CHAPITRE

2 Représentation des données numériques

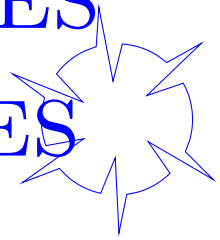


Table des matières

1 Représentation d'un nombre dans une base : 2
2 Les bases en informatiques : 3
3 Vocabulaire : 4
4 Représentation des entiers positifs 6
5 Addition de binaires 6
6 Représentation des entiers négatifs 8

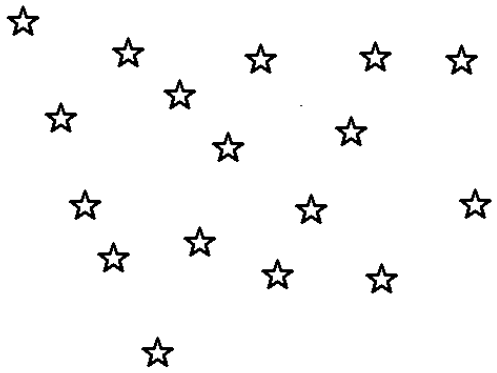
Il y a 10 sortes de personnes ; ceux qui comprennent le binaire et les autres...

Cette phrase bien connue des informaticiens peut surprendre en première lecture.Il faut comprendre qu'un nombre est différent de sa représentation et que selon la base choisie, il peut s'écrire de différentes façons. Mais qu'est ce qu'une représentation, qu'est ce qu'une base ? C'est l'objet de ce cours...

1 Représentation d'un nombre dans une base :

Nous l'avons tous oublié mais nous avons appris dans les classes de maternelles à regrouper les objets par paquets de dix . Pourquoi ? Sûrement parce que nous avons dix doigts...
Cela paraît naturel mais certaines civilisations n'avaient pas fait ce choix et nous en percevons encore les conséquences : les babyloniens (civilisation ancienne) découpaient le temps par paquet de 60(quelle idée !) et nos minutes comptent 60 secondes.
Un ordinateur (ou plutôt son processeur...) ne connaît que deux états physiques contradictoires(tension ou pas, fermé ou ouvert...) modélisés par les chiffres 0 et 1. De cette situation naquit le notation binaire dans une base 2.

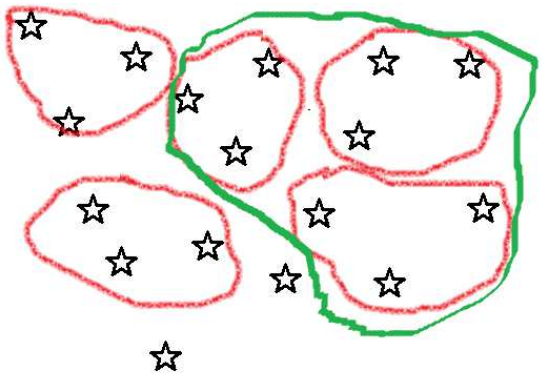
Voici 17 étoiles, vous pouvez compter si vous le souhaitez !



Mais que signifie ce nombre ? Simplement qu'il y a un paquet de 10 (l dizaine) et reste 7 étoiles. Donc naturellement le comptage a été fait en base 10 :

$$17 = 1 \times 10^1 + 7 \times 10^0$$

Mais que se passerait-il si on comptait en base 3 selon le même principe ?
Il suffit de les regrouper par paquets de 3, ou par paquets de multiples de 3 :



On aurait alors :

$$1 \times 3^2 + 2 \times 3^1 + 2 \times 3^0 = 122$$

soit 122 étoiles **mais** en base 3 !. Pour éviter les confusions possibles dans le changement de base, on écrit $(122)_3$ pour indiquer que la base choisie est 3.

> **Exercice n°1**
> À vous : combien y a t-il d'étoiles en base 2 ?

2 Les bases en informatiques :

En informatique, on recense essentiellement trois bases :

- la base 2, appelée **binaire** pour les machines ;
- la base 10, dite **décimale** pour les humains ;
- la base 16, dite **hexadécimale**, qui est finalement un bon compromis pour les deux protagonistes.

Quelques conversions avant de poursuivre plus en avant :

> **Exercice n°2**
>
> ① Que vaut $(1001)_2$ en base 10 ?
> ② Que vaut 175 en base 2 ?

Il y a autant de chiffres que de valeur de la base pour représenter un nombre dans cette base : les chiffres 0 et 1 pour la base 2, les chiffres $0, 1, ..., 9$ pour la base 10 mais ... pour la base 16 ?
Pas de panique, on complète la liste naturelle des chiffres par les premières lettres de l'alphabet jusqu'à F. Ainsi un nombre **hexadécimal** peut-il s'exprimer à l'aide de lettres...

Le nombre $(A6)_{16}$ signifie : $A \times 16^1 + 6 \times 16^0 = 166$ où le chiffre A vaut le nombre 10 dans le système décimal.

Souvent, on préfixe un nombre par « $0 \times ...$ » pour indiquer qu'il est hexadécimal, c'est-à-dire écrit en base 16.

Exercice n°3

① Convertir $0 \times AF$ en base 10.
② Convertir 0×254 en base 10.

Les conversions suivantes sont assez faciles... mais faut trouver l'astuce !

Exercice n°4

① $(0100\ 0111)_2$ en base 16 ;
② $(CAFE)_{16}$ en base 2 ;

3 Vocabulaire :

Dans la notation binaire,

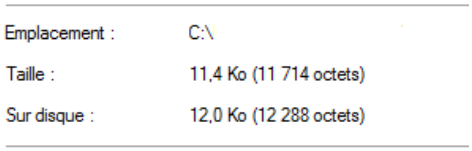
— l'unité élémentaire est le **bit** ;
— un regroupement de 8 bits s'appellent un **octet** ;
— un regroupement de 16 bits (donc deux octets) est un **mot**.

Attention, en anglais un octet s'appelle un « byte ».

Le bit le plus à gauche (respectivement à droite) est le bit de poids **fort** (respectivement **faible**). L'octet est devenu **l'unité** élémentaire des mémoires : on mesure leur capacité en un multiple de l'octet [1]. En effet, il y a de nombreuses confusions dans l'emploi des préfixes traditionnelles pour désigner des quantités d'octets : on a pris l'habitude de désigner par $1ko$ la quantité de $1024 = 2^{10}$ octets alors qu'en général 1 kilo est une préfixe de base 10 pour désigner 1000.
En 1998 l'International Electrotechnical Commission (IEC) a décidé qu'il fallait clarifier la situation. Ainsi de nouvelles unités ont été créées :

1. le **kio** prononcé kibi-octets vaut 1024 octets.
2. le **Mio** prononcé mébi-octets vaut 1024×1024 octets, soit 1048576 octets.

Remarquez la taille du fichier exprimée en Ko puis en octets.



Exercice n°5

① Avec 32 bits, on peut coder combien d'informations différentes ?

② Un CD de musique porte la mention suivante :$16bits/44,1kHz$ stéréo.

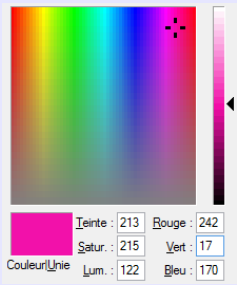
(a) Que signifient ces valeurs ?

(b) Une chanson dure 3 minutes et 26 secondes. Quel est son poids en Mo ?

③ Les couleurs sont codées en général sur 3 octets ; chaque octet représente un niveau de Rouge, de Vert ou de Bleu.

(a) Déterminer combien de couleurs différentes que l'on peut représenter.

(b) Déterminer la représentation décimale de chaque octet de la couleur $F211AA$.



④ Et maintenant comprenez-vous la phrase d'introduction de ce cours ?

1. même s'il a fallu se mettre d'accord !

4 Représentation des entiers positifs

En mathématiques, les nombres entiers comme les nombres réels, sont en nombre infini. En informatique un nombre est représenté par une succession de bits qui ne peut dépasser la capacité limite des mémoires qui les stockent. Comprenons déjà que si l'un veut manipuler des nombres entiers très grands dans un ordinateur, il faudra une capacité de mémoire adaptée.

Exercice n°6

① Avec n bits, combien d'états possibles peut-on coder ?

② Avec n bits, quels entiers positifs pouvons-nous représenter ?

Ainsi avec un octet, soit 8 bits, on peut coder les entiers allant de $0 = (00)_{16} = (0000\,0000)_2$ à $255 = (ff)_{16} = (1111\,1111)_2$.

Exercice n°7

Dans un architecture 32 bits, on peut stocker des nombres comportant 32 bits ; quel est alors le plus grand entier que l'on peut représenter ?

Certains entiers comme 8,16,32,.... s'écrivent rapidement sous forme binaire ou hexadécimale :

Entiers	Représentation binaire sur 8 bits
2	...
8	...
32	...

5 Addition de binaires

Exercice n°8

Que fait $1 + 1$ en binaire ?

L'addition de nombres binaires répond au même algorithme utilisé en base 10 ; il faut savoir prendre en compte la retenue :

	1	1	0	0	1	(25)
+	1	0	0	0	1	(17)
1	0			1		retenues
1	0	1	0	1	0	(42)

Exercice n°9

Poser les opérations suivantes : $(111)_2 + (010)_2$ et $(0101\,0001)_2 + (0000\,1011)_2$

On appelle **complément à 1** d'un nombre binaire, le nombre binaire obtenue en changeant les 0 en 1 et les 1 en 0.

Par exemple,

(1001) devient (0110)

On considère l'algorithme suivant où N est un nombre binaire :

$N \leftarrow$ nombre_binaire
$N_1 \leftarrow$ complement_a_1(N)
$N_2 = N_1 + 1$
$Add = N + N_2$

Exercice n°10

① Tester cet algorithme avec le nombre binaire (1011) puis avec (0110).

② Considérons une mémoire qui ne pourrait stocker que des entiers sur 4 bits. Que vaudrait la variable *add* pour tout N ?

6 Représentation des entiers négatifs

La question est intéressante : comment représenter les nombres négatifs ?
Le premier choix, naïf, serait de réserver le bit de poids fort à représenter le signe (1 pour - et 0 pour + par exemple) puis de représenter avec le reste des bits la valeur absolue du nombre.
Exemple : Supposons une représentation sur 4 bits. Alors pour coder -5 on pourrait écrire :

$$(1101) = \overbrace{1}^{\text{bit de signe}} \quad \underbrace{101}_{\text{représentation de 5}}$$

Mais gros problème la somme de 5 et de -5 ne fait pas 0, comme le montre l'addition suivante :

$$\begin{array}{cccccc} & 0 & 1 & 0 & 1 & (5) \\ + & 1 & 1 & 0 & 1 & (-5) \\ 1 & 1 & & 1 & & \text{retenues} \\ \hline (1) & 0 & 0 & 1 & 0 & (2) \end{array}$$

Pour préserver le compatibilité de l'addition, on préfère une représentation des entiers négatifs par un codage à **compléments à 2** ; la partie précédente montre comment procéder.
En revanche on garde le bit de poids fort pour représenter le signe.Une conséquence immédiate :

Dans une architecture de n bits, il reste $n-1$ bits pour représenter l'information, soit 2^{n-1} négatifs et 2^{n-1} positifs.

Exemple : Imaginons une représentation sur 4 bits. Rappelons que le bit de poids fort indique le signe et le reste la valeur du nombre selon le tableau suivant :

0000	0	1000	-8
0001	1	1001	-7
0010	2	1010	-6
0011	3	1011	-5
0100	4	1100	-4
0101	5	1101	-3
0110	6	1110	-2
0111	7	1111	-1

De façon plus générale, sur une architecture de n bits :

— Le plus grand entier positif représentable est $2^{n-1}-1$ car le premier représenté est 0 ;
— le plus petit négatif est -2^{n-1}

Exercice n°11

Dans une représentation sur $n = 8$ bits,

① Quel est le plus grand entier relatif représentable ? Donner sa valeur décimale et binaire.

② Quel est le plus petit entier négatif représentable ?

③ Quelle est la représentation binaire de 97 ?

④ Quelle est la représentation binaire de -97 ?

⑤ Que vaut le nombre 1001 1100 ?et le nombre 0101 0101 ?

C'est là toute la difficulté : bien distinguer un nombre de sa représentation !