1 Représenter de l'information de façon binaire

L'informatique consiste à **traiter** de **façon automatique** de l'**information**.
Comme nous l'avons vu (chap. P1-4), les ordinateurs reposent sur des transistors qui peuvent être assimilés à des interrupteurs commandés électriquement.

> En conclusion, **toute information doit être représentée de façon binaire** en s'appuyant seulement sur 2 états (ou 2 valeurs) : 0 ou 1 !

Une succession de 0 et de 1 sera donc la représentation d'une information quelconque ; laquelle ? cela dépend du sens que l'on décide de donner à ces 0 et 1. Cela pourra être un nombre, un caractère, une image, un son...
Dans ce chapitre, nous allons voir comment représenter les **nombres entiers** de façon binaire (en base 2). Nous verrons aussi comment changer de base (2, 10 ou 16).

2 Découvrir les bases 2, 10 et 16

2.1 Compter en base décimale - base 10

La base décimale est la base qui nous est habituelle et quotidienne. Prenons le temps de réfléchir ce que signifie **compter** !
En décimal, il existe 10 chiffres (10 symboles) pour compter : `0 1 2 3 4 5 6 7 8 9` .
En comptant, après `9`, il vient 1 dizaine (10^1) et 0 unité (10^0), qui s'écrit `10`.
Après `99`, qui représente 9 dizaines et 9 unités, il vient 1 centaine (10^2) et 0 dizaine (10^1) et 0 unité (10^0), qui s'écrit `100`.

Exemple : Le nombre qui s'écrit `3256` s'interprète ainsi :
3 milliers (10^3), 2 centaines (10^2), 5 dizaines (10^1) et 6 unités (10^0)
$3256 = 3 \times 10^3 + 2 \times 10^2 + 5 \times 10^1 + 6 \times 10^0$

2.2 Compter en base binaire - base 2

> En binaire, il n'existe que 2 chiffres : `0` et `1`.

En comptant, après `1`, il vient donc `10` ! c'est à dire 1 "*deuzaine*" (2^1), et 0 unité (2^0).
En poursuivant avec la même logique, après `11` (1 "*deuzaine*" et 1 unité), il vient `100` : 1 "*quatraine*" (2^2), 0 "*deuzaine*" (2^1) et 0 unité (2^0).

Le tableau suivant compte en binaire jusqu'à l'équivalent de 15 en décimal. (on écrira 15_{10}).

décimal	binaire
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111

décimal	binaire
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

On peut par exemple écrire les conversions suivantes :

$$(101)_2 = (5)_{10} \qquad (1000)_2 = (8)_{10} \qquad (1111)_2 = (15)_{10}$$

Le nombre qui s'écrit `10010110` s'interprète ainsi :
$(10010110)_2 = (1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0)_{10}$
$(10010110)_2 = (128 + 16 + 4 + 2)_{10} = (150)_{10}$

Il sera utile d'apprendre par cœur les 8 premières puissance de 2 :

puissance de 2	décimal	binaire
0	$2^0 = 1$	0000 0001
1	$2^1 = 2$	0000 0010
2	$2^2 = 4$	0000 0100
3	$2^3 = 8$	0000 1000
4	$2^4 = 16$	0001 0000
5	$2^5 = 32$	0010 0000
6	$2^6 = 64$	0100 0000
7	$2^7 = 128$	1000 0000
8	$2^8 = 256$	1 0000 0000

Vocabulaire important :
- un chiffre binaire s'appelle un bit (contraction anglaise de ***bi****nary digi****t***).
- un nombre binaire de 8 bits s'appelle un octet (*byte*, en Anglais).

2.3 Compter en base hexadécimale - base 16

La **base hexadécimale** est également très fréquente en numérique car elle permet d'écrire rapidement (en moins de symboles) un équivalent de nombres binaires.

En hexadécimal, il existe 16 chiffres : `0 1 2 3 4 5 6 7 8 9 A B C D E F` .

En comptant, après `9`, il vient simplement `A` puis `B`, puis `C`, puis `D`, puis `E`, puis `F` et enfin `10` : c'est à dire 1 "*seizaine*" (16^1), et 0 unité (16^0).

Le nombre qui s'écrit `2AD3` s'interprète ainsi :
$(2AD3)_{16} = (2 \times 16^3 + 10 \times 16^2 + 13 \times 16^1 + 3 \times 16^0)_{10}$
$(2AD3)_{16} = (2 \times 4096 + 10 \times 256 + 13 \times 16 + 3 \times 1)_{10} = (10963)_{10}$

L'ensemble des 16 chiffres hexadécimaux peuvent se coder avec 4 bits, et donc un octet s'écrit simplement avec 2 chiffres hexadécimaux, d'où le gain de place lorsqu'on convertit des nombres binaires en hexadécimal.

Exemples :

décimal	binaire	hexadécimal
0	0000 0000	0
1	0000 0001	1
2	0000 0010	2
3	0000 0011	3
4	0000 0100	4
5	0000 0101	5
6	0000 0110	6
7	0000 0111	7
8	0000 1000	8
9	0000 1001	9

décimal	binaire	hexadécimal
10	0000 1010	A
11	0000 1011	B
12	0000 1100	C
13	0000 1101	D
14	0000 1110	E
15	0000 1111	F
16	0001 0000	10
104	0110 1000	68
162	1010 0010	A2
255	1111 1111	FF

2.4 Écriture d'un nombre dans une base b quelconque

Une base b contient b caractères différents.
Un nombre écrit $a_n a_{n-1} \cdots a_1 a_0$ s'interprète ainsi :
$a_n a_{n-1} \cdots a_0 = a_n \times b^n + a_{n-1} \times b^{n-1} + \cdots + a_1 \times b^1 + a_0 \times b^0$

3 Passage d'une base dans une autre

3.1 Obtenir l'écriture binaire d'un entier

On souhaite traduire un entier N dans son écriture binaire (ou plutôt une représentation de celle-ci).

Méthode :
- Diviser le nombre par 2 et noter le reste de la division (0 ou 1).
- Recommencer avec le quotient de la division précédente.
- Poursuivre ainsi jusqu'à ce que le quotient soit nul.

Le nombre binaire est alors la suite des restes (0 ou 1) obtenus en partant de la dernière division vers la première.

Algorithme : Écriture binaire de l'entier N :
(représentation sous forme de liste de bits)

```
1 quotient ← N
2 liste ← [ ]
3 tant que quotient ≠ 0 faire
4 |   reste ← quotient % 2            /* reste de la div. euclidienne */
5 |   liste ← [reste] + liste
6 |   quotient ← quotient // 2     /* quotient de la div. euclidienne */
7 fin tq
8 renvoyer liste
```

Exemple détaillé : passage de $(652)_{10}$ en binaire : $(652)_{10} = (1010001100)_2$
$652 = 0 + 326 \times 2$
$326 = 0 + 163 \times 2$
$163 = 1 + 81 \times 2$
$81 = 1 + 40 \times 2$
$40 = 0 + 20 \times 2$
$20 = 0 + 10 \times 2$
$10 = 0 + 5 \times 2$
$5 = 1 + 2 \times 2$
$2 = 0 + 1 \times 2$
$1 = 1 + 0 \times 2$

3.2 De l'écriture binaire vers l'écriture décimale

Il suffit dans ce cas d'ajouter toutes les puissances de 2 qui sont affectées du bit 1 dans le nombre binaire.
Exemple : $(1010001100)_2 = 2^9 + 2^7 + 2^3 + 2^2 = 512 + 128 + 8 + 4 = (652)_{10}$

1	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**
9	8	7	6	5	4	3	2	1	0
2^9	0	2^7	0	0	0	2^3	2^2	0	0
512	0	128	0	0	0	8	4	0	0

3.3 Obtenir l'écriture hexadécimale d'un entier

On retrouve le même principe que pour l'écriture binaire (en adaptant évidemment le 2 en 16 !) :

Méthode :
- Diviser le nombre par 16 et noter le reste de la division (de 0 à F).
- Recommencer avec le quotient de la division précédente.
- Poursuivre ainsi jusqu'à ce que le quotient soit nul.

Le nombre hexadécimal est alors la suite des restes (de 0 à F) obtenus en partant de la dernière division vers la première.

Algorithme : Écriture hexadécimale de l'entier N :
(représentation sous forme de liste de digits)
C'est le même algo que pour la représentation bianire en remplaçant le 2 par 16 !

Exemple détaillé : Conversion de $(652)_{10}$ en hexadécimal : $(652)_{10} = (28C)_{16}$
$652 = C\ (12) + 40 \times 16$
$40 = 8 + 2 \times 16$
$2 = 2 + 0 \times 16$

3.4 De l'écriture hexadécimale vers l'écriture décimale

Il suffit d'ajouter toutes les puissances de 16 multipliées par le chiffre hexadécimal qui leur est affecté dans le nombre hexadécimal.

Exemple : $(28C)_{16} = 512 + 128 + 12 = (652)_{10}$:

2	**8**	**C**
2	1	0
2×16^2	8×16^1	12×16^0
2×256	8×16	12×1
512	128	12

3.5 Écriture des entiers en Python

Python propose les fonctions `bin` et `hex` pour obtenir les écritures binaire ou hexadécimale d'un entier : renvoient une chaîne de caractères préfixée par '`0b`' pour le binaire et '`0x`' pour l'hexadécimal.

```
>>> bin(652)
'0b1010001100'
>>> hex(652)
'0x28c'
>>> 0b1010001100, 0x28c # on peut aussi directement écrire un entier en binaire ou hexa
(652, 652)
```