

Binaire I

Les ordinateurs transforment toutes les données en nombres et manipulent uniquement ces nombres. Ces nombres sont stockés sous la forme de listes de 0 et de 1. C'est l'écriture binaire des nombres ! Pour mieux comprendre l'écriture binaire, tu vas d'abord mieux comprendre l'écriture décimale.

Vidéo ■ Binaire - partie 1 - écriture décimale

Vidéo ■ Binaire - partie 2 - de l'écriture binaire vers l'entier

Vidéo ■ Binaire - partie 3 - de l'entier vers l'écriture binaire

Cours 1 (Écriture décimale).

L'écriture habituelle des entiers se fait dans le système décimal (en base 10). Par exemple, 70 685 c'est $7 \times 10\,000 + 0 \times 1\,000 + 6 \times 100 + 8 \times 10 + 5 \times 1$:

7	0	6	8	5
10000	1000	100	10	1
10^4	10^3	10^2	10^1	10^0

(on voit bien que 5 est le chiffre des unités, 8 celui des dizaines, 6 celui des centaines...).

Il faut bien comprendre les puissances de 10. On note 10^k pour $10 \times 10 \times \dots \times 10$ (avec k facteurs).

d_{p-1}	d_{p-2}	\dots	d_i	\dots	d_2	d_1	d_0
10^{p-1}	10^{p-2}	\dots	10^i	\dots	10^2	10^1	10^0

On calcule donc l'entier correspondant aux chiffres $[d_{p-1}, d_{p-2}, \dots, d_2, d_1, d_0]$ par la formule :

$$n = d_{p-1} \times 10^{p-1} + d_{p-2} \times 10^{p-2} + \dots + d_i \times 10^i + \dots + d_2 \times 10^2 + d_1 \times 10^1 + d_0 \times 10^0$$

Activité 1 (De l'écriture décimale vers l'entier).

Objectifs : à partir de l'écriture décimale, retrouver l'entier.

Écris une fonction `decimale_vers_entier(liste_decimale)` qui à partir d'une liste représentant l'écriture décimale calcule l'entier correspondant.

`decimale_vers_entier()`

Usage : `decimale_vers_entier(liste_decimale)`

Entrée : une liste de chiffres entre 0 et 9

Sortie : l'entier dont l'écriture décimale est la liste

Exemple : si l'entrée est `[1, 2, 3, 4]`, la sortie est 1234.

Indications. Il faut faire la somme d'éléments du type

$$d_i \times 10^i \quad \text{pour } 0 \leq i < p$$

où p est la longueur de la liste et d_i est le chiffre en position i en comptant à partir de la fin (c'est-à-dire de droite à gauche). Pour gérer le fait que l'indice i utilisé pour parcourir la liste ne correspond pas à la puissance de 10, il y a deux solutions :

- comprendre que $d_i = \text{liste}[p - 1 - i]$ où `liste` est la liste des p chiffres,
- ou bien commencer par inverser `liste`.

Cours 2 (Binaire).

- **Puissances de 2.** On note 2^k pour $2 \times 2 \times \dots \times 2$ (avec k facteurs). Par exemple, $2^3 = 2 \times 2 \times 2 = 8$.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

- **Écriture binaire : un exemple.**

Tout entier admet une écriture binaire, c'est-à-dire une écriture où les seuls chiffres sont des 0 ou des 1. En binaire, les chiffres sont appelés des **bits**. Par exemple, 1.0.1.1.0.0.1 (prononce les chiffres un par un) est l'écriture binaire de l'entier 89. Comment faire ce calcul ? C'est comme pour la base 10, mais en utilisant les puissances de 2 !

1	0	1	1	0	0	1
64	32	16	8	4	2	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0

Donc l'écriture 1.0.1.1.0.0.1 représente l'entier :

$$1 \times 64 + 0 \times 32 + 1 \times 16 + 1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 = 64 + 16 + 8 + 1 = 89.$$

- **Écriture binaire : formule.**

b_{p-1}	b_{p-2}	\dots	b_i	\dots	b_2	b_1	b_0
2^{p-1}	2^{p-2}	\dots	2^i	\dots	2^2	2^1	2^0

On calcule donc l'entier correspondant aux *bits* $[b_{p-1}, b_{p-2}, \dots, b_2, b_1, b_0]$ comme une somme de termes $b_i \times 2^i$, par la formule :

$$n = b_{p-1} \times 2^{p-1} + b_{p-2} \times 2^{p-2} + \dots + b_i \times 2^i + \dots + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0$$

- **Python et le binaire.** Python accepte que l'on écrive directement les entiers en écriture binaire, il suffit d'utiliser le préfixe « 0b ». Exemples :
 - avec `x = 0b11010`, alors `print(x)` affiche 26,
 - avec `y = 0b11111`, alors `print(y)` affiche 31,
 - et `print(x+y)` affiche 57.

Activité 2 (De l'écriture binaire vers l'entier).

Objectifs : à partir de l'écriture binaire, retrouver l'entier (en écriture décimale usuelle).

1. Calcule les entiers dont l'écriture binaire est donnée ci-dessous. Tu peux le faire à la main ou t'aider de Python. Par exemple 1.0.0.1.1 vaut $2^4 + 2^1 + 2^0 = 19$ ce que confirme la commande `0b10011` qui renvoie 19.

- 1.1, 1.0.1, 1.0.0.1, 1.1.1.1
- 1.0.0.0.0, 1.0.1.0.1, 1.1.1.1.1
- 1.0.1.1.0.0, 1.0.0.0.1.1
- 1.1.1.0.0.1.0.1

2. Écris une fonction `binaire_vers_entier(liste_binaire)` qui à partir d'une liste représentant l'écriture binaire calcule l'entier correspondant.

binaire_vers_entier()

Usage : `binaire_vers_entier(liste_binaire)`

Entrée : une liste de *bits* 0 et 1

Sortie : l'entier dont l'écriture binaire est la liste

Exemples :

- entrée `[1, 1, 0]`, sortie 6
- entrée `[1, 1, 0, 1, 1, 1]`, sortie 55
- entrée `[1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1]`, sortie 3383

Indications. Il faut cette fois faire la somme d'éléments du type

$$b_i \times 2^i \quad \text{pour } 0 \leq i < p$$

où p est la longueur de la liste et b_i est le *bit* (0 ou 1) en position i de la liste en comptant à partir de la fin.

3. Voici un algorithme qui effectue le même travail : il permet le calcul de l'entier à partir de l'écriture binaire, mais il a l'avantage de ne jamais calculer directement des puissances de 2. Programme cet algorithme en une fonction `binaire_vers_entier_bis()` qui a les mêmes caractéristiques que la fonction précédente.

Algorithme.

Entrée : `liste` : une liste de 0 et de 1

Sortie : le nombre binaire associé

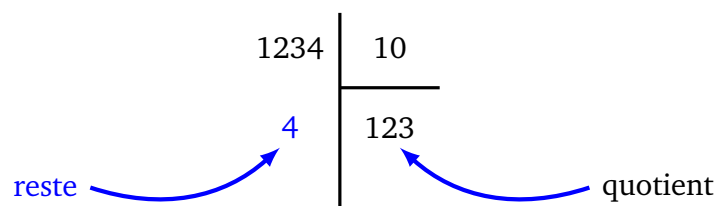
- Initialiser une variable n à 0.
- Pour chaque élément b de `liste` :
 - si b vaut 0, alors faire $n \leftarrow 2n$,
 - si b vaut 1, alors faire $n \leftarrow 2n + 1$.
- Le résultat est la valeur de n .

Cours 3 (Écriture décimale (bis)).

Bien sûr, quand tu vois le nombre 1234 tu sais tout de suite trouver la liste de ses chiffres `[1, 2, 3, 4]`. Mais comment faire en général à partir d'un entier n ?

1. On aura besoin de la division euclidienne par 10 :

- on calcule par `n % 10` le **reste**, on l'appelle aussi ***n modulo 10***
- on calcule par `n // 10` le **quotient** de cette division.



2. Les commandes Python sont simplement `n % 10` et `n // 10`.

Exemple : `1234 % 10` vaut 4 ; `1234 // 10` vaut 123.

- 3.
- Le chiffre des unités de n s'obtient comme le reste modulo 10 : c'est $n\%10$. Exemple $1234\%10 = 4$.
 - Le chiffre des dizaines s'obtient à partir du quotient de la division de n par 10, puis en prenant le chiffre des unités de ce nombre : c'est donc $(n//10)\%10$. Exemple : $1234//10 = 123$, puis on a $123\%10 = 3$; le chiffre des dizaines de 1234 est bien 3.
 - Pour le chiffre de centaines, on calcule le quotient de la division de n par 100, puis on prend le chiffre des unités. Exemple $1234//100 = 12$; 2 est le chiffre des unités de 12 et c'est le chiffre des centaines de 1234. Remarque : diviser par 100, c'est diviser par 10, puis de nouveau par 10.
 - Pour le chiffre des milliers on calcule le quotient de la division par 1000 puis on prend le chiffre des unités...

Activité 3 (Trouver les chiffres d'un entier).

Objectifs : décomposer un entier en la liste de ses chiffres (en base 10).

Programme l'algorithme suivant en une fonction `entier_vers_decimale()`.

entier_vers_decimale()

Usage : `entier_vers_decimale(n)`

Entrée : un entier positif

Sortie : la liste de ses chiffres

Exemple : si l'entrée est 1234, la sortie est `[1, 2, 3, 4]`.

Algorithme.

Entrée : un entier $n > 0$

Sortie : la liste de ses chiffres

- Partir d'une liste vide.
- Tant que n n'est pas nul :
 - ajouter $n\%10$ au début de la liste,
 - faire $n \leftarrow n//10$.
- Le résultat est la liste.

Cours 4 (Écriture binaire avec Python).

Python calcule très bien l'écriture binaire d'un entier grâce à la fonction `bin()`.

python : bin()

Usage : `bin(n)`

Entrée : un entier

Sortie : l'écriture binaire de n sous la forme d'une chaîne de caractères commençant par `'0b'`

Exemple :

- `bin(37)` renvoie `'0b100101'`
- `bin(139)` renvoie `'0b10001011'`

Cours 5 (Calcul de l'écriture binaire).

Pour calculer l'écriture binaire d'un entier n , c'est la même méthode que pour calculer l'écriture décimale mais en remplaçant les divisions par 10 par des divisions par 2.

Nous avons donc besoin :

- de $n\%2$: le reste de la division euclidienne de n par 2 (appelé aussi n modulo 2) ; le reste vaut soit 0 soit 1.
- de $n//2$: le quotient de cette division.

Note que le reste $n\%2$ vaut soit 0 (quand n est pair) soit 1 (quand n est impair).

Voici la méthode générale pour calculer l'écriture binaire d'un entier :

- On part de l'entier dont on veut l'écriture binaire.
- On effectue une suite de divisions euclidiennes par 2 :
 - à chaque division, on obtient un reste qui vaut 0 ou 1 ;
 - on obtient un quotient que l'on divise de nouveau par 2... On s'arrête quand ce quotient est nul.
- On lit l'écriture binaire comme la suite des restes, mais en partant du dernier reste.

Exemple.

Calcul de l'écriture binaire de 14.

- On divise 14 par 2, le quotient est 7, le reste est 0.
- On divise 7 (le quotient précédent) par 2 : le nouveau quotient est 3, le nouveau reste est 1.
- On divise 3 par 2 : quotient 1, reste 1.
- On divise 1 par 2 : quotient 0, reste 1.
- C'est terminé (le dernier quotient est nul).
- Les restes successifs sont 0, 1, 1, 1. On lit l'écriture binaire à l'envers c'est 1.1.1.0.

Les divisions se font de gauche à droite, mais on lit les restes de droite à gauche.

14	2	7	2	3	2	1	2
0	7	1	3	1	1	1	0

←

Exemple.

Écriture binaire de 50.

50	2	25	2	12	2	6	2	3	2	1	2
0	25	1	12	0	6	0	3	1	1	1	0

Les restes successifs sont 0, 1, 0, 0, 1, 1, donc l'écriture binaire de 50 est 1.1.0.0.1.0.

Activité 4.

Objectifs : trouver l'écriture binaire d'un entier.

1. Calcule à la main l'écriture binaire des entiers suivants. Vérifie tes résultats à l'aide de la fonction

`bin()` de Python.

- 13, 18, 29, 31,
- 44, 48, 63, 64,
- 100, 135, 239, 1023.

2. Programme l'algorithme suivant en une fonction `entier_vers_binaire()`.

Algorithme.

Entrée : un entier $n > 0$

Sortie : son écriture binaire sous la forme d'une liste

- Partir d'une liste vide.
- Tant que n n'est pas nul :
 - ajouter $n\%2$ au début de la liste,
 - faire $n \leftarrow n//2$.
- Le résultat est la liste.

entier_vers_binaire()

Usage : `entier_vers_binaire(n)`

Entrée : un entier positif

Sortie : son écriture binaire sous forme d'une liste

Exemple : si l'entrée est 204, la sortie est `[1,1,0,0,1,1,0,0]`.

Vérifie que tes fonctions marchent bien :

- pars d'un entier n ,
- calcule son écriture binaire,
- calcule l'entier associé à cette écriture,
- tu dois retrouver l'entier de départ !