## *Avant-Propos*

*Ce polycopié a été conçu et écrit dans un souci de clarté et de simplicité, les notions de base de l'informatique y sont abordées.*

*Le chapitre I introduit les systèmes informatiques à travers des définitions précises ; le codage des informations est présenté et étudié, une attention particulière a été apportée à l'explication de la représentation des nombres réels en machine. Avant leurs codages, les informations non numériques telles que le son et l'image sont d'abord numérisées, le lecteur curieux est renvoyé à l'annexe.*

*Le chapitre II, introduction à l'algorithmique et à la programmation, commence par définir les notions d'algorithme et de programme, donne une vue d'ensemble de l'algorithme ainsi que les étapes de construction d'un programme ; les types de données y sont aussi abordés.*

*Le chapitre III introduit les instructions de base en les définissant, donnant leurs syntaxes et en expliquant leurs exécutions à l'aide d'exemples. Définition, syntaxe et exemples en algorithmique et en langage C++ est la démarche suivie pour les chapitres suivants aussi.*

*Le chapitre IV poursuit la présentation des instructions, en définissant les structures de contrôle qui ne sont autres que des instructions où la machine a un certain contrôle sur l'enchainement des instructions à exécuter. Les différentes formes de l'alternative et de la répétition y sont abordées.*

*Le chapitre V traite des sous programmes, les notions de procédure et de fonction y sont présentées.*

*Les variables indicées ainsi que les structures sont étudiées dans le chapitre VI sur les types composés, on y trouvera les notions de tableaux à une et deux dimensions et d'enregistrement.*

*Enfin, le dernier chapitre présente quelques notions avancées telles que la récursivité et les pointeurs.*

*Nous espérons que ce modeste travail puisse aider nos étudiants dans la compréhension de la belle matière qu'est l'informatique et nous les incitons à faire de la programmation des ordinateurs un jeu, une passion et un entrainement pour leurs cerveaux.*

*L'auteur: Mahi.LS*

# CHAPITRE I : INFORMATIQUE et CODAGE DE L'INFORMATION

## 1/ Définitions

### l'informatique

*Informatique = contraction des mots **Informa**tion et auto**matique***

*L'informatique est une science qui regroupe l'ensemble des théories et des techniques permettant de **traiter** l'**information** à l'aide d'une machine.*

*Dans le langage courant, l'informatique peut aussi désigner à tort tout ce qui se rapporte au matériel informatique (l'électronique), et la bureautique.*

*À ce sujet on attribue une phrase à Edsger Dijkstra qui résume assez bien cela :*

*« L'informatique n'est pas plus la science des ordinateurs que l'astronomie n'est celle des télescopes ».*
*(en Anglais : "Computer science is no more about computers than astronomy is about telescopes.")*

### L'information

*Une information est un élément de connaissance pouvant se trouver sous différentes formes (texte, son, image, vidéo, Nombres) et susceptible d'être représentée à l'aide **d'un système de codage**, afin d'être transmise, conservée ou traitée*

### Le traitement

*C'est une suite d'opérations réalisées sur une représentation d'une information donnée pour aboutir à une information résultante résolvant un problème donné.*

# 2/ Les deux aspects d'un système informatique

*Un système informatique est caractérisé par deux aspects : le matériel (hardware) et le logiciel (software)*

## L'aspect Hardware (Matériel)

*Les éléments physiques, les composants électroniques, les câbles électriques- → l'ordinateur.*

***Ordinateur : Définition***

*Un ordinateur est une machine électronique capable de recevoir des informations, de les enregistrer, de les traiter selon un programme préalablement écrit et de restituer l'information résultante; c'est une machine dotée de mémoires à grande capacité et de moyens de calculs ultrarapides.*

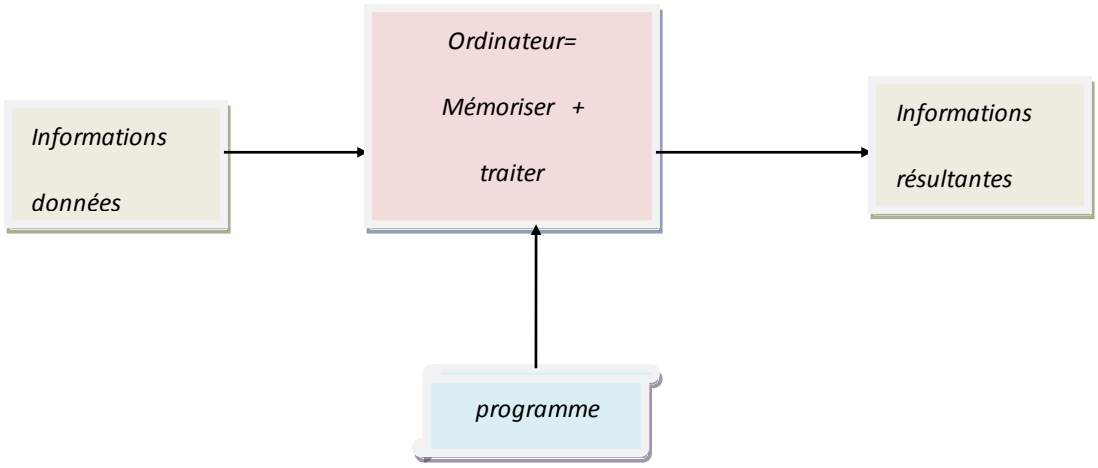



***Figure 1.1 : Schéma définition ordinateur***

## *Structure d'un ordinateur*

*La plupart des ordinateurs actuels sont construits selon une même architecture décrite depuis 1945 par John Von Neumann.*

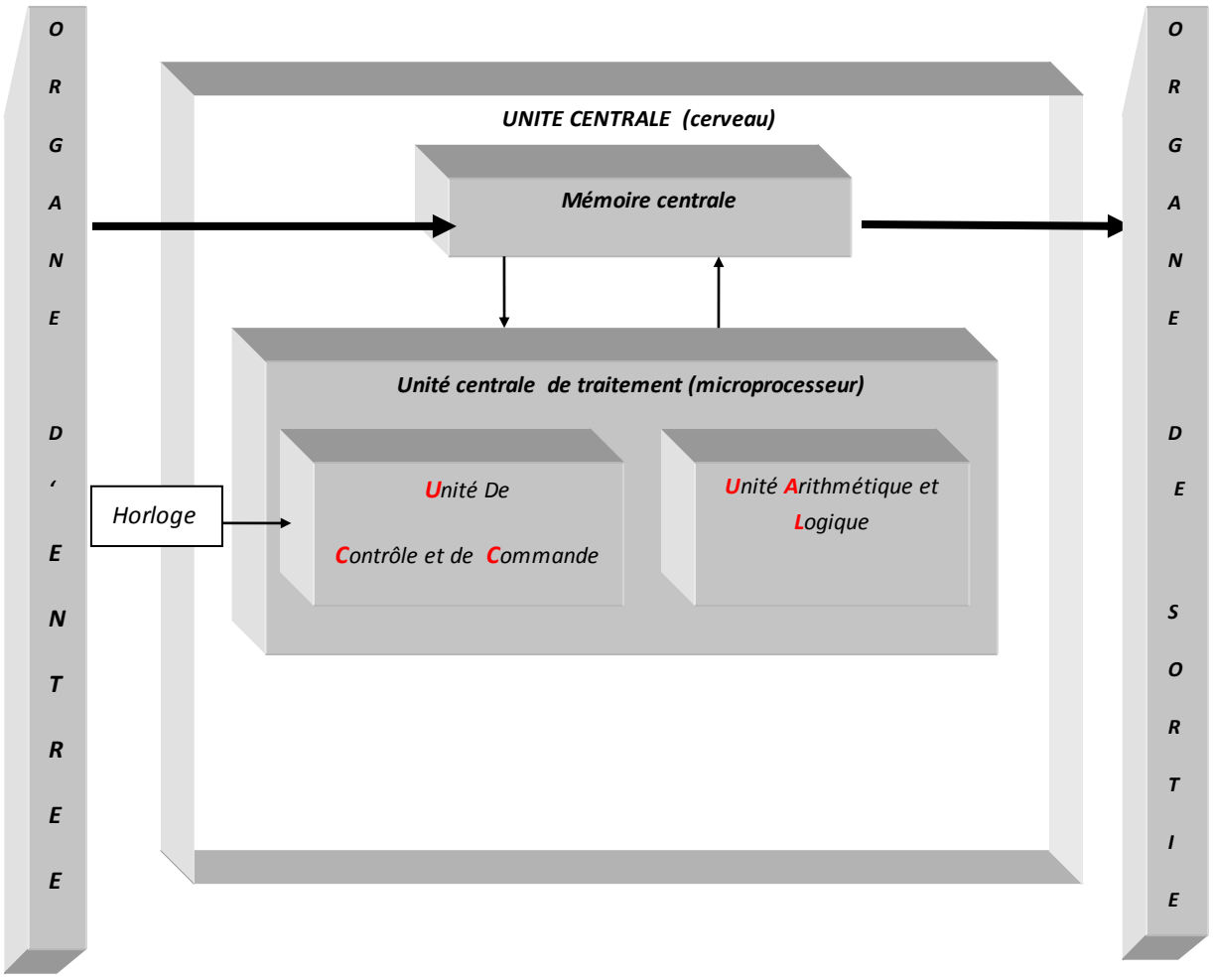



***Fig 1.2 : Structure de base d'un ordinateur : Architecture de Von Neumann***

- *Les flêches reliant les composants entre eux sont des ensembles de composants permettant de transporter plusieurs bits en parallèle, on les appelle pour cela des bus.*
- *La mémoire centrale ou RAM (Random Access Memory) est la partie de l'ordinateur qui stocke les données avant leurs traitements, les programmes pendant leurs exécutions, les résultats intermédiaires ainsi que les résultats finaux avant leurs affichages.*

- *L'Unité Centrale de Traitement est chargée d'effectuer les traitements des opérations de type arithmétique ou logique. L'UAL est son principal constituant. .*
- *l'Unité de Contrôle et de Commande est chargée de commander et de gérer les différents constituants de l'ordinateur (contrôler les échanges, gérer l'enchaînement des différentes instructions,…etc.) comme un policier chargé de la circulation.*

## *L'aspect logiciel*

*Un logiciel ou une application est un ensemble de programmes qui coopèrent afin de réaliser un objectif bien précis. Pour être actif, un programme doit résider en mémoire centrale (RAM). Un programme en exécution est appelé processus.*

- *Il existe trois catégories de logiciels :*

  - *Les langages de programmation : ce sont des logiciels qui permettent de produire d'autres logiciels. Un langage de programmation permet de rédiger un programme. Ce genre de logiciel est destiné aux personnes initiées à l'informatique et à la programmation.*

  - *Les logiciels d'application : ce sont des logiciels destinés généralement à des utilisateurs non informaticiens (Logiciel de comptabilité, logiciel de gestion de stock, tableurs, traitement de texte, jeux…).*

  - *Les systèmes d'exploitation : Le système d'exploitation est le logiciel de base de tous les systèmes informatiques, on peut dire que c'est le logiciel de gestion de la machine (ordinateur). C'est un ensemble de programme qui permet d'optimiser la gestion des ressources de l'ordinateur (mémoires, périphériques tels que imprimantes, disques,..etc.)*

- *Les principales fonctions d'un système d'exploitation sont :*

  - *Gérer les travaux confiés à la machine comme par exemple introduire les programmes en mémoire et contrôler leurs enchainements et leurs exécutions.*
  - *Gérer les opérations d'E/S des informations en organisant leurs stockages et leur protection*
  - *Utilisation optimale des ressources.*

# *3/ Codage des informations*

*Le **codage** est l'opération qui fait passer une information d'une représentation compréhensible par l'Homme à une représentation « compréhensible » par le système informatique.*

***Le système de codage** utilisé en Informatique est **le système binaire** basé sur deux valeurs logiques 0 et 1 correspondant à deux états stables des composants électroniques de base des circuits de la machine.*

*Les informations non numériques comme le son ou l'image par exemple sont d'abord représentées en numérique (on appelle cela **numérisation, voir Annexe A**) ensuite les nombres sont convertis en binaire (**codage**).*

*Donc, toute information, doit d'abord être représentée par une suite de 0 et de 1 (**codée**) pour pouvoir être traitée par la machine.*

## Système d'unités de quantification de l'information

*Le bit est la plus petite unité d'information. Une suite de huit bits constitue un **octet** ou **byte**. Pour quantifier l'information en machine, on utilise un système d'unités basé sur l'octet :*

*Un kilo-octet =1024 octets=$2^{10}$ octets (noté Ko)*

*Un méga-octet=1024 kilo-octest=1024*1024 octets=$2^{20}$ octets noté(Mo)*

*Un Gyga-octet=1024 méga-octets=$2^{30}$ octets (noté Go)*

*Un téra-octet=1024 Gyga-octets=$2^{40}$ octets (noté To)*

## Les systèmes de numération et le système binaire

*Un système de numération décrit la façon avec laquelle les nombres sont représentés.*

*Un système de numération est défini par un **ensemble de symboles** ou de chiffres et des **règles d'écriture** des nombres.*

***Exemple 1 : Numération romaine :***

***Ensemble de symboles**= {I,V,X,L,C,D,M]*

| I | V | X | L | C | D | M |
|---|---|---|---|---|---|---|
| 1 | 5 | 10 | 50 | 100 | 500 | 1000 |

***Règles d'écriture :***

- *Lorsqu'un symbole est placé à la droite d'un symbole plus fort ou égal sa valeur s'ajoute.*

  *Exemple : CCLXI→261*

- *Lorsqu'un symbole est placé à gauche d'un symbole plus fort, sa valeur est retranché.*

  *Exemple : CCXLII → 242*

- *On ne place jamais 4 symboles identiques à la suite : 9 s'écrit IX et non VIIII.*

*Ce système n'est pas adapté au calcul*

***Exemple 2 : Numération décimale :***

*Ce système de numération est le plus pratiqué,*

***Ensemble de symboles=[0,1,2,3,4,5,6,7,8,9]***

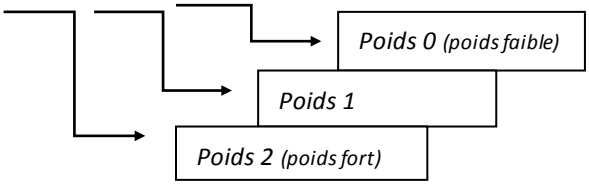
*Ces symboles représentent les valeurs que peut prendre chaque chiffre du nombre ; le choix de **10** valeurs a été naturellement dicté par le nombre de doigts des deux mains de l'être humain, on dit que **10** est la **base** de ce système l'origine de ce système est très ancienne 3000 ans avant Jésus Christ.*

***Règles d'écriture :*** *Nous parlerons ici du système décimal qu'on appelle système décimal de position :( le plus récent). On l'appelle **de position** ou **positionnel car chaque position de chiffre possède un poids.***

*$N= a_{n-1}a_{n-2}.....a_2a_1a_0$ où les $a_i$ sont les chiffres du nombre et chaque chiffre peut prendre l'une des valeurs de l'ensemble des symboles.*

*Exemple : N=439 a0=9 a1=3 a2=4*

*$439=4*10^2 + 3*10^1 + 9*10^0$*




***Système de numération positionnel pondéré à base b***

*Un système de numération à base b est défini sur un alphabet de B chiffres (ou symboles) :*

*$A=\{c_0,c_1,c_2,....,c_{B-1}\}$*

*Soit $N = a_{n-1}a_{n-2}.... a_2a_1a_0$ I La représentation d'un nombre en base b;*

*L'équivalent décimal de ce nombre est donné par la formule :*

*$N(10)= a_{n-1} * b^{n-1} + a_{n-2}*b^{n-2} + .... + a_2 * b^2 + a_1 * b^1 + a_0 * b^0$*

***Exemple1 : soit le système de numération à base 4 ;***

*les seuls chiffres utilisés dans ce système sont les chiffres de l'alphabet A={0,1,2,3]*

***soit le nombre N=213 représenté en base 4***

*L'équivalent décimal de ce nombre est :*

*N en base 10*

$N(10)= 2*4^2 +1*4^1 + 3*4^0 = 2*16 + 1*4 + 3*1 = 39$

***Exemple 2 : Soit le système de numération à base 2 (système binaire) ;***

*les seuls chiffres utilisés dans ce système sont les chiffres de l'alphabet A={0,1]*

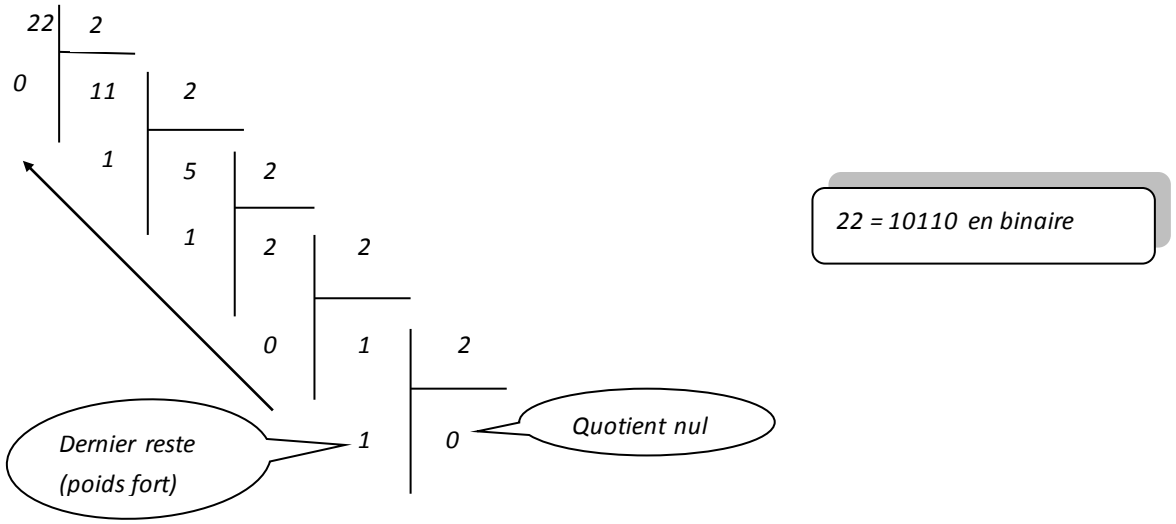
*Soit le nombre N=10110 représenté en binaire*
*L'équivalent décimal de N est :*
$N (10)=1*24 +0*23+1*22+1*21+0*20 = 1*16+0*8+1*4+1*2+0*1= 22$

***Passage de la base 10 à une base b***

*Pour trouver l'équivalent en base b d'un nombre N représenté dans la base 10, on procède à des divisions successives de N par b jusqu'à obtenir un quotient nul ; la suite des restes obtenus constitue la représentation de N dans la base b, le dernier reste étant le chiffre de poids fort.*

*Exemple :*

*Pour trouver l'équivalent binaire du nombre décimal 22, on procède à des divisions successives de 22 par 2.*



## *Passage d'une base b1 à une base b2 :*

- *Si b1 n'est pas une puissance de b2 ni b2 n'es t une puissance de b1 : une seule méthode.*

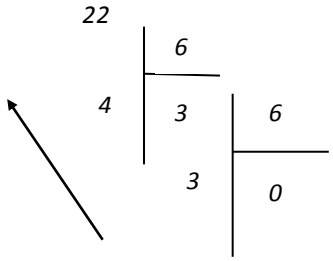
*Pour passer d'une base b1 à une base b2 , on passe par la base 10, c'est-à-dire on convertit le nombre N représenté en b1 au décimal (en appliquant la formule du système positionnel pondéré à base b), ensuite on convertit le résultat décimal trouvé en b2 en faisant des divisions successives par b2.*

*Exemple : convertir le nombre 10110 binaire en base 6 (b1=2 ; b2=6)*

- *On passe à la base 10 en appliquant la formule du système positionnel pondéré à base 2*

$10110 = 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0 = 1*16+0*8+1*4+1*2+0*1=$ ***22***

- *On convertit 22 en base 6 en faisant des divisions successives par 6*




*22 = 34 en base 6 donc*

*10110 binaire = 34 en base 6*

- *Si b1 est une puissance de b2 :on peut utiliser une deuxième méthode (plus rapide, sans passer par 10)*

***Exemple** : convertir N= 1010010 de la base 2 vers l'octal (base 8)*

*B1=2 et b2=8 ; b2 est une puissance de b1 (8 est une puissance de 2 car $8=2^3$)*

*On sépare les chiffres de notre nombre à convertir en groupes de 3 ($8=2^3$) comme suit :*

*N=1010010 → 001 010 010*

*Ensuite, on remplace chaque groupe par son équivalent octal :*

*001 010 010 → 122 (001=1 en octal et 010=2 en octal)*

# 4/ Représentation des nombres en machine

## Représentation des nombres entiers naturels (positifs)

*Les nombres entiers positifs sont représentés directement en binaire pur*

## Représentation des nombres entiers relatifs (signés)

- ***Signe et valeur absolue*** → *traitement spécial pour le signe* → *nécéssité de réalisation de nouveaux circuits pour l'addition et la soustraction ; cette représentation n'est pas utilisée en machine , elle est étudiée comme introduction aux autres représentations .*

***Exemple :***

| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Représente le nombre entier relatif -5

Bit de signe : 1 signifie négatif

- ***Complément à 2***

*Soit N un nombre écrit dans sa base B comprenant n chiffres, le complément à B de N est :*

$C_B(N)=B^n-N$

*Exemple1 : En décimal* $C(40)=10^2-40=60$ *càd le nombre qui complète N pour obtenir* $10^2$*, en fait c'est le complément à* $B^n$ *appelé complément à B.*

***Exemple en décimal*** *: si on représente -40 par 60 ; une soustraction revient à faire l'addition du complément*

*Avec des nombres à deux chiffres 70-40=30 → 70+60=130 et on ne prend que les n chiffres de poids faible (n=2 dans notre exemple).*

*la même chose se passe en binaire.*

***Exemple en binaire :***

$C_2(00001001)=2^8-00001001=100000000-00001001=11110111$

*En machine, seuls les nombres entiers négatifs sont représentés par leurs compléments à 2.*

# Représentation des nombres réels

## ➢ Virgule fixe et virgule Flottante

*Tout nombre réel est composé d'une partie entière et d'une partie fractionnaire.*

*Nous avons deux possibilités pour écrire les nombres réels :*

*1/ **PE,PF** (PE :Partie entière PF : Partie Fractionnaire)*

*exemple : 23,58 (PE=23 PF=58)*

*2/ **Notation scientifique** : $M*B^e$ (M : mantisse B : Base e : exposant )*

*exemple : le nombre réel 23,58 peut s'écrire de différentes manières en notation scientifique dans le système décimal ;*

*$2{,}358\ 10^1$ (M=2,358 B=10 e=1) Ou bien*

*$235{,}8\ 10^{-1}$ (M=235,8 B=10 e=-1 ) ou bien*

*$0{,}2358\ 10^2$ (M=0,2358 B=10 e=2)) dans ce cas $0 <M <1$ ; on dit que la mantisse est normalisée car elle obéit à une norme (une loi d'écriture des nombres réels en notation scientifique).*

*En notation scientifique, nous remarquons que nous pouvons déplacer la position de la virgule en changeant la valeur de l'exposant ; cette notation lorsqu'elle est utilisée en machine est appelée **virgule flottante** ( qui bouge) par opposition à la première représentation (PE,PF) qui est appelée **virgule Fixe**.*

*Si nous disposons de seulement 6 cases pour stocker les chiffres du nombre ;*

1. *pour la virgule fixe , nous devons choisir un nombre de cases pour PE et un nombre de cases pour PF*

***Exemple** :*

- *2 cases pour PE et 4 cases pour PF.*

*si nous ne considérons que les nombres positifs , Les nombres représentables seront de 00,0000 à 99,9999.*

- *Avec 3 cases pour PE et 3 pour PF nous pourrions représenter de 000,000 à 999,999.*
- *Avec 4 cases pour PE et 2 cases pour PF de 0000,00 à 9999,99*

***Nous remarquons que plus le nombre s'agrandit plus la précision s'affaiblit .***

**2.** *En notation scientifique, avec le même nombre de cases :6 cases,*

*Nous pouvons, par exemple, prendre 4 pour la mantisse et 2 pour l'exposant et nous pourrions dans ce cas représenter les nombres de 0,000 $10^{00}$ à 0,999 $10^{99}$ (bien plus de nombres qu'avec la virgule fixe)*

*Si nous voulons représenter des exposants négatifs et positifs sur 2 cases sans représenter le signe, alors au lieu de représenter seulement les exposants positifs de 00 à 99 nous allons partager cet intervalle en deux et représenter les exposants négatifs par les valeurs de 00 à 48 et les exposants positifs par les valeurs de 5O à 99 et le zéro par 49 comme suit :*

| *Exposant réel* | *valeur de représentation -* |
|---|---|
| *49* | *00* |
| *-48* | *01* |
| *-47* | *02* |
| ***-46*** | ***03*** |
| *.....* | *…* |
| *-2* | *47* |
| *-1* | *48* |
| *0* | *49* |
| *1* | *50* |
| *2* | *51* |
| *…….* | |
| *50* | *99* |

- *Nous remarquons que la valeur de représentation est décalée de 49 par rapport à l'exposant réel.*

  *Exemple : exposant réel =-46 → valeur de représentation =-46+49=03*

- *La valeur de décalage 49 est appelée biais et est calculée dans le cas général en divisant $B^n$ par 2 et en retranchant 1 ; pour notre exemple la base B=10 et n=2 (n est le nombre de cases pour la représentation de l'exposant) biais=$(10^2/2)-1=50-1=49$*

- *Nous remarquons aussi que les exposants sont ainsi ordonnés du plus petit nombre négatif au plus grand nombre positif ce qui facilite leur comparaison.*

## *Conclusion :*

***En virgule fixe*** *Il n'est pas commode de représenter par exemple à la fois*

*la masse de la terre = 5 973 600 000 000 000 000 000 000*

*et la masse de l'électron =0,00….00091093822*

*27 zéros*

*à moins de prévoir beaucoup de cases pour PE et beaucoup pour PF*

***En notation scientifique :***

***La masse de la terre s'écrit*** $5{,}9736 \times 10^{24}$ ***Et la masse de l'électron*** $9{,}1093822 \times 10^{-31}$

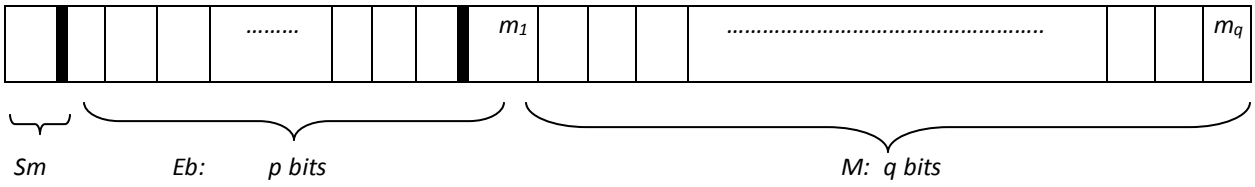
***10 cases (8 pour la mantisse et 2 pour l exposant) nous suffisent pour représenter les deux dans le même format***

- ***Représentation virgule Flottante standard IEEE 754***

***IEEE = Institute of Electrical and Electronics Engineers***
***(association professionnelle internationale, de droit américain) : fixe entre autres les lois de représentation des nombres réels en machine en simple (4octets) et double (8 octets) précision***

- ***En C, la simple précision correspond au type float et la double précision au type double.***
- ***Le format du nombre dans ce standard est composé de :***

- *Le signe du nombre (Sm : Signe mantisse) est codé sur 1 bit : le signe – : bit =1 Le signe + : bit =0*
- *Exposant biaisé (Eb) placé avant la mantisse pour simplifier la comparaison Codé sur p bits et biaisé (Eb=exposant réel+biais avec biais=$(2^p/2)-1$) donc (exposant réel=Eb-biais)*
- *Mantisse normalisé(M) •Normalisé : virgule est placé après le bit à 1 ayant le poids fort •M est codé sur q bits •Exemple : 11,01 → 1,101 donc M =101*




- *La formule donnant la valeur du nombre représenté en virgule flottante dans ce standard est:*

$N = \pm 1,M * 2^{Eb-biais}$ *ou bien en formule mathématique complète :*

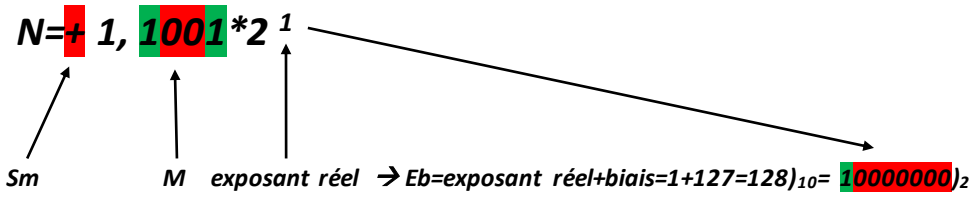
$N = (-1)^{Sm} * (1+\sum_{i=1}^{q} m_i * 2^{-i}) * 2^{Eb-biais}$

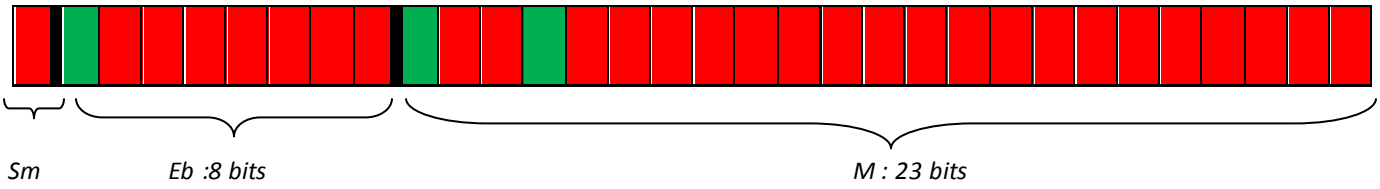
- ***Exemple de représentation en simple précision***

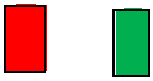
*En Simple précision, les nombres réels sont codés sur 4 octets = 32 bits répartis ainsi :*

*1 bit pour Sm ; 8 bits pour Eb (p=8 ; **biais**=($2^8$/2)-1=$2^7$-1=**127**) ; 23 bits pour la mantisse normalisée (q=23)*

*Soit N=+3,125 $)_{10}$ → convertir en binaire N=+11,001 → normaliser la mantisse **N=+1,1001*$2^1$***




- *Représentation en double precision:*

*En double precision, les nombres reels sont codés sur 8 octets =64 bits répartis comme suit:*

*1 bit pour SM ; 11 bits pour Eb (biais=($2^{11}$/2)-1=1023) ; 52 bits pour la mantisse normalisée.*

# Chapitre II : Notions d'algorithme et de programme

## 1/ Concept d'algorithme et de programme

*L'ordinateur n'a pas la capacité de trouver la solution à un problème, nous l'utilisons comme un agent exécutant très efficace pour effectuer rapidement des opérations que nous pourrions effectuer à la main, il revient à l'être humain de décrire les actions à exécuter, à partir des données pour obtenir les résultats voulues.*

*Un algorithme est la description pour un exécutant donné, d'une méthode de résolution de problème, par une suite d'actions fournissant le résultat cherché.*

*Dans la vie courante nous utilisons souvent des algorithmes, une recette de cuisine, un mode d'emploi,… sont des algorithmes qui sont destinés à un être humain. (l'être humain est dans ce cas l'exécutant)*

*L'écriture d'un algorithme doit tenir compte des caractéristiques, compétences et limites de l'exécutant.*

*L'algorithme ne doit pas contenir une action que l'exécutant ne comprend pas ou ne peut pas réaliser.*

*Un algorithme ne peut pas être exécuté directement par la machine pour ce faire, il faut le traduire dans un langage de programmation donné (pascal, C++, Java,…etc)*

*Un programme n'est autre que la traduction d'un algorithme dans un langage de programmation ; Un programme est un ensemble d'instructions écrites dans un langage « compris » par la machine.*

***Exemple*** *: algorithme pour la résolution d'une équation du second degré :*

```
Algorithme équation                                   ← En-tête
Variables  a,b,c,delta,x1,x2  :réel ;                 ← Bloc d'environnement
Début                                                 ← Bloc d'instructions
    Ecrire ('donnez les coéfficients  a b et c') ;
    Lire(a,b,c) ;                          //saisie des coefficients
    Delta ← b*b-4*a*c ;                    // calcul du déterminant delta
    Si delta<0  alors écrire ('pas de solution') sinon
        Si delta=0  alors écrire ('solution double x1=x2=',-b/(2*a))
            sinon écrire ('x1=',(-b-sqrt(delta))/2*a,  'X2=', (-b+sqrt(delta))/2*a))
FIN.
```

***Traduction de l'algorithme equation en langage C++***

```
Algorithme équation                                   #include<stdio.h>
Variables  a,b,c,delta,x1,x2  :réel ;                 #include<stdlib.h>
Début                                                 #include<math.h>
    Ecrire ('donnez les coéfficients  a b et c') ;    int main()
    Lire(a,b,c) ;                                     {
        //saisie des coefficients                     float a,b,c,delta,x1,x2;
    Delta ← b*b-4*a*c ;                               printf("donnez les coefficients  a b et c\n");
        // calcul du déterminant delta                scanf("%f%f%f",&a,&b,&c);
    Si delta<0  alors écrire ('pas de solution')      delta=b*b-4*a*c;
sinon Si delta=0 alors écrire ('solution double       if(delta<0)printf("pas  de solutions\n");
x1=x2=',-b/(2*a))                                     else if(delta=0)
            sinon écrire ('x1=',(-b-                  printf("solution double x1=x2=%f",-b/2*a);
        sqrt(delta))/2*a,  'X2=', (-                      else  printf("x1=%f  x2=%f",(-b-
        b+sqrt(delta))/2*a))                          sqrt(delta))/2*a,(-b+sqrt(delta))/2 *a);
FIN.                                                      system("pause");
                                                          return  0;
                                                          }
```

## 2/ Structure d'un algorithme ou d'un programme

*Comme vu dans l'exemple précédent, un algorithme est structuré comme suit*

***En-tête*** *: précise le nom de l'algorithme et indique le commencement de celui-ci.*

***Bloc d'environnement*** *: Déclare tous les objets qui seront utilisés dans le bloc d'instructions.*

***Bloc d'instructions*** *:Contient les actions à exécuter par la machine (les instructions).*

### *Les objets de l'environnement :*

*Les objets de l'environnement peuvent être :*

- *Des types*
- *Des constantes*
- *Des variables*
- *Des sous programmes (procédures ou fonctions)*

### *Les types :*

*Un type définit l'ensemble des valeurs que peut prendre une variable ainsi que l'ensemble des opérations et fonctions qu'on peut appliquer à ces valeurs.*

*Il existe des types primitifs qui sont prédéfinis, que la machine connaît déjà qui sont*

*Les entiers, les réels, les booléens, les caractères et les chaines de caractères*

***Les types primitifs*** *sont utilisés par le programmeur directement sans qu'il les définisse avant , la machine reconnaît leurs noms(entier, réel,..) . comme dans l'exemple de l'algorithme équation : variables a,b,c,delta,x1,x2 :* ***Réel.***

*Les types primitifs sont des types simple c'est-à-dire qu'une variable d'un type primitif n'est composée que d'une seule valeur.*

- *Avant de voir en détail chacun des type primitifs, nous allons parler des types non primitifs :*

*Dans certains cas, le programmeur peut avoir besoin de créer de nouveaux types :* ***types non primitifs***

- ***Les types non primitifs : le type énuméré et le type intervalle***

*Exemple : nous voulons représenter par une variable un jour de semaine, sachant qu'un jour de semaine ne peut prendre que les valeurs : samedi, dimanche, …vendredi, il nous faut créer un nouveau type par énumération de ses valeurs en le déclarant comme suit :*

***Type*** *jour_de_semaine=(samedi,dimanche,lundi,mardi,mercredi,jeudi,vendredi) ;*

*Suite à cette déclaration, nous pouvons utiliser ce type pour déclarer une variable comme suit :*

***Variable*** *j :jour_de_semaine ; //la variable j ne pourra prendre que l'une des valeurs définies dans la déclaration du type jour_de_semaine.*

*Ce genre de type non primitif est appelé* ***type énuméré****.*

*Il existe un autre genre de type non primitif qu'on appelle* ***le type intervalle****, il s'agit de définir un intervalle à partir d'un type primitif ordinal (un type ordinal est un type dont les valeurs peuvent être comptées :type entier ou caractère).*

*Exemple : on veut représenter par une variable l'âge d'une personne, sachant que l'âge est compris entre 0 et 160 maximum ; il serait intéressant de le représenter par une partie du type entier (un intervalle) en déclarant un nouveau type*

*Type t_age= 0..160;*

*Variable age : t_age;*

*Ou bien, on peut utiliser le type directement dans la déclaration de la variable:*

*Variable age :0..160 ;*

*L'intérêt des type non primitifs est que ce sont des types sur mesure, de cette façon la machine peut détecter des erreurs sans que le programmeur ne se donne la peine d'ajouter des tests dans son programme.*

*Il existe d'autre types non primitifs qui sont des types structurés où la variable est composée de plusieurs valeurs: les tableaux, les piles, les enregistrements, les fichiers,.. sont des types structurés qu'on étudiera par la suite.*

## *Le type entier :*

***Les valeurs*** *de type entier sont un sous ensemble des entiers relatifs, les langages de programmation proposent généralement plusieurs types entiers ( l'entier court, l'entier standard,l'entier positif,l'entier long,. etc.) dont le nombre de bits en représentation interne change, d'où la différence dans la valeur max et la valeur min permises pour chaque type.*

*Exemple :en langage C, il existe les types entiers short, int, long, et unsigned short, unsigned int,..etc., en ajoutant le mot clé unsigned, il y aura un bit supplémentaire pour représenter la valeur du nombre mais le nombre sera forcément positif ( pas de signe).*

***Les opérations et fonctions*** *sur les entiers sont : (soient x et y deux entiers)*

***L'addition*** *(x+y), la* ***soustraction*** *(x-y),la* ***multiplication****(x*y) la* ***division****(x /y) dont le résultat est réel, la* ***division entière*** *(x div y) dont le résultat est entier, le* ***modulo*** *ou le reste de la division entière (x mod y) et les opérateur unaires (-x et +X) ; les fonctions utilisables sont la valeur absolue (abs (x)), la racine carré (sqrt (x)), le carré (sqr (x)), le successeur (succ (x)), le prédécésseur (pred(x)), ainsi que les fonctions trigonométriques (sin(x), cos(x), …) .*

## *Le type reel*

*Comme pour les entiers,* ***Les valeurs*** *du type réel en informatique sont un sous ensemble de l'ensemble des réels, on dit toujours sous ensemble car la machine est limitée, les nombres ne vont pas jusqu'à plus l'infini et moins l'infini. Les langages de programmation proposent différentes représentations des réels ainsi en C par exemple, les types* ***float****,* ***double*** *et* ***long double*** *représentent tous des nombres réels dont la représentation interne est en virgule flottante. C'est le nombre de bits pour la représentation de la mantisse et de l'exposant qui est plus ou moins grand pour chaque type donc si on veut utiliser des valeurs très précises (avec beaucoup de chiffres après la virgule), on utilisera le type long double par exemple sinon on utilisera le float ou double.*

***Les opérations et fonctions*** *utilisées dans la majotité des langages pour les réels sont : l'addition, la soustraction, la multiplication la division (/), les fonctions trigonométriques, le carré (sqr), la racine carré (sqrt), la troncature (trunc(x) qui a pour résultat la partie entière de la variable réelle x), l'arrondi (round(x) qui a pour valeur la valeur entière la plus proche de X, exemple : round(12.3)=12 et round(12.7=13).*

## *Le type caractère*

***Les valeurs*** *du type caractère sont en général tous les caractères du clavier (alphabétiques majuscules et minuscules, numérique de '0' à '9', spéciaux ( '?','.',' !','@', …etc.) et même les caractères de contrôles qui ne sont pas affichables (la touche retour, maj, ctrl,…etc.), tous ces caractères sont ordonnés suivant leur code (code ASCII) ; on peut donc comparer les caractères ('a'<'b' est vrai car le code de 'a' est inférieur au code de 'b').*

*Les valeurs de type caractère s'écrivent toujours entre apostrophes , le caractère apostrophe lui-même est doublé pour qu'il n y ai pas d'ambiguité ("").*

*Il ne faut pas confondre les valeurs numériques de type caractère ('0', '1', ..'9') avec les valeurs entières (0,1,..9) de type entier).*

***Les Opérateurs et fonctions sur les caractères*** *sont les* ***opérateurs*** *de relation (<,>,=,<=,>=,<>),* *les* ***fonctions :*** *succ (successeur : qui donne le caractère qui vient après), exemple : succ('c')='d',* *pred (prédécésseur : le caractère qui est avant), ord (ordre : qui donne le code ascii Du caractère donné entre parenthèses), exemple : ord('A') donne le code ASCII du caractère 'A'. la fonction char(chr) qui donne le caractère correspondant au code qui est donné entre parenthèse ( char est l'inverse de la fonction ord).*

## Le type chaine de caractères

*Le type chaine de caractères définit des variables dont la valeur est une suite de caractères comme un nom, un prénom ou une adresse par exemple. La valeur '27 rue de la paix' est de type chaine de caractère, elle est composée de 17 caractères, les espaces sont considérés comme des caractères. Une valeur de type chaine de caractère peut être composée , au maximum, de 255 caractères. Le type chaine de caractère est en réalité un tableau de plusieurs caractères (un type structuré qu'on étudiera par la suite).*

*Les* ***fonctions*** *les plus utilisées sur les chaines sont la fonction* ***concat*** *(caoncaténer : assembler) exemple :concat( 'bon' , 'jour') va assembler les deux chaines 'bon' et 'jour' pour donner en résultat la chaine 'bonjour'. La fonction* ***length*** ***(****qui donne la longueur de la chaine****),*** *exemple : length ('27 rue de la paix')=17. D'autres fonctions sont proposées par la majorité des langages (copy , delete, insert,…etc.) et on peut manipuler les chaines comme on manipule les variables de type tableau.*

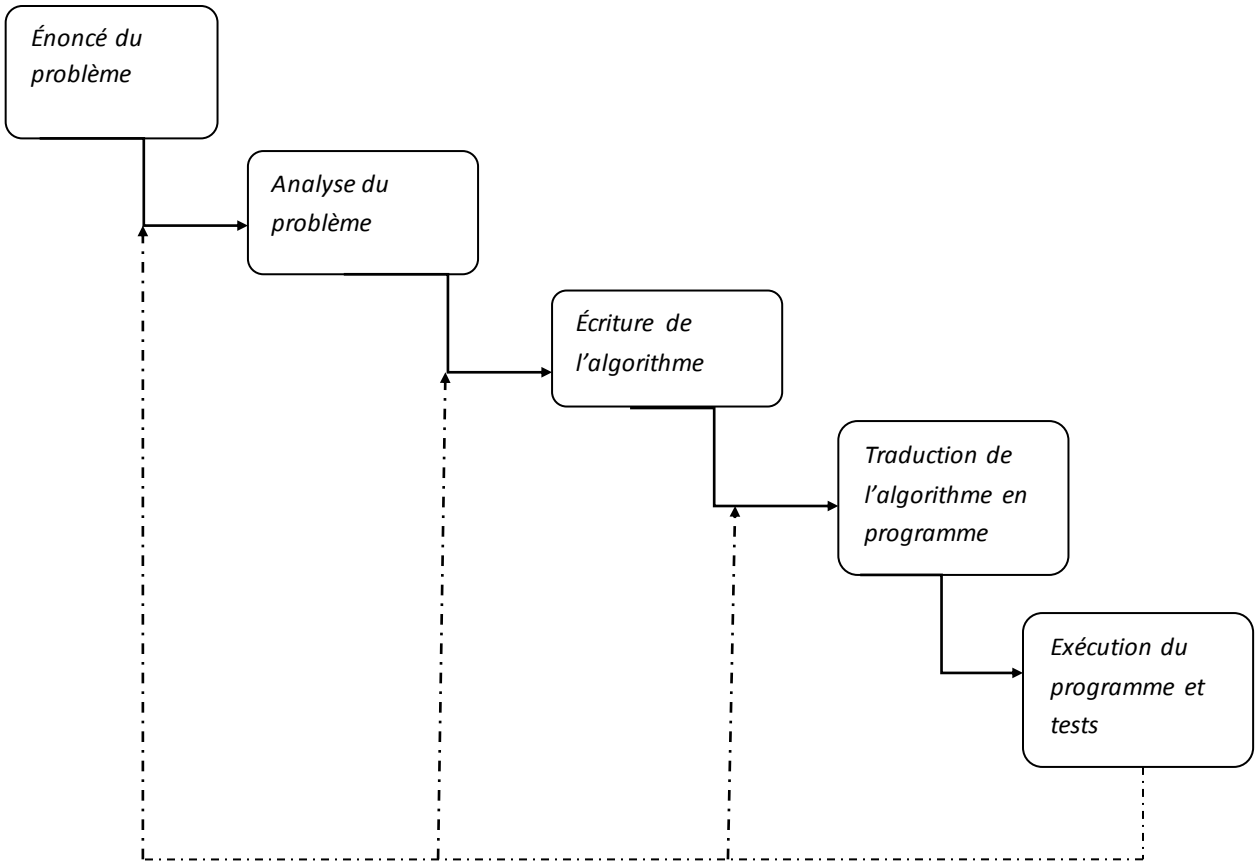
## Le type Logique (Booléen)

***Les seules valeurs*** *du type logique sont la valeur vrai ( ou bien 1)et la valeur faux (ou bien 0) ; ce sont des valeurs logiques. Les* ***opérateurs*** *utilisés sur ces valeurs sont les opérateurs logiques (et, ou , non) dont les résultats obéissent aux tables de vérité suivantes*

| A | B | A OU B | A ET B | NON A |
|---|---|---|---|---|
| Faux | Faux | Faux | Faux | Vrai |
| Faux | Vrai | Vrai | Faux | Vrai |
| Vrai | Faux | Vrai | Faux | Faux |
| Vrai | Vrai | Vrai | Vrai | Faux |

## 3/ Étapes de construction d'un programme

*Le schéma suivant montre les étapes de construction d'un programme.*



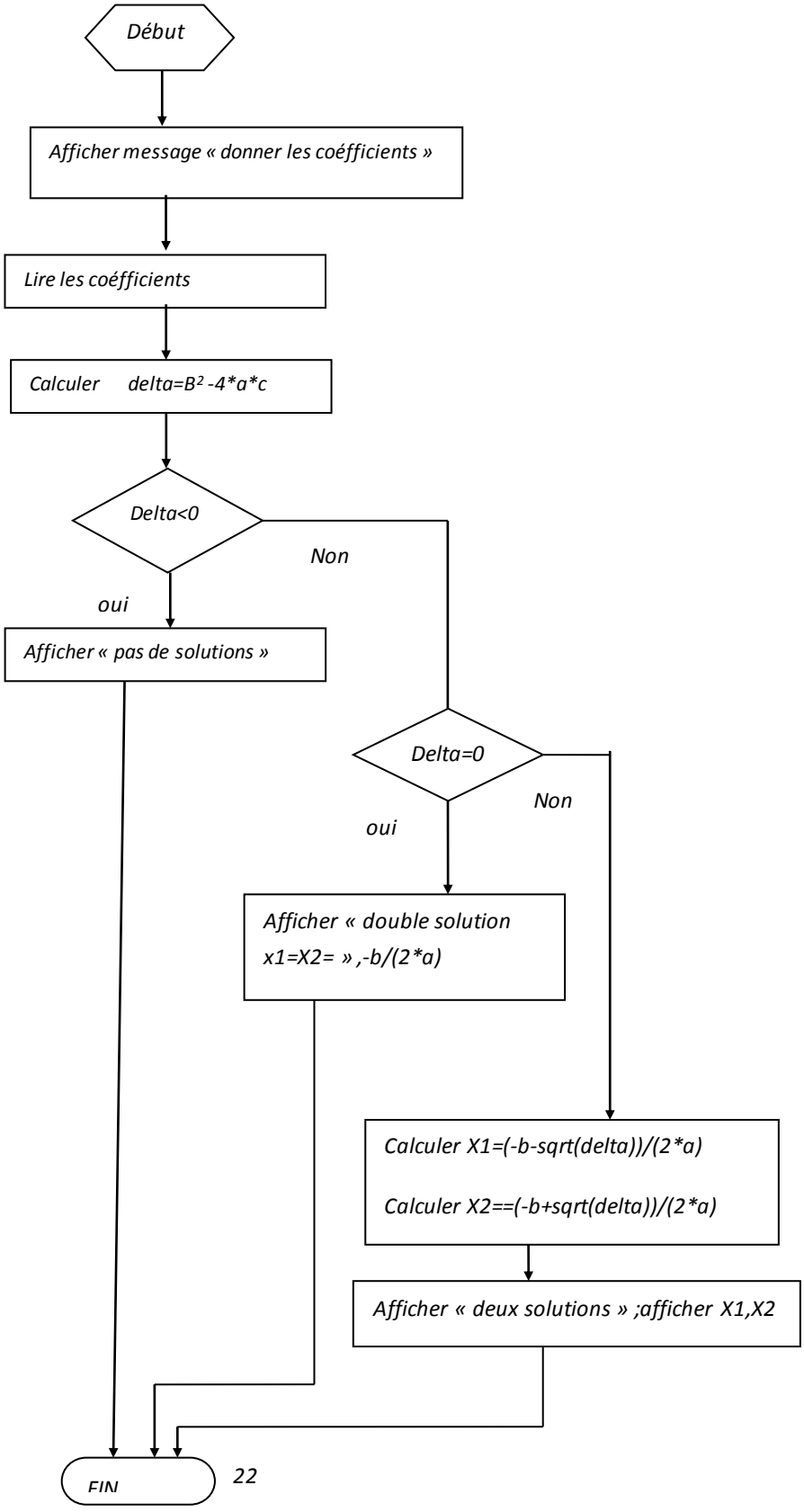

*L'analyse consiste à monter, à partir de l'énoncé, quelles sont les données, les résultats, les résultats intermédiaire si besoin et les traitements à faire sur les données pour aboutir aux résultats*

*Lors des tests, en cas d'anomalies, c'est-à-dire si le programme ne fonctionne pas correctement, on revient aux différentes étapes pour apporter des correctiosn, c'est ce qui est représenté par les flêches en pointillés*

## 4/ Représentation d'un algorithme en organigramme

*Un algorithme peut être représenté graphiquement à l'aide d'un organigramme :*

***Organigramme correspondant à l'algorithme de résolution d'équation du second degré.***



# CHAPITRE III : LES INSTRUCTIONS DE BASE (Affectation, lecture, écriture)

## 1/ Notion de variable en informatique

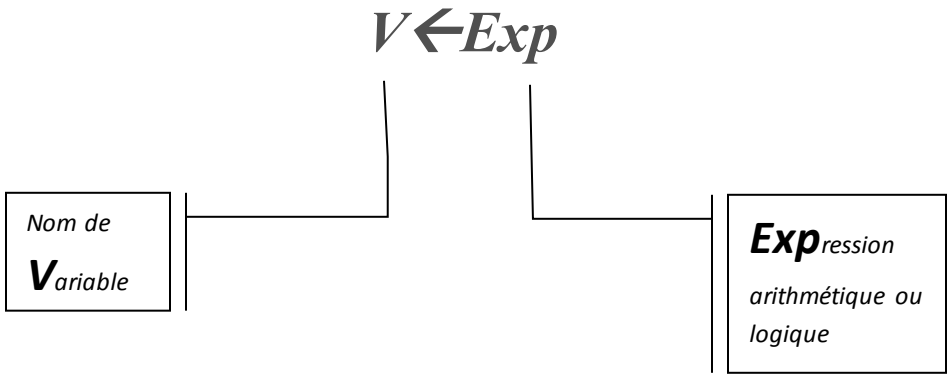
*Une variable en informatique est une case mémoire qui peut contenir une valeur. On l'appelle variable car la valeur de la case mémoire varie (change) au cours du temps (pendant l'exécution du programme)*

*La variable a un identificateur (nom) qui permet de la désigner ; un type, la connaissance du type permet au compilateur (langage) de vérifier la validité de la valeur donnée à la variable et la validité des opérations effectuées sur cette variable, et permet aussi au processeur d'utiliser les méthodes adaptées à chaque type, vu leur codage les différents types ne sont pas traités de la même manière. En général, un nom de variable est constitué d'une ou plusieurs lettres, les chiffres sont aussi permis à condition de ne pas apparaître en premier : exemple : N1 est un nom de variable valide (accepté) 1N est un nom de variable invalide (non accepté)*

## 2/ L'instruction d'affectation

*L'instruction d'affectation permet de mettre une valeur dans une variable*

***Syntaxe :***




*Après vérification de la compatibilité des types de V et de Exp, l'expression Exp est calculée et le résultat trouvé est mis dans la variable V.*

***Exemple*** *: Soient N1 et N2 deux variables entières ,Moyenne Une variable réelle*

***N1←15*** *est une instruction d'affectation valide, lorsque le processeur exécute cette instruction, il met la valeur 15 dans la variable (case mémoire) qui a pour nom N1.*

***Moyenne ← (N1+N2)/2*** *est une instruction d'affectation valide, l'exécution de cette instruction se fait en deux étapes :*

*1°/ Calculer (N1+N2)/2*

*2°/ Mettre le résultat dans la variable moyenne*

***N1←'note'*** *est une affectation invalide : les types de la variable N1 et de la constante 'note' sont incompatible ; 'note' est une constante de type chaine de caractère.*

***N1+N2←Moyenne :*** *est syntaxiquement incorrect ; la partie gauche de l'affectation doit obligatoirement être un nom de variable. N1+N2 n'est pas un nom de variable c'est une expression.*

***Exercice :***

*Qu'obtiendra -t-on dans les variables N1 et N2 après l'exécution des instructions suivantes*

| *1er Cas* | *2ème Cas* |
|---|---|
| *N1←5* | *N1←5* |
| *N2←7* | *N2←7* |
| *N1←N2* | *N2←N1* |
| *N2←N1* | *N1←N2* |

*Comment faire pour pouvoir échanger les valeurs des deux variables N1 et N2.*

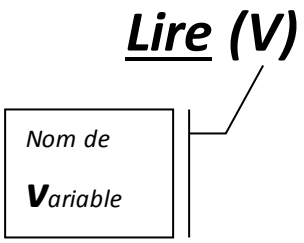
*Réponse : utiliser une troisième variable pour sauvegarder la valeur de N1.*

## 3 / L'instruction de lecture

*L'instruction de lecture permet de récupérer une valeur introduite au clavier (ou un autre périphérique d'entrée ) et de la mettre dans une variable.*

*Remarque :On considèrera pour la suite que le périphérique d'entrée est le clavier.*

***Syntaxe :***




*Lorsque l'instruction lire est exécutée, le processeur se met en attente de la frappe d'une valeur au clavier, une fois la valeur saisie, cette dernière est stockée dans la variable V ; et ceci après vérification de compatibilité des types de V et de la valeur saisie.*

***Exemple*** *: Soient les variables nom de type chaine ; note de type réel et N de type entier.*

***Lire (Nom)*** *: est une instruction de lecture valide ; l'exécution de cette instruction affiche à l'écran un curseur clignotant ou un point d'interrogation signifiant que le programme est en attente de l'introduction d'une valeur ; une fois que l'utilisateur a introduit sa valeur et si le type de cette dernière est compatible avec le type de la variable Nom, la valeur est stockée dans Nom ; le type de la valeur introduite doit donc être de type chaine de caractère.*

***Lire (Note) :*** *Il se passera la même chose que pour lire(Nom) sauf que cette fois ci le type de la valeur saisie doit être réel.*

***Remarque1 :*** *On peut lire plusieurs variables dans une même et seule instruction lire :*

***Exemple*** *:*

***Lire (Nom,Note)*** *est équivalent à la suite d'instructions lire(Nom) ; Lire (Note).*

*La première valeur saisie est stockée dans la variable Nom et la deuxième dans la variable Note.*

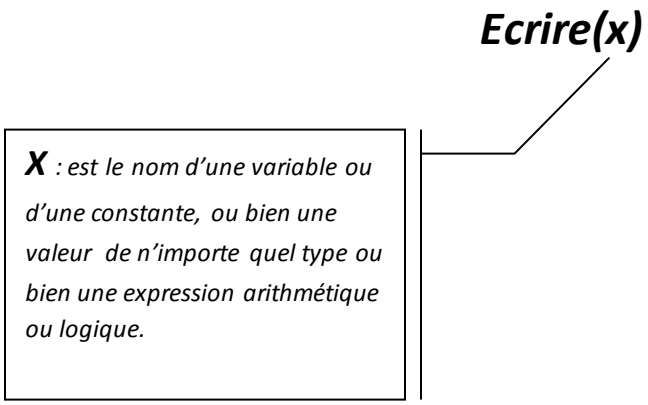
***Remarque2 :***

***On ne peut pas lire une variable de type énuméré, ni une constante de n'importe quelle type. Exemple : lire(2) ou bien lire('note') sont des instruction invalides.***

## 4/ L'instruction d'écriture

*L'instruction d'écriture permet d'afficher sur un périphérique de sortie (en général l'écran) les valeurs de variables, de constantes ou d'xpressions.*

***Syntaxe :***




***Exemple :***

*Ecrire('Bonjour') : l'exécution de cette instruction affichera à l'écran la valeur 'Bonjour' qui est de type chaine de caractère.*

*Ecrire(N1) : affiche la valeur de la variable N1*

*Ecrire ('N1') : affichera la valeur 'N1' qui est de type chaine de caractère.*

*Ecrire(2*N1+6) : affichera le résultat de l'expression donnée entre parenthèse)*

***Remarque** :*

*On peut afficher plusieurs valeurs avec une seule instruction Ecrire*

*Exemple :*

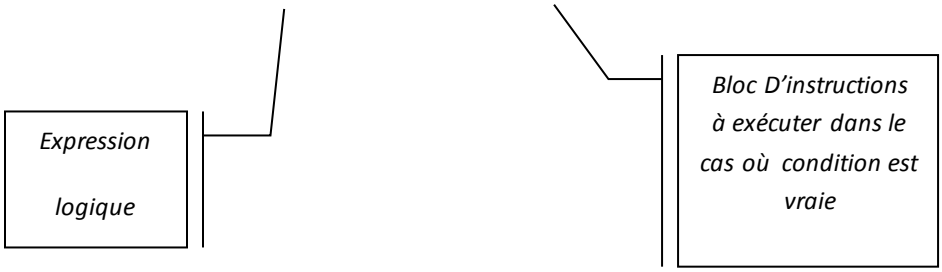
*Ecrire('bonjour', 'N1=', N1)*

# CHAPITRE IV : LES STRUCTURES DE CONTROLE (L'alternative, la répétition)

## 1/ L'alternative

### L'alternative simple

*Syntaxe :* ***Si*** *Condition* ***alors*** *instructions* ***Fsi***




***Exemple*** *: Soient deux nombres X et Y ranger dans X la plus petite valeur et dans Y la plus grande valeur.*

*Dans cet exemple si au départ X contient déjà la plus petite valeur , on ne fais rien, C'est seulement dans le cas contraire (X>Y) qu'il faut inverser les valeurs de X et Y.*

*Le traitement sera :*

***Si*** *X>Y* ***alors***
*Z←X ; X←Y ; Y←Z*
***Fsi***

### L'alternative complète

*Syntaxe :* ***Si*** *condition* ***alors*** *instructions1* ***sinon*** *instructions2* ***Fsi***


*Exemple : soit un nombre N non nul, dire si ce nombre est positif ou négatif*

***Si*** *N > 0* ***alors*** *ecrire ('positif')* ***sinon*** *ecrire ('négatif')* ***Fsi***

## L'alternative imbriquée

*Dans une alternative complète, on peut avoir parmi les instructions du bloc d'instructions2 une autre alternative, la syntaxe devient alors la suivante :*

***Si** condition1 **alors** Instructions1 **sinon***

***Si** condition2 **alors** instructions2 **sinon***

***Si** condition3 **alors** instructions3 **sinon** instructions4 **Fsi***

***Fsi***

***Fsi***

***Exemple :***

*Afficher l'etat de l'eau (glace, liquide, vapeur) suivant la température donnée.*

*Si t (la température ) est supérieur à 100 alors on affiche vapeur, sinon il reste deux cas : 1er cas inférieur à 0 ; 2ème cas compris entre 0 et 100 donc il faut tester de nouveau ; le traitement sera donc comme suit :*

***Si** t>100 **alors** ecrire ('vapeur') **sinon***

***Si** t<0 **alors** ecrire ('glace') **sinon** ecrire('liquide') **Fsi***

***Fsi***

## Le choix multiple

*Syntaxe :*

***Selon** V **vaut** :*

*listeVal1 : Instructions1*

*listeVal2 : Instructions2*

*….*

*listeVal n : Instructions n*

***Autre** : Instructions*

***Fin***

***V** : est un nom de variable ;*

***Listevali** : sont des valeurs séparés par des virgules ou bien des intervalles de valeurs*

*Remarque : ces valeurs doivent être du même type que V*

***Exemple :***

*Ecrire un algorithme qui réalise au choix la somme , le différence, la multiplication, la division de deux nombres donnés.*

```
Algorithme Choix_Operations
Var A, B, resultat : REEL
    Choix : CARACTERE
Début
 ECRIRE ('1- Addition')
 ECRIRE ('2- Multiplication')
 ECRIRE ('3- Différence')
 ECRIRE ('4- Division')
 ECRIRE ('Donner votre choix (1 / 2 / 3 / 4) : ')
 LIRE(Choix)

 ECRIRE ('Donner le premier nombre : ')
 LIRE (A)
 ECRIRE ('Donner le deuxième nombre : ')
 LIRE (B)
 SELON Choix VAUT
  '1' : Début
          Resultat ← A+B
          ECRIRE('La somme des deux nombres est : ', Resultat)
        Fin
  '2' : Début
          Resultat ← A*B
          ECRIRE('Le produit des deux nombres est : ', Resultat)
        Fin
  '3' : Début
          Resultat ← A-B
          ECRIRE('La différence des deux nombres est : ', Resultat)
        Fin
  '4' : SI B ≠ 0 ALORS
           Resultat ← A/B
           ECRIRE('La division des deux nombres est : ', Resultat)
        SINON ECRIRE('La division par zéro est impossible')
        FSI
 Fin
Fin
```

## *Programme c++ choix multiple*

```
#include <stdio.h>
#include <stdlib.h>
main()
{ float a,b,resultat;
char choix;
 printf("1- Addition\n");
 printf("2- Multiplication\n");
 printf("3- Différence\n");
 printf("4- Division\n");
 printf("Donner votre choix 1 2 3 4 \n");
 scanf("%c",&choix);
 printf("Donner le premier nombre :\n");
 scanf("%f",&a);
 printf("Donner le deuxième nombre :\n");
 scanf("%f",&b);
 switch (choix)
   {
     case '1':{ resultat=a+b;
          printf("La somme des deux nombres est : %f\n", resultat);
          break;}
     case '2':{ resultat=a*b;
          printf("Le produit des deux nombres est :%f\n ",resultat);
          break;}
     case '3':{ resultat=a-b;
          printf("La différence des deux nombres est :%f\n",resultat);
          break;}
     case '4' :{if(b!=0)
           {resultat=a/b;
           printf("La division des deux nombres est :%f\n",resultat);}
           else printf("La division par zéro est impossible\n");
           break;
           };
     default: {printf("code opération incorrect\n");}
     }
system ("pause");
return 0; }
```

# 2/ La répétition

*Les structures de répétition sont les structures les plus puissantes de la programmation, elles permettent d'exécuter plusieurs fois un même bloc d'instructions, on distingue les répétitions conditionnelles et les répétitions avec compteur.*

## Les répétions conditionnelles

*La poursuite de la répétition du bloc d'instruction dépend d'une condition qui peut être examinée soit avant le bloc d'instruction (répétition **tant que**) ou bien après le bloc d'instructions (répétition **répéter jusqu'à**).*

### 1. Répétition « Tant que »

*Syntaxe :*

***Tant que** condition **Faire** Instructions **Fin tantque***


***Exemple** :Devinette*

*Ecrire un algorithme qui demande à l'utilisateur de deviner un nombre et tant que ce nombre n'est pas trouvé, afficher « non ce n'est pas ça, recommencez »et lui permettre de donner un autre nombre.*

***Algorithme principal** devinette ;*

```
Const nb_secret=54 ;        // 54 à titre d'exemple, on peut prendre n importe quel autre nombre
Var n : entier ;
Début
Ecrire( « devinez le nombre secret ») ;
Lire(n) ;
Tant que (n<>nb_secret) faire
Ecrire (« non c n est pas cela recommecez ») ;
Lire(n) ;
Finftantque
Ecrire(« bravo vous avez trouvez ») ;
Fin
```

Programme c++ devinette

```
#include<stdio.h>
#include<stdlib.h>
main()
{const int nb_secret=54  ; int n;
printf("devinez le nombre secret\n");
scanf("%d",&n);
while (n!=nb_secret)
{ printf("non ce n est pas cela recommecez  \n");
scanf("%d",&n);
}
printf("bravo vous avez trouvez\n");
system("pause");
return 0;
}
```

## 2. Répétition « Répéter …. jusqu'à »

### Syntaxe :

**Répéter** Instructions **Jusqu'à** condition

**Exemple1**

Ecrire un algorithme qui demande un nombre et affiche son carré jusqu'à ce que le nombre donné soit égal à zéro :

```
Algorithme principal exemple1
Var n :réel ;
Début
Répéter
Ecrire(« donner n ») ;
Lire(n) ;
Ecrire (« carré de n : »,n*n)
Jusqu'à (n=o)
FIN
```

```
#include<stdio.h>
#include<stdlib.h>
main()
{ float n;
Do
printf("donner n\n");
scanf("%f",&n);
printf("carré de n: %f \n", n*n);
while (n!=0)
system("pause");
return 0;
}
```

**Exemple 2**

Calculer la moyenne de dix notes données.

```
Algorithme principal exemple 2 ;
Var note,s,moy :réel ;  i :entier ;
Debut
Repeter
Lire(note);
S=s+note;
I=i+1;
Jusqu'à (i>10);
Moy=s/10;
Ecrire("moyenne=",moy);
Fin
```

```
#include<stdio.h>
#include<stdlib.h>
main()
{ float note,s,moy;  int i;
Do
scanf("%f",&note);
s=s+note;  i=i+1;
while (i<=10)
moy=s/10;
scanf("%f",&note);
s=s+note;  system("pause");
return 0;
```

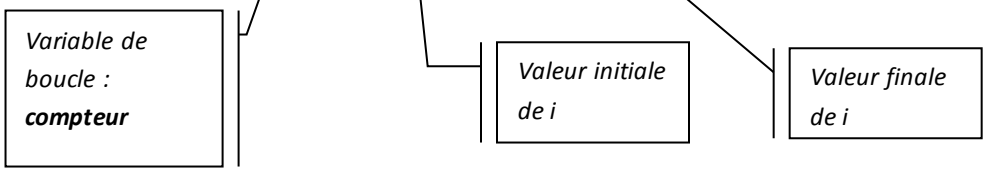
## La répétion avec compteur

***L'instruction « Pour »***

*Cette instruction utilise un compteur qui commence à compter à partir d'une valeur initiale jusqu'à une valeur finale et à chaque valeur, les instructions sont répétées. Cette instruction est utilisée lorsque le nombre de répétitions est connu à l'avance.*

## *Syntaxe :*

***Pour** i **de** exp1 **à** exp2 **faire** Instructions **Finpour***




***Exemple :*** *On reprend l'exemple 2 précédent (moyenne de 10 notes)*

*Au lieu de repeter jusqua on aura :*

```
.....
pour i de 1 à 10 faire
Lire(note);
S=s+note;
Finpour
Moy=s/10
Ecrire(moy)
.....
```

```
.....

For(i=1 ;i<=10 ;i++) ;

{ scanf("%f",&note);
s=s+note;
}
Moy=s/10
scanf("%f",&note);
s=s+note;
.....
```

*Remarque : La variable de boucle ne doit jamais être changé dans le bloc d'instructions de la boucle.*

# CHAPITRE V : LES SOUS PROGRAMMES : (Les procédures et les fonctions)

## 1/ Notion de sous programme

*Une des méthodologies appliquées en informatique est de décomposer un gros problème en plusieurs sous problème indépendants et d'écrire des algorithmes pour chacun de ses sous problème, ces algorithmes sont appelés des sous programme, un sous programme est une suite d'instructions nommée. Le programme principal ou algorithme principal utilise ces sous programmes en les appelant par leur nom comme il utilise les instructions. Les instructions lire et ecrire sont d'ailleurs des sous programmes standards. Un sous programme peut aussi appeler un autre sous programme, on les appelle alors l'appelant et l'appelé.*

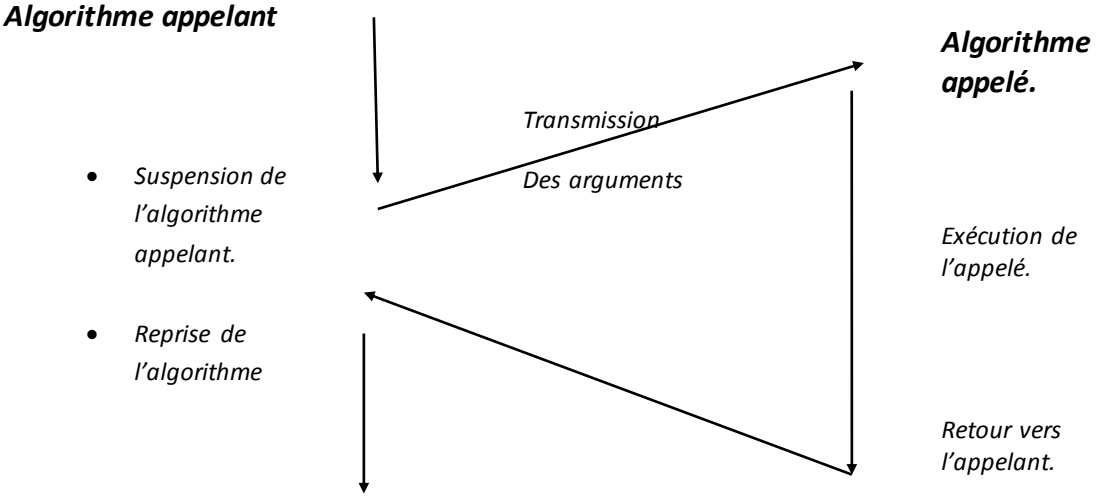



*Fig 5.1 : Mécanisme d'appel d'un sous programme*

*Structure d'un algorithme utilisant un sou-programme :*

*…….*

*Il existe deux types de sous programme : les procédures et les fonctions.*

## *2/ Les procédures*

### *Définition d'une procédure*

### *Syntaxe :*

***procédure*** *nom-de –procédure (paramètres formels) ;*

*Déclaration de Variables et de constantes ;*

***Début***

*Instructions*

***Fin*** *;*

*Les paramètre sont des variables donnée, résultat ou donnée et résultat à la fois.*

*Voir explication ci-dessous.*

### *Appel de procédure*

*L'appel de procédure se trouve parmi les instructions de l'algorithme principal.*

### *Syntaxe :*

*Nom-de-procédure (paramètres effectifs)*

*Le nombre des paramètres effectifs doit être le même que celui des paramètres formels ; et les types compatibles respectivement à l'ordre.*

### *Les paramètres :*

*Les paramètres se trouvant dans la définition de la procédure sont appelés les paramètres formels, on précise leur type (donné, résultat ou donnée /résultat) respectivement par les symboles suivants (🡻, 🡹, 🡻🡹).*

- *Un paramètre donnée est utilisé comme donnée d'entrée à la procédure, sa **valeur** est transmise par l'algorithme principal lors de l'appel par l'intermédiaire du paramètre effectif correspondant. Le paramètre effectif peut donc être une variable, une constante ou une expression ,tous ces objets peuvent véhiculer une **valeur**.*

- *Un paramètre résultat est utilisé pour stocker un résultat de la procédure, il est transmis à l'algorithme principal par la procédure qui le stocke en même temps dans le paramètre effectif correspondant ; ce paramètre effectif doit obligatoirement être une variable, c'est le seul objet dans lequel on peut stocker des valeurs.*
- *Un paramètre donnée/résultat joue en même temps les deux rôles.*

***Exemple 1:***

*Ecrire une procédure qui affiche le maximum de deux nombre.*

```
Procédure affiche_max(↓x,y :réel) ;
Début
Si x>y alors ecrire('le maximum est :',x) sinon ecrire('le
maximum est :',y) Fsi
Fin.
Algorithme principal maximum ;
Var a,b :réel ;
Procédure affiche_max ;  //déclaration de la procédure
Début
Ecrire('donnez deux nombres') ;
Lire(a,b) ;
Affiche_max(a,b) ;          //Appel de la procédure
Fin.
```

```
#include<stdio.h>
#include<stdlib.h>
void affiche_max (float x,float y)
{if( x>y) printf(" le max est :%f",x);
else printf(" le max est :%f",y);}

int main()
{float a,b;
printf("donnez deux nombres");
scanf("%f%f",&a,&b);
affiche_max(a,b);
system("pause");
}
```

***Exemple2 :*** *ecrire une procédure qui renvoie le maximum de deux nombres.*

```
Procédure renvoie_max( ↓x,y :rée ;, ↑max :réel) ;

Début

Si x>y alors max ← x sinon max ← y fsi ;

Fin ;

Algorithme principal maximum ;

Var a,b ,m:réel ;

Procédure renvoie_max ;      //déclaration de la procédure

Début

Ecrire('donnez deux nombres') ;

Lire(a,b) ;
```

```
renvoie_max(a,b,m) ;      //Appel de la procédure
ecrire('le maximum est :',m) ;
Fin.
```

## 3/ Les fonctions

*Les fonctions sont des procédures qui renvoient un seul résultat ( qui ont un seul paramètre résultat) et qui ont obligatoirement des paramètres d'entrée.*

### Définition de fonctions

### Syntaxe :

**Fonction** Nom_de_fonction(paramètres) :type de fonction

Déclaration de Variables locales

**Début**

Instructions ;

Nom_de_fonction ← résultat //le résultat est obligatoirement affecté au nom de la fonction

**Fin ;**

### Appel de fonction :

*Comme une fonction renvoie un résultat unique, elle peut dons être utilisée comme on utilise une variable dans une expression. L'appel de fonction se trouve dans les expressions.*

*Exemple : ecrire une fonction qui renvoie la factorielle d'un nombre donné.*

```
Fonction fact (n :entier) :entier ;
Var i,f :entier ;
Début
f ← 1
```

***Pour** i **de** 1 **à** n **faire** f←f*i **Finfaire***

*Fact ← f ;*

*Fin ;*

*Ecrire un algorithme qui affiche la factoriel d'un nombre donné en utilisant la fonction fact.*

***Algorithme principal** usefact ;*

***Var** nb :**entier** ;*

*Fonction fact :entier ;*

***Début***

***Ecrire**('donnez un nombre entier') ;*

***Lre**(nb) ;*

***Ecrire**(nb,' ! =', fact(nb)) ;*

***Fin**.*

***Programme c++ Factoriel avec fonction fact.***

```
#include<stdio.h>
#include<stdlib.h>
int fact (int n)
{int i,f;
f=1;
for (i=1;i<=n;i++)
 f=f*i;
return (f) ;
}

int main()
{int nb ;
printf("donnez un nombre entier\n") ;
scanf("%d",&nb) ;
printf("nb! =%d \n", fact(nb)) ;
system("pause");
}
```

# CHAPITRE VI : LES TYPES COMPOSÉS (Les tableaux et les enregistrements)

## 1/ Type simple et type composé

Les types utilisés dans les chapitres précédents (entiers, réels, caractères et booléen) sont appelés des types simples car une variable de l'un de ces type ne peut contenir qu'une seule valeur à la fois ; les types composés définissent des variables composées de plusieurs parties (appelées éléments pour les tableaux ou champs pour les enregistrements).

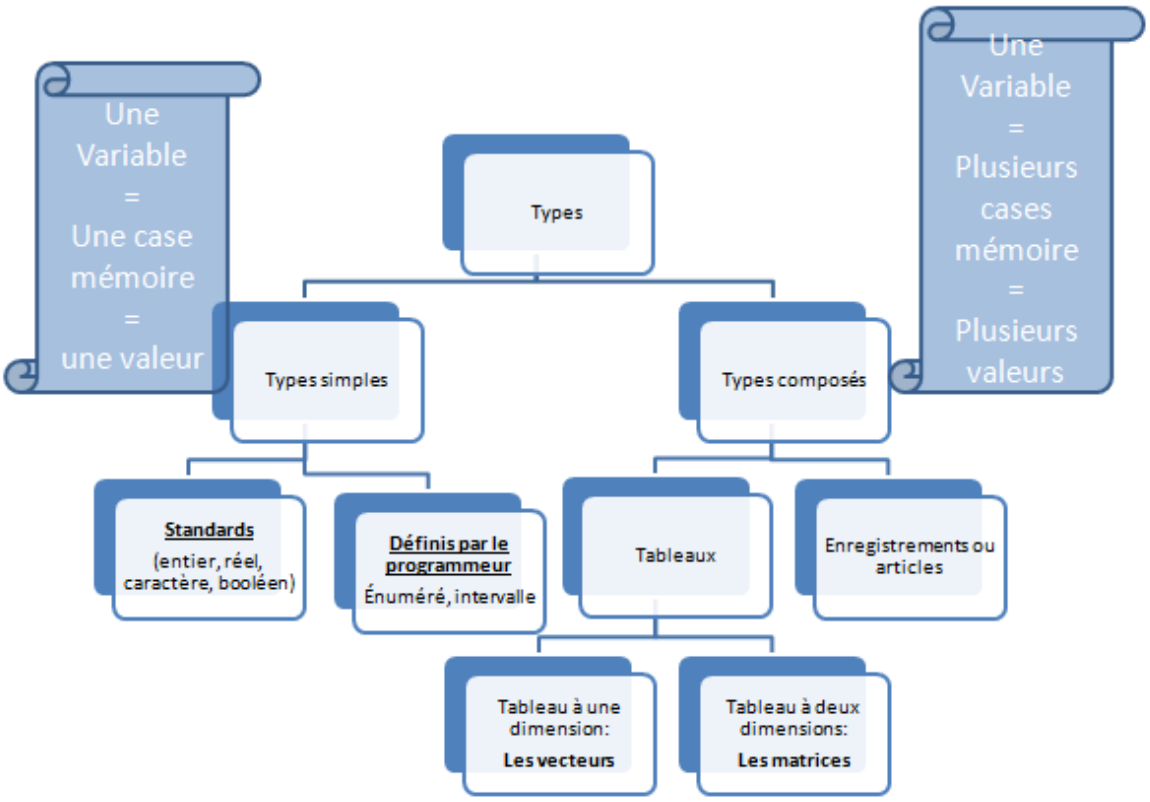



*Fig.6.1 : Types simples et types composés*

# 2/ Les tableaux

Une variable de type tableau est une variable composée de plusieurs variables de **même type.**

**Problème :**

- On veut écrire un programme qui stocke et traite les températures du mois de mai; il nous faut pour cela stocker les 31 valeurs correspondantes aux températures de chaque jour du mois de mai.

On peut déclarer **31 variables** de type réel (valeurs des températures) ;
ce qui donne:

**Var**
t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15,t16,t17,t18,t19,t20,t21,t22,t23,t24,t25,t26,t27,t28,t29,t30,t31: réel;

Et pour les lire:

**lire** (t1,t2,t3,t4,t5,t6,t7,……,t30,t31) ;

Et si on veut stocker les températures de tous les jours de l'année, il nous faudra **365 variables** !

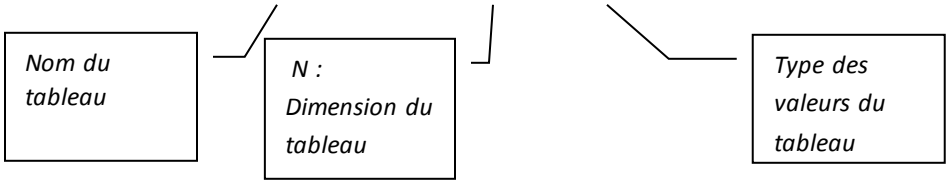
Le type tableaux à une dimension (appelé aussi vecteur) est là pour régler ce problème:

## Tableau à une dimension

**Déclaration d'une variable de type tableau à une dimension**

**Syntaxe :**

**Var** nomvar **: tableau [**1..N**] de type de base**




Pour notre exemple des températures du mois de mai, nous allons déclarer 31 variables en une,
Comme ceci:

**Var** t:tableau [1..31] de réel;

Et pour les lire: nous utilisons une boucle;

**Pour** i de 1 à 31 **faire lire**(T[i]) **finfaire**

```
Float  t [31] ;   int I;

For (i=0 ;i<31 ;i++)

Scanf  (« %f »,&t[i] )
```

## *Tableau à deux dimensions*

Un tableau à deux dimensions est un tableau où chaque élément est lui-même un tableau à une dimension, c'est un tableau de tableaux .

*Déclaration d'une variable de type tableau à deux dimensions :*

**Syntaxe :**

**Var** nomvar **: tableau [**1**..**N**,** 1**..**M**] de type de base.**

*Problème :*
*On veut maintenant stoker les températures des 12 mois avec chaque mois 30 ou 31 jours. Il nous faudra donc au maximum 12*31 cases mémoires (variables);*
*On sait déjà déclarer 31 cases :*

***Var T :tableau[1..31]de réel;***

Il nous faut maintenant déclarer 12 tableaux de 31 cases chacun:
Ce qui donne:
**Var T :Tableau [1..12,1..31] de réel,**
On dit que la première dimension =12
Et la deuxième dimension=31

```
Float t[12][31] ;
```

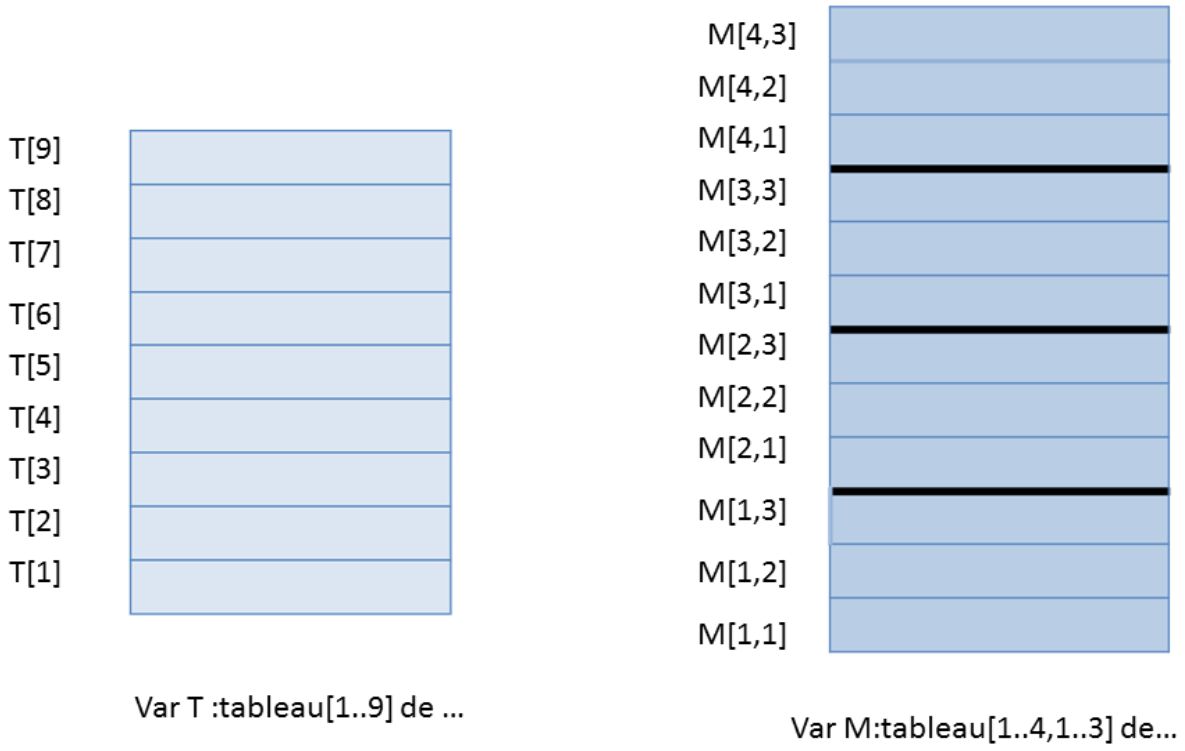
Et pour lire ces 12*31 valeurs de températures:

```
Pour i de 1 à 12 faire
Pour j de 1 à 31 faire
Lire (T[i,j])
Ffp                ffp (fin faire pour=fin de la boucle j)
Ffp                        ffp (fin faire pour=fin de la boucle i)
```

```
For (i=0 ;i<31 ;i++)

For (j=0 ;j<31 ;j++)

Scanf (« %f »,&t[i][j] )



//il faut bien sur déclarer i et j
```

### *Représentation des tableaux en mémoire*

*Exemple de représentation en mémoire d'un tableau T à une dimension égale à 9 et d'un tableau M à deux dimensions égales respectivement à 4 et 3*




## *3/ Les enregistrements ou articles*

Les tableaux nous permettent de regrouper plusieurs informations de même type dans une seule variable. Lorsque les informations ne sont pas toutes de même type mais concernent un même objet; nous pouvons alors déclarer cet objet comme étant de type article composé de plusieurs champs.

*Déclaration d'un type enregistrement :*

**Syntaxe :**

**Type** nomtype **= article**

Nomchamp1 **: Type**

Nomchamp2 **: Type**

**……**

Nomchampn **: Type**

**Fin**

*Les types des différents champs peuvent être différents et peuvent être des types simples, standards énumérés ou intervalle ou bien des types composés.*

**Exemple :**

Nous voulons écrire un programme qui gère les clients d'une banque et leurs comptes.

Pour la banque un client ou un compte bancaire comporte beaucoup d'informations:

- Le numéro du compte
- Le nom et prénom du client
- L'adresse du client
- L'avoir (solde dans le compte)

- Pour pouvoir stocker toutes ces informations en mémoire de façon organisée, nous déclarons un nouveau type :

**Type** client=**article**

Num,Nom,pren,adr: **chaine de caractère**;

Avoir: **réel**;

**Fin**;

```
Typedef struct
{ char num[30] ; char nom[30] ;
  char pren[30] ; char adr[60] ;
 float avoir ;} client ;
```

Le type client contient plusieurs parties (num, nom,pren,adr et avoir) appelées champs.

Nous pouvons maintenant déclarer une variable cl de type client:

**Var** cl: client;

```
Client  cl ;
```

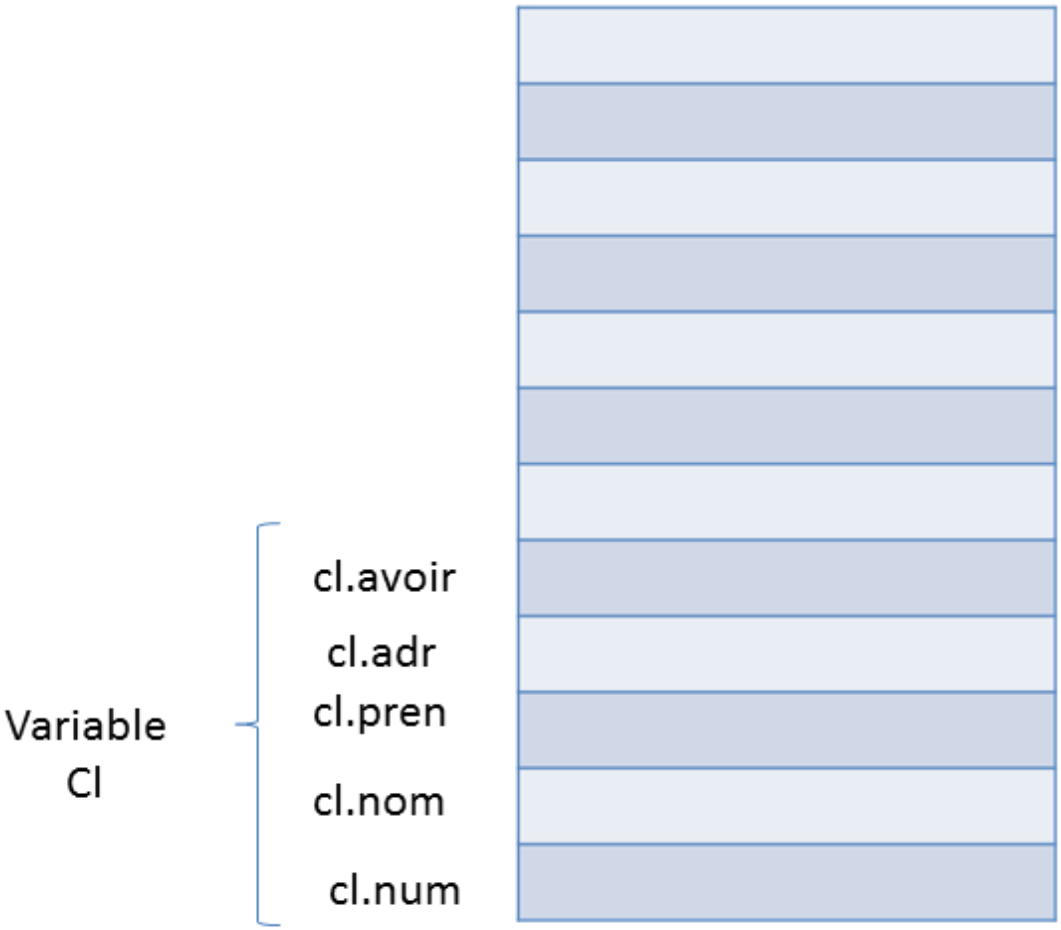
Et pour lire les informations d'un client, nous écrirons les instructions suivantes dans notre algorithme.

Ecrire ('donnez le nom,prénom, adresse et avoir du client'); Lire (cl.nom, cl.pren, cl.adr, cl.avoir)

```
Printf(« donnez le nom,prénom, adresse et avoir du client ») ;
Scanf (« %s%s %s %f\n », &cl.nom, &cl.pren, & cl.adr, &cl.avoir);
```

## *Représentation d'un article en mémoire*

La variable article cl sera stockée en mémoire comme suit :



# CHAPITRE VII : NOTIONS AVANCEES

## 1/ Les pointeurs

Il est possible dans certain langages de programmation comme en c++, de manipuler des variables en utilisant non pas leurs identificateurs (noms) mais directement leurs adresses mémoire ; en effet, toute variable a une adresse en mémoire.

En c, il existe un type spécial appelé pointeur qui nous permet de travailler avec des adresses mémoire.

*Déclaration d'une variable de type pointeur :*

**Syntaxe :**

**Var** nomvar **: * type** en **c++** **type *** nomvar

Exemple :

```
Algorithme principal pointeur;
Var a,b : entier;
P,q : * entier;     // p et q sont des variables de type pointeur sur des entiers
                    Càd elles contiendront des adresses mémoire de variables de type entier
Debut
 A← 2 ;  b← a+3 ;
P ← & a ;           // le symbole & signifie adresse   (&a veut dire adresse mémoire de la variable a)
Ecrire ('*p=',  *p , 'a=', a, 'b=', b) ;       // *p signifiie le contenu de la variable qui a pour adresse p
*p ← *p+1 ;  Ecrire ('*p=' , *p, 'a=', a);
 q ← p;   ecrire ( '*p=', *p, '*q=', *q) ;
q ← &b ;    ecrire ('*p=', *p,'*q=', *q) ;
fin.
```

Programme pointeur c++

```
#include<stdio.h>
#include<stdlib.h>
int main()
{int a,b;
int* p ; int*q;
a=2; b=a+3; p=&a;
printf("*p=%d   a=%d  b=%d\n",*p,a,b);
*p=*p+1;     printf("*p=%d   a=%d \n",*p,a);
q=p;  printf("*p=%d   *q=%d \n",*p,*q);
p=&b;  printf("*p=%d   *q=%d \n",*p,*q);
system("pause");
return  0; }
```

Ecran d'exécution du programme :

```
*p= 2      a=2       b=5

*p=3      a=3

*p=3      *q=3

*p=5      *q=3
```

# 2/ La récursivité

*En programmation, une fonction récursive est une fonction qui s'appelle elle-même, la récursivité permet de résoudre de manière efficace et rapide certains problèmes.*

*Exemple : la fonction factorielle.*

*On a* $n! = \underbrace{1*2*3*4*\ldots\ldots\ldots*(n-1)}_{(n-1)\,!}*n$ *Et* $(n-1)\,! = 1*2*3*4*\ldots\ldots\ldots*(n-1)$

*Donc* $n\,! = (n-1)\,! * n$

*Et* $(n-1)\,! = (n-2)\,! * (n-1)$

*Etc…*

## Autre écriture de la fonction factorielle utilisant la récursivité.

```
Fonction fact (n :entier) : entier ;

Debut

Si (n=1) alors fact ← 1 sinon fact ← fact (n-1) *n;    fsi;

Fin
```

*Programme c++*

```
#include<stdio.h>
#include<stdlib.h>
int fact (int n)
{if (n==1) return(1); else return (fact(n-1)*n);
}
int main()
{int a;
printf("entrez un nombre entier\n");
scanf("%d",&a);
printf("factoriel de a=%d\n",fact(a));
system("pause");
return 0; }
```

## *Déroulement de la function récursive fact sur un exemple*

*Soit a=3*

*Fact(3) premier appel de fact avec n=3*

*If (n=1) return 1 else return ( fact (n-1)*n)*

*1/ n= 3 différent de 1 donc return (fact (3-1)*3 =fact(2)*3*

*pour calculer fact(3) on doit calculer fact(2)*

*Fact(2) deuxième appel de fact avec n=2*

*2/ n=2 different de 1 donc return (fact (2-1)*2)= fact(1)*2*

*pour calculer fact(2) on doit calculer fact(1)*

*Fact(1) troisième appel de fact*

*N=1 donc return 1*

*On revient en 2/et on remplace fact(1) par 1, on obtient fact(2)=1*2=2*

*Puis on remonte encore vers 1 et on remplace fact(2) par 2 et on obtient fact(3)=2*3=6*

# Annexe : Numérisation du son et de l'image

## 1/ La numérisation de l'image

### La couleur et la lumière

- La couleur que possèdent les objets qui nous entourent dépend de la lumière qu'ils diffusent.
- Rappel : lors du phénomène de diffusion un objet reçoit de la lumière et en renvoi une partie dans toutes les directions.
- Lors de la diffusion une partie de la lumière blanche reçue est absorbée tandis que l'autre est renvoyée et donne sa couleur à l'objet :
  Un objet rouge absorbe toutes les couleurs de la lumière blanche, sauf le rouge, un objet vert absorbe toutes les lumières sauf le vert, etc.
- Éclairé par une lumière blanche un objet possède la couleur de la lumière qu'il n'absorbe pas.
- la lumière blanche est un mélange de toutes les couleurs

### Le scanner

- Un scanner de documents est un appareil qui transforme une information physique en une information électrique qui est ensuite elle-même transformée en un fichier informatique.
- Le principe du scanner est de lire la surface d'un document (ligne par ligne) et d'y "mesurer" les couleurs et leur intensité ; ces informations sont alors traitées.
- Pour lire un document, il faut une source lumineuse, un capteur de lumière et le déplacement du capteur par rapport au document. Il n'y a alors que deux possibilités proches :
- Lire par transparence (pour les scanners de diapos et négatifs par exemple).
  - Lire par réflexion (pour les documents).
- Un capteur est un composant électronique photosensible servant à convertir un rayonnement électromagnétique (UV, visible ou IR) en un signal électrique analogique. Le signal électrique est ensuite converti en signal numérique grâce à un montage électronique appelé Convertisseur analogique numérique. (reçoit une tension en entrée et donne un nombre en sortie)
- La performance d'un scanner se mesure par sa vitesse et la qualité de la numérisation cette dernière dépend de la résolution c'est-à-dire en combien de point par pouce l'image est divisée ?
- Plus il ya de capteurs dans le scanner plus la résolution est grande car un capteur représente un point.
- Une lumière blanche est projetée sur l'image et cette dernière renvoie (par diffusion) des faisceaux de lumière de différentes couleurs et intensité.
- Si la résolution est de 300 dpi (dot per inch ou point par pouce(2.54cm) ; cela veut dire que il ya 300 capteurs de lumières élémentaires sur 2.54 cm. Couleurs primaires (rouge vert et bleu)

- *Si la largeur du scanner permet de numériser une feuille 21/29.7 alors il ya combien de capteurs sur la largeur du scanner ? 300 par 2.54 cm → 300/2.54=118.11par cm →118.11*21=2480 capteurs sur la largeur du scanner.*

- *Le point coloré est codé sur 24 bits (8 bits pour le rouge, 8 pour vert et 8 pour le bleu)*
- *8 bits permettent d'avoir 256 codes différents →256 nuances différentes ( intensités lumineuses) pour chaque couleur primaire (R V B).*

## RGB color table

Basic colors:

| Color | HTML / CSS Name | Hex Code #RRGGBB | Decimal Code (R,G,B) |
|---|---|---|---|
| | Black | #000000 | (0,0,0) |
| | White | #FFFFFF | (255,255,255) |
| | Red | #FF0000 | (255,0,0) |
| | Lime | #00FF00 | (0,255,0) |
| | Blue | #0000FF | (0,0,255) |
| | Yellow | #FFFF00 | (255,255,0) |
| | Cyan / Aqua | #00FFFF | (0,255,255) |
| | Magenta / Fuchsia | #FF00FF | (255,0,255) |
| | Silver | #C0C0C0 | (192,192,192) |
| | Gray | #808080 | (128,128,128) |
| | Maroon | #800000 | (128,0,0) |
| | Olive | #808000 | (128,128,0) |
| | Green | #008000 | (0,128,0) |
| | Purple | #800080 | (128,0,128) |
| | Teal | #008080 | (0,128,128) |
| | Navy | #000080 | (0,0,128) |

*FigA.1: Table des couleurs RGB*

# Sommaire