BINAIRE - HEXADECIMAL

1 - Comment comptons nous en décimal ?

Tout le monde compte en base 10, mais comment fonctionne notre mode comptage réellement ? Comment est construit notre système numérique ? Pour répondre à cela, reprennons depuis le début : comment avez vous appris à compter à l'école ?

Certains diront que notre base 10 est venue de nos 10 doigts, mais ce qui est sûr c'est qu'il en découle deux choses :

- Il existe **10 chiffres** : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9.
- Avec ces chiffres ont peut compter jusqu'à 9. (La plus haute valeur des chiffres.)

Pour aller au delà de 9 il faut **changer de rang**.
Ça veut dire que si le rang des unités est plein, on commence le rang des dizaines et on remet les unités à zéro et ainsi de suite.

Par exemple, arrivé à 19, le rang des unités est plein. On ajoute donc une dizaine et on remet à zéro le rang des unités : on arrive donc à 20.

J'ai parlé de rangs des centaines, de dizaines et d'unités. On voit que une centaine vaut 10 dizaines et que une dizaines vaut 10 unités. Plus mathématiquement, **un rang est égale au précédent multiplié 10**.
On peut dire que chaque rang est à une puissance de 10 supérieur au précédent.

De cette manière, le nombre 56 = 50 + 6 mais que l'on peut aussi écrire $56 = 5\times10^1 + 6\times10^0$.

Ce que je viens de faire, c'est décomposer 56 en puissances de 10 (unités, dizaines, centaines…).

On peut décomposer chaque nombre en puissances de 10 successives. Par exemple, $3506 = 3\times10^3 + 5\times10^2 + 6\times10^0$.

Avec cette explication, vous devez avoir compris qu'en base 10 :

- On change de rang dès que la précédente est à 9.
- On peut décomposer tous les nombres en une somme de puissances de 10.
- Si on décompose un nombre en puissances de 10, c'est parce que 10 est notre base.

Ce dernier point est important car en base 2, il faut décomposer en puissances de deux !

2 - Le binaire

2.1 - Présentation

Le binaire est le mode de comptage non plus en base 10 mais **en base 2**. Il est utilisé par les ordinateurs, car les machines ne peuvent comparer que deux valeurs : des 1 et des 0.

Je vous avais parlé des rangs (unités, dizaines, centaines…), et bien sachez qu'en binaire on emploie le mot « bit » (contraction de « binary-digit », signifiant simplement « rang binaire »).
Par exemple, le nombre en base 2 « 10011 » s'étale sur 5 bit.

Là où cela se complique, c'est qu'en binaire **chaque rang ne peut prendre que deux valeurs** (il pouvait en prendre dix en décimal). Donc, dès que le rang atteint sa deuxième – la plus haute – valeur on change de rang. **En binaire, un rang commence à 0 et se termine à 1.**

Vous pouvez en comprendre que chaque bit représente une puissance de 2, tout comme chaque rang en base 10 est une puissance de 10.
Bon, pour commencer et tenter d'y voir un peu plus clair, on va compter en binaire jusqu'à dix :

valeur en décimal :	équivalent en binaire :	explications :
0	0	logique !
1	1	simple !
2	10	Le premier rang a atteint le maximum autorisé ! Qu'à cela ne tienne, on passe au rang suivant. On met le second à 1 et on remet le premier à 0.
3	11	On re-remplit le rang 1.
4	100	Le rang 1 est plein, mais le 2 aussi ! On passe donc au troisième et on remet les précédents à 0 (comme on le fait lorsque l'on passe de 0999 à 1000, par exemple).
5	101	On procède de même.
6	110	
7	111	
8	1000	On entame le quatrième rang.
9	1001	On recommence au premier...
10	1010	On rempli les rangs.

Il suffit d'appliquer une règle : entamer le rang suivant quand celui en cours est plein!

Bon, pour compter jusqu'à 10 ou même 20, cela va encore de remplir ce tableau, mais si je vous demande de convertir 450 en binaire ? Vous n'allez pas monter un par un, si ?
Dans ce qui suit, on va voir une technique générale.

2.2 - Conversion du décimal en binaire

Pour le moment, on n'a compté jusqu'à dix. Mais on ne sait pas encore **convertir**. Sans plus attendre donc, voici la conversion !

2.2.1 - Méthode 1 : les puissances de 2

Pour y arriver, on doit décomposer notre nombre en puissances de 2. C'est le même principe que la décomposition en puissances de dix, sauf que l'on ne décompose pas en milliers, centaines et dizaines, mais en puissances de deux ; qui sont : 1, 2, 4, 8, 16, 32, 64 ..., 512, 1024, etc (une valeur est égale à la précédente multipliée par 2).

Ainsi, si l'on prend l'exemple du nombre 26, on obtient la décomposition suivante : 26 = 16 + 8 + 2. Il suffit ensuite de remplacer ces nombres par les puissances :

$26 = 16 + 8 + 2$
$26 = 1\times16 + 1\times8 + 1\times2$
$26 = 1\times2^4 + 1\times2^3 + 1\times2^1$ (on écrit les coef sous forme de puissances de 2)
$26 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$ (on ajoute les puissances de 2 qui manquent)
$26 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$ (voyez les puissances de 2 qui sont toutes là)
$26 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$ (en orange : notre nombre en binaire !)

Il est important de ne pas oublier les puissances dont les coefficients sont zéro.

Finalement, pour obtenir le nombre 26 en binaire, il suffit de mettre les coefficients qui sont devant les puissances de 2 à la suite. On obtient : 11010.

On écrit : $(26)_{dec} = (1\ 1010)_{bin}$
Je récapitule la méthode :

1. On a notre nombre en décimal.
2. On le décompose en valeurs de puissances de 2
3. Si certaines puissances manquent, on les rajoutent en mettant 0 devant.
4. On lit les coefficients devant les puissances de 2, ce sera notre nombre en binaire !
5. Par commodité, d'écriture, on regroupe les chiffres par 4.
   (par ex : 101010101 se notera 1 0101 0101). On verra pourquoi plus loin.

2.2.2 - Méthode 2 : les divisions euclidiennes par 2

Tout aussi simple à comprendre. Cette méthode est mieux pour des grands nombres et est plus facile à utiliser en programmation (il est facile d'en faire un *algorithme*). Voilà comment on fait :

- On a notre nombre en décimal.
- On le divise par 2 et on note le reste de la division (c'est soit un 1 soit un 0).
- On refait la même chose avec le quotient précédent, et on met de nouveau le reste de coté.
- On re-itère la division, et ce jusqu'à ce que le quotient est 0.
- Le nombre en binaire apparaît : le premier à placer est le dernier reste non nul. Ensuite, on remonte en plaçant les restes que l'on avait. On les place à droite du premier 1.

Comme rien ne vaut un exemple :

- `Notre nombre est 164`
- `164 ÷ 2 = 82 + 0`
- `82 ÷ 2 = 41 + 0`
- `41 ÷ 2 = 20 + 1`
- `20 ÷ 2 = 10 + 0`
- `10 ÷ 2 = 5 + 0`
- `5 ÷ 2 = 2 + 1`
- `2 ÷ 2 = 1 + 0`
- `1 ÷ 2 = 0 + 1`

On voit apparaître notre nombre binaire en rouge : **il faut le lire de bas en haut**, ce qui donne 1010 0100.
Joli non ?

2.3 - Conversion du binaire en décimal

Dans l'autre sens maintenant : convertir un nombre en base 2 en un nombre en base 10 ! je vous rassure tout de suite, c'est plus simple!

Prenons le nombre (au hasard) : 101 0110. On voit qu'il s'étale sur 7 rangs, et sait que chaque rang correspond à une puissance de 2 : le premier (en partant de la **droite**) est le rang 0, le second est le rang 1, etc.

Pour le convertir en décimal, on procède de la manière suivante : on multiplie par 2^0 la valeur du rang 0, par 2^1 la valeur du rang 1, par 2^2 la valeur du rang 2, [...], par 2^{10} la valeur du rang 10, etc.

Pour notre nombre 101 0110, on a donc $0\times2^0 + 1\times2^1 + 1\times2^2 + 0\times2^3 + 1\times2^4 + 0\times2^5 + 1\times2^6$.
Ensuite, il suffit simplement de remplacer les puissances de 2 par leurs valeurs et de faire la somme : $0\times1 + 1\times2 + 1\times4 + 0\times8 + 1\times16 + 0\times32 + 1\times64 = 86$.

donc : $(101\ 0110)_{bin} = (86)_{dec}$

3 - l'Hexadécimal

3.1 - Présentation

Après le binaire, voici venu une autre base : **le système hexadécimal** qui travaille en base 16.

Si vous avez suivi jusqu'ici, vous devinerez qu'il faudra 16 caractères différents pour représenter chacune des 16 valeurs.

C'est alors qu'avec une originalité déroutante, en hexadécimal, les caractères sont 0, 1, 2 etc. jusqu'à 9 ainsi que A, B, C, D, E et F.

Vous l'aurez compris : A en hexadécimal vaut 10 en décimal, B vaut 11, ... et F vaut 15.

En hexadécimal, le changement de rang se fait donc à F. Ainsi E+1 = F et F+1 = 10 (dire “un-zéro”).

Plus compliqué : F+B = 1A.

Ça va ? Alors passons à la conversion !

3.2 - Conversion du décimal en hexadécimal

La conversion d'un nombre de la base 10 en base 16 est aussi “facile” qu'avec le binaire. Pour le binaire il fallait décomposer en puissances de 2, ici on décompose en puissances de 16.

Ces puissances de 16 sont ? Alors ? Ok, je vous donne les premiers :

- $16^0 = 1$
- $16^1 = 16$
- $16^2 = 256$
- $16^3 = 4096$
- $16^4 = 65536$
- ...

Pour l'exemple, je prendrais le nombre 1680. Il faut donc commencer par le décomposer en puissances de 16 :
$1680 = 6\times256 + 9\times16 + 0\times1$
$1680 = 6\times16^2 + 9\times16^1 + 0\times16^0$.

La conversion en hexadécimal de 1680 est donc 690 (lire “six-neuf-zéro”).

Un autre exemple : convertissons 2009 en hexadécimal : $2009 = 7\times16^2 + 13\times16^1 + 9\times16^0$. Le nombre en base 16 correspondant à 2009 est donc 7D9 (rappelez vous, chaque rang peut monter jusqu'à 15 en base 16, et le D vaut 13).

C'est le même principe qu'avec le binaire, le changement de base se fait juste à 16 au lieu de 2.

3.3 - Conversion de l'hexadécimal en décimal

Dans ce sens, c'est plus simple : prenons un nombre : 4F2C. Il a 4 rangs : chaque rang est une puissance de 16 : pour convertir, on multiplie le premier rang (en partant de la droite) par 16^0, le second par 16^1, etc.

Ainsi on obtient :
$4F2C = 4\times16^3 + F\times16^2 + 2\times16^1 + C\times16^0$
$4F2C = 4\times16^3 + 15\times16^2 + 2\times16^1 + 12\times16^0$
$4F2C = 4\times4096 + 15\times256 + 2\times16 + 12\times1$
4F2C hex = 20 268 dec.

C'est simple non ? Il suffit de prendre les puissances de 16 croissantes.

3.4 - Conversion du binaire en hexadécimal

La conversion entre l'hexadécimal et le binaire est super facile si vous savez manipuler ces bases entre les nombres 0 et 15.

Prenons un nombre en binaire : 101 0011 1011.

Notez que je l'ai séparé en blocs de 4 chiffres (comme on sépare les nombres en bloc de 3. Par exemple, 30000 s'écrit 30 000).

Ceci nous simplifie la tache : en effet, on sait que 4 rangs binaires permettent de monter jusqu'à 15. Et bien, 1 rang en hexadécimal aussi ! (Cela vient du fait que 2^4 (4 rangs en base 2) = 16^1 (1 rang en base 16)).

De cette façon, **4 bits en binaire** seront représentés par **un rang en hexadécimal** !

Ainsi, le premier quadruplet : 1011 deviendra un seul rang en hexadécimal :
1011 = 11 en décimal = B en hexadécimal. Le second quadruplet 0011 devient 3 en hexadécimal ; et finalement le dernier : 101 (ou 0101) devient : 5.

Ainsi, $(101\ 0011\ 1011)_{bin} = (53B)_{hex}$.

3.5 - Conversion de l'hexadécimal en binaire

On va utiliser le même principe que ci-dessus, à savoir qu'un rang en base 16 correspond à 4 rangs en base 2.

On convertira le nombre hexadécimal **BE57**. On prend chaque rang que l'on convertit individuellement en binaire :

- $(B)_{hex} <=> (11)_{dec} <=> (1011)_{bin}$
- $(E)_{hex} <=> (14)_{dec} <=> (1110)_{bin}$
- $(5)_{hex} <=> (5)_{dec} <=> (0101)_{bin}$
- $(7)_{hex} <=> (7)_{dec} <=> (0111)_{bin}$

Prenez bien soin de mettre 0101 au lieu de 101, car il ne faut pas se tromper quand on va mettre les quadruplets bout à bout :
BE57 <=> 1011 1110 0101 0111 .

4 - Généralisation à toutes les bases

C'est tout aussi simple, mais on va utiliser des mathématiques :-).

Notons simplement que si l'hexadécimal utilise les chiffres de 0 à 9 et les lettres de A à F, les base plus grandes utilisent la même chose : ainsi la base 18 utilise les caractères 0123456789ABCDEFGH.

Au delà de 36 (on serait à Z), il faudrait utiliser autre chose, par exemple des lettres grecques (α, β, γ, δ, ε...), russes, etc. Mais on s'en passera.

J'ajoute aussi qu'il est parfaitement inutile d'apprendre à convertir dans toutes les bases.
Seules les bases 10 (décimal), 16 (hexadécimal), 2 (binaire) et 8 (octal) sont utilisées (en informatique surtout).

4.1 - Principe

Le principe est sensiblement la même qu'en base 2 ou 16 : faire des divisions euclidiennes avec les puissances successives de la base.

4.1.1 - Exemple : 160 en base 7

Pour avoir 160 en base 7, il faut trouver les puissance de 7 inférieures à 160. Il y'en a 3 :

- $7^2 = 49$
- $7^1 = 7$
- $7^0 = 1$

Puis on fait les divisions euclidiennes, en commençant par 49 :

```
160 ÷ 49 = 3 et il reste 13.
13 ÷ 7 = 1 et il reste 6
6 ÷ 1 = 6 et il reste 0
```

D'où : 160 vaut 316 en base 7, que l'on note finalement 160 = $(316)_7$.

4.1.2 - Exemple : 600 000 en base 82

Les puissances de la base :

- $82^3 = 551368$
- $82^2 = 6724$
- $82^1 = 82$
- $82^0 = 1$

Et les divisions successives :

```
600 000 ÷ 551 368 = 1 et il reste 48 632.
48 632 ÷ 6724 = 7 et il reste 1564
1564 ÷ 82 = 19 et il reste 6
6 ÷ 1 = 6 et il reste 0
```

D'où : 600 000 en base 82 qui vaut 17J6 (avec J le dix-neuvième symbole, en partant de 0).

Vous pouvez dés lors vérifier : `1 × (82^3) + 7 × (82^2) + 19 × (82^1) + 6 × (82^0) = 600 000`.

Je n'ai pas grand chose à ajouter : c'est aussi simple que cela en fait. Le plus dur, c'est de trouver la puissance la plus haute de la base qu'il faut prendre, ensuite ce ne sont que des divisions euclidiennes successives.

5 - Effectuer des opérations dans d'autres bases

On sait additionner et multiplier des nombres en base 10 depuis l'école primaire. Au fil du temps, c'est devenu naturel, donc encore une fois, revenons aux sources : pour additionner et multiplier en base dix à l'école primaire **on pose l'opération** :

```
   12        12        12        12        12        12        12
 × 23      × 23      × 23      × 23      × 23      × 23      × 23
 -----     -----     -----     -----     -----     -----     -----
   ??         6        36        36        36        36        36
                                  0     +  40     + 240     + 240
                                                            -----
                                                            = 276
```

Souvenir du bon vieux temps ?
Ben, on va faire ça ici, mais dans les autres bases, et vous verez qu'il n'y a rien de compliqué : c'est la même chose.

Avec de l'entraînement, je pense qu'il est possible de multiplier deux nombres en binaire mentalement, mais c'est un peu inutile.

5.1 - En binaire

5.1.1 - Additionner en binaire

Additionner en binaire n'est pas compliqué : c'est le même principe que dans les autres bases. Il suffit de poser l'opération et de faire attention aux retenues. Après, il est aussi possible de convertir en base 10, de faire l'opération de tête, puis de revenir en base 2, mais c'est mieux de savoir faire l'opération directement en binaire.

Il n'y a rien de mieux qu'un exemple pour comprendre, donc en voici un : 2+2, soit $(10)_2+(10)_2$

```
  10      10      10
+ 10     +110     + 10
----     ----     ----
   0       00      100
```

Sur l'opération du milieu, 1+1 en binaire donne 10, donc 0 et une retenue.

Voici un autre exemple, avec des nombres un peu plus grands. La difficulté n'est pas plus grande, mais il faut parfois faire attention aux retenues qui se propagent : 22+2, donc 10110+10.

```
                1        11        11        11
 10110    10110    10110    10110    10110
+   10   +   10   +   10   +   10   +   10
------   ------   ------   ------   ------
     0       00      000     1000    11000
```

Et on a bien 11000 qui vaut 24, ce qui est bien 22+2.

5.1.2 - Soustraire en binaire

Là encore, il faut raisonner comme on raisonne à la petite école : en posant l'opération. Ça se fait tout seul, il suffit de bien faire attention aux retenues.
Mais je vais vous apprendre une autre méthode, qui transforme les soustractions en additions, et simplifie alors toute cette histoire de retenues.

En fait, au lieu de faire A – B, on fera A + ($\overline{B}$ + 1). Ici, $\overline{B}$ (prononcer « B barre ») est le complément à 1 de B.

5.1.2.1 - Complément à 1

Le complément à 1 est un nombre qui existe dans toutes les bases, mais en binaire il est très facile à trouver : il suffit de changer les 1 en 0 et les 0 en 1. C'est tout :

Nombre (B)	Complément ($\overline{B}$)
0	1
1	0
1100	0011

C'est assez simple, non ?

5.1.2.2 - Soustraire

Maintenant qu'on a le complément à 1 d'un nombre, il est possible de faire des soustractions.
Souvenez-vous de ce qu'il faut faire : au lieu A – B, on fera A + ($\overline{B}$ + 1).

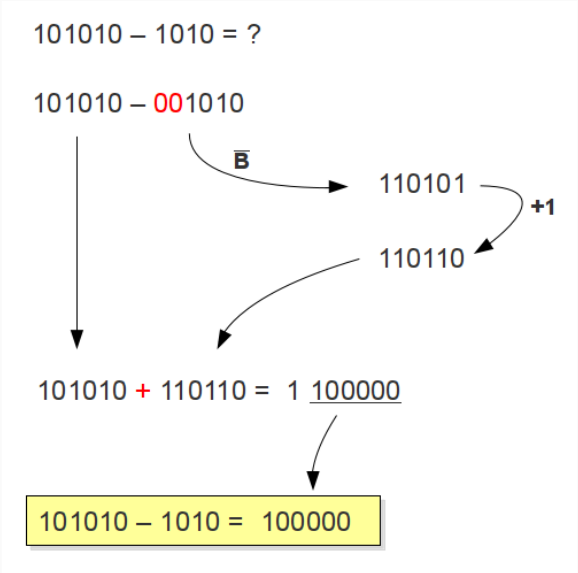
Exemple : calculons 101010 – 1010.

Premièrement, il faut commencer à donner le même nombre de rangs à chaque terme : le premier nombre s'écrit sur 6 bit et le second seulement sur 4. Il faut donc écrire le second sur 6 bit aussi : 1010 devient 001010. C'est de ce nombre qu'il faudra inverser les bits.

- On écrit le plus petit nombre avec autant de bits que le grand : 1010 devient 001010.
- En inversant tous les bit de 001010 on obtient 110101.
- On ajoute 1, ce qui fait 110110.
- On fait maintenant 101010 + 110110. Cela donne : 1100000.
- En remarquant que la différence de deux nombre positifs ne peut pas être supérieure au plus grand des deux nombres, il est facile de conclure que le résultat doit être plus petit que 101010. Pour cela, on supprime le premier bit.
  Donc 1100000 devient 100000.
- 100000 est le résultat de notre soustraction.

C'est barbare, mais ça marche (c'est démontrable mathématiquement, merci PJ pour l'info !) :
$(101010)_2 - (1010)_2 = (100000)_2$
$(42)_{10} - (10)_{10} = (32)_{10}$

Je pense que c'est plus simple avec ce diagramme :



5.1.3 - Multiplier deux nombres en binaire

Là encore, je vais vous dire que c'est très simple.
C'est très simple.

Comme je l'ai suggéré en intro, on va poser l'opération comme quand on apprend à multiplier des nombres à l'école primaire.

Mais faisons-le en binaire : c'est exactement la même chose, chaque nombre de la ligne du bas sera distribué à la ligne du haut. Ensuite, chaque ligne sera sommée (ne pas oublier les retenues à ce moment là) et le résultat sera alors obtenu.

On va commencer avec $(1010)_{bin}$ fois $(101010)_{bin}$:

```
        1010
×     101010
----------
        0000
+      10100
+     000000
+    1010000
+   00000000
+  1010000000
----------
=110100100
```

On peut vérifier, si vous voulez :

En base 2 : 1010 × 10 1010 = 1 1010 0100.
Or, $(1010)_{bin} = (10)_{déc}$; $(101010)_{bin} = (42)_{déc}$; donc théoriquement, on devrait avoir 420.
Vous pouvez vérifier : $(1\ 1010\ 0100)_{bin}$ vaut bien 420 en base dix.

Voilà, c'est tout :-). C'est aussi simple que ça ! Il suffit de poser la multiplication.
N'oubliez pas les retenues (1+1 : ça fait 0, je retiens 1, donc : $(1+1 = 10)_{bin}$), et n'oubliez pas les « zéros de droite ».

6 - Conclusion

Voilà : vous avez la méthode pour convertir des nombres entiers entre les bases 2, 10 et 16 et même toutes les bases.

Juste un petit code de programmation qui convertit le décimal en binaire. Ci dessous, j'utilise la méthode des divisions successives.

En Python

```python
# -*- coding:Utf-8 -*-

a = int(input("nombre à convertir : "))
bin = []

# convertisseur
while (a != 0):
    bin = bin + [a % 2]
    a /= 2

# on inverse la liste
bin.reverse()
for i in bin:
    print i,
```

Un convertisseur en ligne: https://lehollandaisvolant.net/tout/tools/bases/