# CHAP 01 Arithmétique - Numération

### ▸ Compétences

- Comprendre le fonctionnement d'une base de numération
- Effectuer des conversions d'une base à une autre
- Savoir effectuer des opérations simples en binaire ou en hexadécimal
- Comprendre l'intérêt et l'utilisation de la base hexadécimale

### ▸ Fil rouge *coloration drone*

Les données de vol d'un drone peuvent être encodées en binaire ou hexadécimal dans les logs.
Il peut donc être intéressant que savoir convertir des valeurs de capteurs en binaire ou de savoir lire une trame hexadécimale simplifiée.

## I. Introduction

### ▸ Intro

Les nombres entiers sont de deux sortes :

- les nombres **cardinaux** qui permettent de dénombrer des objets et de faire des calculs,
- les nombres **ordinaux** qui permettent d'établir un ordre entre des objets d'un ensemble ordonné.

On s'intéresse, en arithmétique, à la fonction cardinale des nombres.

### ▸ Point histoire *(Numération dans l'histoire)*

Pour compter des objets, l'homme a utilisé des cailloux (*calculus* en latin, qui a donné **calcul** ; cf. calculs rénaux), mais aussi différentes parties de son corps, en particulier les doigts. Ainsi a été créée la numération *décimale* (méthode de dénombrement par dizaines).

Certaines civilisations du passé (les Celtes, les Mayas, les Aztèques) ont compté par *vingtaines*. Ainsi, le calendrier maya comportait des mois de 20 jours et envisageait des cycles de 20 ans. Chaque puissance de 20 portait un nom particulier (comme dans la numération décimale : dix, cent, mille...).
Il nous reste des Celtes des nombres comme quatre-vingts, quatre-vingt-dix, l'hôpital des quinze-vingts...
La numération décimale est dite de *base 10* ; la numération *vigésimale* a pour base 20.
Le système *sexagésimal*, c'est-à-dire de base 60, a été utilisé chez les Sumériens puis chez les Babyloniens et chez les Grecs par les astronomes. Il nous en reste les unités de mesure du temps et de mesure d'angles.
Lorsque nous dénombrons des œufs, nous comptons en base 12.

Pour représenter les nombres, nous utilisons des *symboles* : les **chiffres**. Les chiffres que nous utilisons sont les chiffres arabo-indiens, adoptés à partir de 1202, date de la publication du Liber Abaci de Leonardo Fibonacci.
On connaît aussi les chiffres *romains* : I, II, III, IIII, V, VI, VII, VIII, VIIII, X, XI, ..., XV, ..., XX, ..., L, ..., C, ..., D, M.
Chez les Romains, le principe de numération est dit *additif* : pour connaître la valeur d'un nombre, on ajoute les valeurs des symboles (chiffres) qui le composent LXVIII signifie 50 + 10 + 5 + 3 = 68.

Nous utilisons aujourd'hui une *numération de position* dans laquelle c'est la position du chiffre dans le nombre qui donne sa valeur. Ainsi, dans 124, 51, 10 078 ou 713, le chiffre "1" n'a pas la même "valeur". Ce principe de numération permet d'effectuer les opérations usuelles grâce à des *algorithmes* simples.

# II. Bases et numération de position

## II.1. Bases

▸ **Définition 2.1**

La base d'un système de numération est le nombre de chiffres utilisés pour représenter les nombres.
En numération de position, la place d'un chiffre dans le nombre indique sa valeur.

▸ **Définition 2.2**

Dans une base $b$, on utilise $b$ chiffres (de 0 à $b-1$) pour écrire les nombres.
La position de chaque chiffre dans le nombre correspond à une puissance de la base $b$.
Un nombre quelconque $n$ s'écrit alors, dans la base $b$ :

$$n = (a_p a_{p-1} a_{p-2} \dots a_1 a_0)_b = a_p \times b^p + a_{p-1} \times b^{p-1} + a_{p-2} \times b^{p-2} + \dots + a_1 \times b^1 + a_0 \times b^0$$

où les $a_i$ sont les $p+1$ chiffres (non nécessairement tous distincts) qui composent le nombre $n$.

▸ **Remarque 2.1**

Cette écriture, appelée *écriture polynomiale*, permet de convertir un nombre de la base $b$ vers la base 10.

## II.2. Conversion d'une base vers la base 10

▸ **Méthode 2.1**

Pour convertir un nombre de la base $b$ vers la base 10 :

— multiplier chacun des chiffres du nombre par la puissance de $b$ correspondant à sa position dans le nombre ;

— faire la somme des produits obtenus.

# III. La base 2, la base 16

## III.1. Conversions 2 – 10

▸ **Propriété 3.1**

En base 2, on utilise uniquement les deux chiffres 0 et 1.
La position d'un chiffre dans le nombre binaire correspond à une puissance de 2.

Pour effectuer une conversion du binaire vers la base 10, on utilise la méthode donnée dans le paragraphe précédent à partir de l'écriture polynomiale.

▸ **Remarque 3.1**

On peut également « apprendre » les valeurs des premières puissances de 2 pour aller plus vite !

▸ **Python 3.1**

En python, on peut travailler de la base 2 à la base 10, avec la commande `int` par exemple :

```
Début de la console Python
>>> int(0b10), int(0b1010), int(0b111111), int(0b11100001)
(2, 10, 63, 225)
Fin de la console Python
```

Le fait que le nombre soit en *binaire* est signifié, en python, par `0b…`.

▸ **Méthode 3.1** *(Avec le tableau des puissances de 2)*

Écrire un nombre en binaire revient à l'exprimer comme une somme de puissances de 2.

▸ **Remarque 3.2**

Cette méthode peut cependant se révéler longue et fastidieuse lorsque le nombre à convertir est très grand. La méthode suivante, ou méthode des *divisions successives*, est alors plus efficace.

### ▸ Méthode 3.2 *(Les divisions successives par 2)*

Pour convertir un nombre décimal en base 2 on effectue les divisions euclidiennes successives, et on « remonte les restes ».

### ▸ Remarque 3.3

Cette méthode est valable en remplaçant la base 2 par toute autre base $b$ !

### ▸ Python 3.2

En ALGO python, on peut travailler de la base 10 à la base 2, avec la commande `bin` par exemple :

```
🐍 Début de la console Python 🐍
>>> bin(2), bin(10), bin(63), bin(225)
('0b10', '0b1010', '0b111111', '0b11100001')
🐍 Fin de la console Python 🐍
```

## III.2. Conversions 16 – 10

### ▸ Méthode 3.3

La méthode est la même que pour passer du binaire à la base 10 : on écrit le nombre sous forme polynomiale, avec les puissances de 16.

### ▸ Python 3.3

En ALGO python, on peut travailler de la base 16 à la base 10, avec la commande `int` par exemple :

```
🐍 Début de la console Python 🐍
>>> int(0x10), int(0x47), int(0xCA), int(0xAFF)
(16, 71, 202, 2815)
🐍 Fin de la console Python 🐍
```

Le fait que le résultat soit *hexa* est signifié, en ALGO python, par `0x…`.

### ▸ Méthode 3.4

Le plus simple est d'effectuer les divisions successives par 16 jusqu'à obtenir un quotient égal à 0.

### ▸ Python 3.4

En ALGO python, on peut travailler de la base 10 à la base 16, avec la commande `hex` par exemple :

```
🐍 Début de la console Python 🐍
>>> hex(10), hex(16), hex(64), hex(255)
('0xa', '0x10', '0x40', '0xff')
🐍 Fin de la console Python 🐍
```

### III.3. Conversions directes 2 – 16

▸ **Méthode 3.5**

On procède en regroupant les chiffres par paquets de quatre, en commençant par la droite et en étendant, si nécessaire, à gauche par des zéros. On utilise pour cela le tableau donné ci-dessous :

| Base 10 | Base 2 | Base 16 |
|---|---|---|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |

| Base 10 | Base 2 | Base 16 |
|---|---|---|
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |

▸ **Méthode 3.6**

On remplace chaque chiffre de la base 16 en binaire à 4 chiffres (en complétant éventuellement à gauche par des zéros).

## IV. Opérations

### IV.1. Calculs en binaire

▸ **Propriété 4.1** *(Addition)*

Le principe est le même qu'en base 10. Il est utile, pour commencer, de connaître les additions suivantes :

- $0+0=0$ ; $1+0=1$ ; $1+1=10$
- $10+1=11$ ; $11+1=100$
- $100+1=101$ ; $101+1=110$ ; $110+1=111$

On pose l'opération puis on calcule en colonne comme en base 10, sans oublier les éventuelles retenues !

▸ **Propriété 4.2** *(Soustraction)*

En base 2 comme en base 10, les nombres négatifs sont précédés du signe –.
Pour soustraire en binaire, on peut utiliser la méthode du « complément à 2 » :

- on inverse les bits de l'écriture binaire de sa valeur absolue (sans le –) ;
- on ajoute 1 au résultat (les dépassements sont ignorés).

▸ **Exemple 4.1**

$5-3=2$ en décimal ! et $101-11=101-011=101+(100+1)=101+(101)=1010=10$ !

▸ **Méthode 4.1** *(Multiplication)*

Les seules multiplications à effectuer sont des multiplications par 0 ou par 1 ; celles-ci se calculent comme en base 10 : $0\times0=0$ ; $0\times1=0$ ; $1\times0=0$ et $1\times1=1$. L'addition finale s'effectue comme indiqué précédemment.

## IV.2. Calculs en hexa

▸ **Propriété 4.3** *(Addition)*

Le principe est le même qu'en base 10 (mais un peu plus *long* qu'en binaire). Il est utile, pour commencer, de savoir retrouver les additions classiques :

- $0+0=0$ ; $1+0=1$ ; $1+1=2$ ; etc
- $A+1=B$ ; $A+2=C$ ; ... ; $A+6=10$ (car $10+6=16$!) ; etc
- $A+A=14$ (car $10+10=20+1\times16+4$!)
- etc

On pose l'opération puis on calcule en colonne comme en base 10, sans oublier les éventuelles retenues et les symboles en hexa !

▸ **Remarques 4.1**

Les soustractions et multiplications sont également possibles, mais elles ne seront pas étudiées de manière formelle.

## IV.3. Tables de Pythagore

▸ **Remarque 4.2**

Il est tout à fait possible d'utiliser des tables de Pythagore, qui sont les tables d'additions et de multiplications dans une base donnée.

▸ **Illustrations 4.1**

| + | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 10 |

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

| × | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 |
| 2 | 0 | 2 | 4 | 6 | 8 | A | C | E | 10 | 12 | 14 | 16 | 18 | 1A | 1C | 1E | 20 |
| 3 | 0 | 3 | 6 | 9 | C | F | 12 | 15 | 18 | 1B | 1E | 21 | 24 | 27 | 2A | 2D | 30 |
| 4 | 0 | 4 | 8 | C | 10 | 14 | 18 | 1C | 20 | 24 | 28 | 2C | 30 | 34 | 38 | 3C | 40 |
| 5 | 0 | 5 | A | F | 14 | 19 | 1E | 23 | 28 | 2D | 32 | 37 | 3C | 41 | 46 | 4B | 50 |
| 6 | 0 | 6 | C | 12 | 18 | 1E | 24 | 2A | 30 | 36 | 3C | 42 | 48 | 4E | 54 | 5A | 60 |
| 7 | 0 | 7 | E | 15 | 1C | 23 | 2A | 31 | 38 | 3F | 46 | 4D | 54 | 5B | 62 | 69 | 70 |
| 8 | 0 | 8 | 10 | 18 | 20 | 28 | 30 | 38 | 40 | 48 | 50 | 58 | 60 | 68 | 70 | 78 | 80 |
| 9 | 0 | 9 | 12 | 1B | 24 | 2D | 36 | 3F | 48 | 51 | 5A | 63 | 6C | 75 | 7E | 87 | 90 |
| A | 0 | A | 14 | 1E | 28 | 32 | 3C | 46 | 50 | 5A | 64 | 6E | 78 | 82 | 8C | 96 | A0 |
| B | 0 | B | 16 | 21 | 2C | 37 | 42 | 4D | 58 | 63 | 6E | 79 | 84 | 8F | 9A | A5 | B0 |
| C | 0 | C | 18 | 24 | 30 | 3C | 48 | 54 | 60 | 6C | 78 | 84 | 90 | 9C | A8 | B4 | C0 |
| D | 0 | D | 1A | 27 | 34 | 41 | 4E | 5B | 68 | 75 | 82 | 8F | 9C | A9 | B6 | C3 | D0 |
| E | 0 | E | 1C | 2A | 38 | 46 | 54 | 62 | 70 | 7E | 8C | 9A | A8 | B6 | C4 | D2 | E0 |
| F | 0 | F | 1E | 2D | 3C | 4B | 5A | 69 | 78 | 87 | 96 | A5 | B4 | C3 | D2 | E1 | F0 |
| 10 | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | A0 | B0 | C0 | D0 | E0 | F0 | 100 |