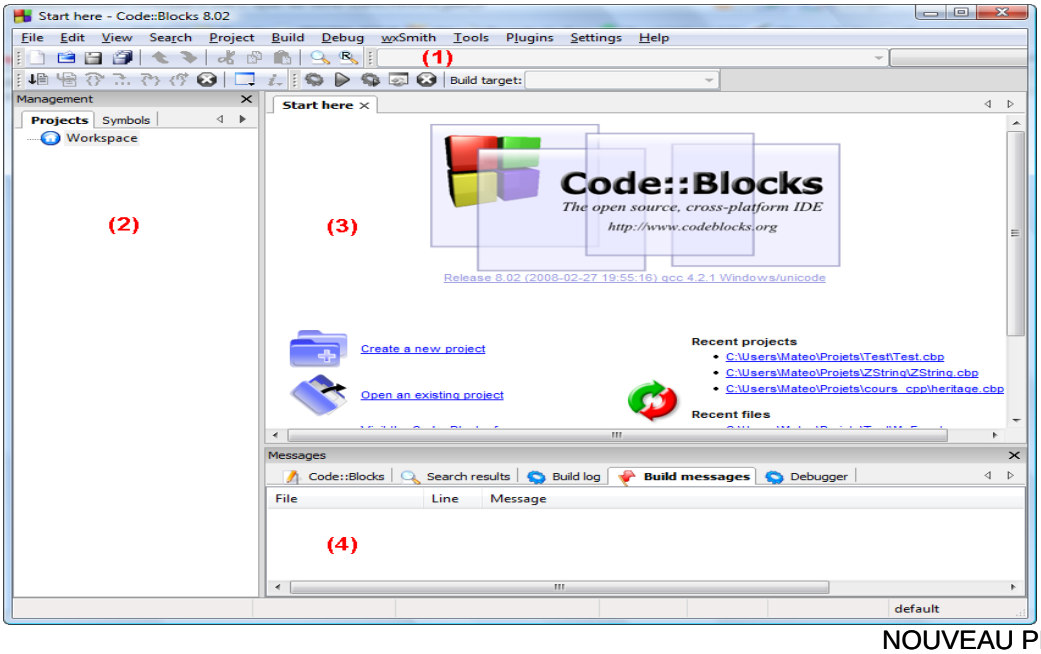
L'exécution d'un programme écrit en langage C se fait traditionnellement en utilisant un compilateur (gcc). Les fichiers sources sont écrits en utilisant un éditeur de texte et les programmes seront compilés depuis un terminal ou une invite de commande.

Pour faciliter la compilation des programmes C, l'on a mis sur pied des outils adaptés appelés **environnements de développement integré** (IDE : Integrated development environment).

Un environnement de développement intégré permet de prendre en charge les différentes opérations de compilation : il regroupe donc l'édition de texte (créer le code source) et les outils de compilation (activation du compilateur gcc par exemple) directement depuis le même logiciel. Il évite ainsi le lancement d'un terminal ou d'une invite de commande pour lancer les opérations de compilation.

En outre, l'environnement de développement intégré facilite la correction du programme (debug) en lançant l'outil de debug (gdb) auquel il ajoute la convivialité de l'environnement graphique (visualisation de l'évolution des variables en temps réel, par exemple).

Apres démarrage de code::Blocks, l'interface suivante apparait :



Ses différentes parties sont :

1. **La barre d'outils** : elle comprend de nombreux boutons.
2. La **liste des fichiers du projet** : c'est à gauche que s'affiche tous les fichiers sources de votre programme
3. La **zone principale** : c'est là que vous pourriez écrire votre code en langage C
4. La **zone de notification** : aussi appelée « la zone morte », c'est ici que vous pourriez voir les erreurs de compilation.

Sur la barre d'outils, on y trouve les boutons suivants :

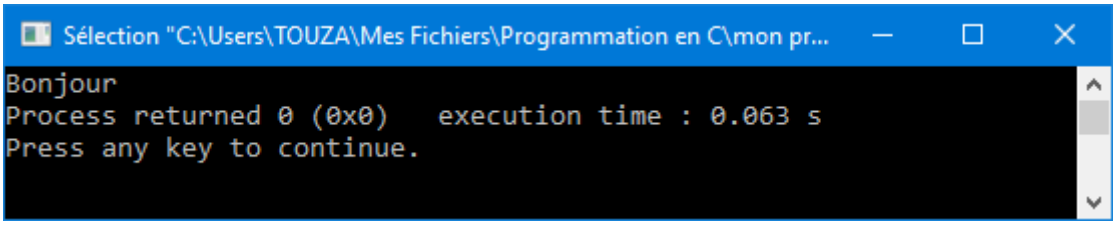
Boutons	Signification
 Compiler	Tous les fichiers source de votre programme sont envoyés au compilateur qui va se charger de créer un fichier exécutable
 Exécuter	Cette icone lance juste le dernier exécutable que vous avez compilé, cela vous permettra de tester votre programme
 Compiler & exécuter	Permet de lancer la compilation ensuite le fichier executable produit par le compilateur (raccourcis en utilisant la touche F9 du clavier)
 Tout recompiler	Permet de recompiler tous les fichiers de votre code source

Exemple :

Un programme C qui affiche bonjour est :

```c
#include <stdio.h>
#include <stdlib.h>
int main( )
{
printf( "Bonjour") ;
return 0;
}
```

Apres compilation et exécution en cliquant sur le bouton  , on obtient :



SITUATION D'INTEGRATION

1. Démarrer l'IDE installer sur votre machine puis écrire le code d'un programme C qui calcule la somme de trois nombres saisis par l'utilisateur.
2. Cliquer sur le bouton de compilation pour compiler votre programme.
3. Exécuter ce programme avec les valeurs 2, 3,23, 4 et 10.

REINVESTISSEMENT

Utiliser l'IDE code::Blocks pour écrire et exécuter un programme C qui calcule le PGDC de deux nombres entiers naturels.