Représentation des entiers

Les entiers positifs 0, 1, 2, … aussi appelés entiers naturels, forment un ensemble, noté $\mathbb{N}$. Un nombre est un concept abstrait, qui désigne une quantité, comme **un** arbre, **deux** mains, **dix** doigts, **vingt** moutons, etc. Sa définition rigoureuse n'est pas aisée (voir les Entiers de Peano), et c'est pour cela que nous allons faire appel à l'intuition que chacun a du concept.

À chaque nombre est associé un nom, c'est à dire un son. En dépit de ce que certains philosophes ont affirmé, le son n'est pas le nombre, et il ne faut pas confondre le son *un* avec le nombre **un**: on pourrait bien décider un jour de le changer de nom et de l'appeler *pimplochonne*, cela restera toujours le nombre **un**.

De façon encore plus vicieuse, à chaque nombre est associé un symbole: à **un** on associe 1, à **deux** 2, à **dix** 10, etc. De même que pour le nom, le symbole n'est qu'une convention, une façon de mettre à l'écrit ce concept volatile qu'est le fait de posséder **un**, **deux**, **cent** moutons. Un même nombre peut être représenté par plein de symboles différents, comme l'arabe 2, le romain II, ou le grec β'.

Par la suite, lorsque on voudra parler d'un nombre, et pas de son symbole, on écrira son nom français en **gras**. Ainsi, **douze** est le concept, alors que 12 est seulement un symbole.

Chacun des différents systèmes de notations des nombres a ses avantages et désavantages. L'immense succès du système *positionnel* arabe, introduit en Europe par Leonardo Fibonacci au XIII siècle, est dû à la facilité avec laquelle on peut réaliser les quatre opérations avec celui-ci. Ici nous allons voir comment les mêmes symboles arabes peuvent être utilisés pour représenter différemment les nombres, et quels sont les avantages de ces autres représentations.

Le système arabe, ou la base dix

Un *système numéral* est une convention permettant de représenter les nombres avec une quantité finie de symboles. Les nombres étant infinis, il va de soit qu'on ne peut pas inventer un nouveau symbole pour chaque nombre. Le système arabe invente les dix symboles bien connus pour les dix premiers nombres:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Ces symboles, ou chiffres, sont juxtaposés pour représenter tout nombre grâce à une chaîne d'additions et de multiplications. En effet, comme il est bien connu, le symbole

1234

représente le nombre

$4 + 3 \times$ **dix** $+ 2 \times$ **cent** $+ 1 \times$ **mille** $=$ **mille-deux-cent-trente-quatre**

(ici les chiffres représentent leur nombre).

C'est la position du chiffre qui donne sa *magnitude*: plus précisément, le chiffre en **n**-ème position (en comptant de **zéro** en partant de la droite) sera multiplié par **dix** à la puissance **n** avant d'être ajouté aux autres.

D'autres bases

Le système arabe est dit *en base dix* parce qu'il est fondé sur les puissances de **dix**. On peut aisément imaginer d'autres systèmes où un autre nombre **n** prend la place du **dix**, ces systèmes sont appelé alors *en base* **n**. Il faudra faire attention, dans ce cas, à ne donner des chiffres que pour les nombres plus petits que **n** (tout comme pour la base **dix** on n'a des chiffres que jusqu'à **neuf**) si on veut garantir que cette représentation soit unique.

La base deux

La base un n'étant pas bien définie, le premier et plus simple système numéral positionnel est fourni par la base deux. On utilise seulement deux chiffres, 0 et 1, avec la signification habituelle, et on compose tout nombre comme pour la base dix, mais en utilisant des puissances de deux.

Par exemple, le symbole

1011

correspond au nombre

$1 + 1 \times$ **deux** $+ 0 \times$ **quatre** $+ 1 \times$ **huit** $=$ **onze**.

Lorsque le contexte n'est pas suffisant pour comprendre si une suite de chiffres représente un nombre en base deux ou dix, il est d'usage d'écrire $(1011)_2$ pour la base deux et $(1234)_{10}$ pour la base dix. Ainsi

$$(1011)_2 = (11)_{10} = \text{onze}.$$

Addition

L'algorithme d'addition classique marche également bien avec les nombres en base deux. Les retenues sont dans ce cas des puissance de **deux**, plutôt que des puissances de **dix**. Par exemple la somme de 10101 (**vingt-et-un**) et de 1111 (**quinze**) est calculée comme suit:

$$\begin{array}{cccccccc} {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & & \\ & 1 & 0 & 1 & 0 & 1 & + \\ & & 1 & 1 & 1 & 1 & = \\ \hline 1 & 0 & 0 & 1 & 0 & 0 & \end{array}$$

remarquez les retenues dès qu'on additionne deux ou plus chiffres 1, ce qui correspond bien à

$1 + 1 =$ **deux** $= 10$

et

$1 + 1 + 1 =$ **trois** $= 11$.

Multiplication

L'algorithme de multiplication marche également bien en base deux, et il est même plus simple à mettre en œuvre que pour d'autres bases. En effet, comme il n'y a que des chiffres 0 et 1 qui interviennent dans le calcul, les opérations intermédiaires ne comporteront que des multiplications par 0 et par 1. Voici l'exemple de la multiplication de 10101 (**vingt-et-un**) par de 1011 (**onze**):

$$\begin{array}{ccccccccc} & & & 1 & 0 & 1 & 0 & 1 & \times \\ & & & & 1 & 0 & 1 & 1 & = \\ \hline & & & 1 & 0 & 1 & 0 & 1 & + \\ & & 1 & 0 & 1 & 0 & 1 & & + \\ & & & & & 0 & & & + \\ 1 & 0 & 1 & 0 & 1 & & & & + \\ \hline 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & \end{array}$$

Il est maintenant évident que la puissance **n**-ième de **deux** s'écrit en base deux comme 1 suivi de **n** fois 0, analogiquement au cas de la base dix, où les puissances de **dix** s'écrivent 1, 10, 100, etc.

De plus, multiplier un nombre quelconque par une puissance de **deux**, revient à le décaler de **n** positions vers la gauche en ajoutant des 0 à droite. Réciproquement, diviser par une puissance de **deux** revient à décaler vers la droite, et les chiffres qui sont éliminés par ce décalage constituent le reste de la division Euclidienne. Par exemple:

$$\begin{aligned} (1011001)_2 \div 2^4 &= (101)_2, \\ (1011001)_2 \mod 2^4 &= (1001)_2, \end{aligned}$$

(**Rappel:** le symbole $\div$ dénote le *quotient*, **mod** dénote le *reste*.)

Nombres fractionnaires

La base deux se prête également bien à représenter autre chose que les naturels. D'un côté il est évident qu'avec l'emploi du signe $-$ on peut représenter les nombres négatifs. D'un autre côté, on peut représenter des nombres fractionnaires de la même façon qu'en base dix on peut écrire des nombres décimaux.

Par exemple, le symbole 123.456 est à interpréter comme

$$1 \times \text{cent} + 2 \times \text{dix} + 3 + 4 \times \frac{1}{\text{dix}} + 5 \times \frac{1}{\text{cent}} + 6 \times \frac{1}{\text{mille}}.$$

De façon analogue, le symbole 1.011 est à interpréter en base deux comme

$$1 + \frac{1}{\text{quatre}} + \frac{1}{\text{huit}}.$$

Les autres bases

De la même façon qu'on a pu représenter les nombres en base deux, n'importe quel entier positif peut servir de base. Le principe est le même: si on se fixe une base **b**, on aura besoin d'un chiffre pour chaque entier plus petit que **b**. Par exemple pour la base cinq, on utilisera les chiffres 0, 1, 2, 3 et 4, et les nombres plus grands seront représentés par des sommes de puissances de **cinq**. Ainsi, on aura

$$(1234)_5 = 1 \times \text{cent-vingt-cinq} + 2 \times \text{vingt-cinq} + 3 \times 5 + 4 = \text{cent-quatre-vingt-quatorze}.$$

Un peu plus délicat est le cas des bases plus grandes que dix. Dans ce cas, n'ayant pas assez de symboles pour représenter tous les nombres plus petits que **b**, on est obligés d'en ajouter des nouveaux. La solution classique, consiste à donner la signification usuelle aux chiffres de 0 à 9, et à utiliser les lettres de l'alphabet pour les chiffres suivants. Le cas le plus important est celui de la base 16, où on prend la convention

- A = **dix**
- B = **onze**
- C = **douze**
- D = **treize**
- E = **quatorze**
- F = **quinze**

Les algorithmes d'addition, de multiplication, etc. marchent exactement comme pour les bases deux et dix. Il n'y a rien de spécial à remarquer sur ces autres bases.

Conversion entre bases

Savoir convertir de tête et rapidement un nombre d'une base vers une autre est un atout fondamental pour un programmeur. Il y a deux cas fondamentaux: le cas général et celui des puissances d'une base.

Cas général

On va seulement discuter le passage entre la base dix et une autre base **b**. Faire une conversion entre deux bases quelconques ne présente aucune différence théorique, et est souvent plus aisément réalisé en passant par la base dix.

À partir de maintenant, on arrête d'écrire les nombres en lettres, et on utilise la base dix pour les représenter, sauf là où c'est explicitement indiqué. On a déjà vu comment on convertit un nombre écrit en base **b** vers la base dix. Il suffit de l'écrire comme une somme de puissances de **b** et de faire le calcul.

Par exemple, en supposant que 1, 2, 3, 4 soient plus petits que **b**, le nombre $(1234)_b$ vaut

$$(1234)_b = 1 \times b^3 + 2 \times b^2 + 3 \times b^1 + 4 \times b^0.$$

Le passage inverse est similaire. On écrit **dix** et les chiffres de 0 à 9 en base b, et on fait le calcul comme ci-dessus. Par exemple, pour convertir $(1234)_{10}$ en base deux, on écrit

$$\begin{aligned}
10 &= (1010)_2, \\
0 &= (0)_2, \\
1 &= (1)_2, \\
2 &= (10)_2, \\
3 &= (11)_2, \\
4 &= (100)_2, \\
&\vdots \\
1234 &= (1)_2 \times (1010)_2^3 + (10)_2 \times (1010)_2^2 + (11)_2 \times (1010)_2 + (100)_2.
\end{aligned}$$

Ce calcul n'est pas très aisé, et en général on préfère un algorithme plus simple pour effectuer la conversion de la base dix. L'algorithme se base sur les propriétés de la division par **b**. Pour fixer les idées, prenons **b**=2. Si on divise de façon répété un nombre écrit en base deux par **deux**, on peut lire son écriture en base deux dans sa suite de restes. Par exemple, dans

$$\begin{aligned}
(10110)_2 &= (1011)_2 \times 2 + 0, \\
(1011)_2 &= (101)_2 \times 2 + 1, \\
(101)_2 &= (10)_2 \times 2 + 1, \\
(10)_2 &= (1)_2 \times 2 + 0, \\
(1)_2 &= (0)_2 \times 2 + 1,
\end{aligned}$$

on voit que le nombre $(10110)_2$ réapparaît à la droite en lisant du bas vers le haut. Ceci n'est pas très utile, mais nous n'avons utilisé que des propriétés des entiers (et non pas de leurs représentation) pour montrer ce phénomène. Si maintenant l'entier $(10110)_2$ est écrit en base dix, plutôt qu'en base deux, nous pouvons appliquer la même méthode. Prenons par exemple 30, on a

$$\begin{aligned}
30 &= 15 \times 2 + 0, \\
15 &= 7 \times 2 + 1, \\
7 &= 3 \times 2 + 1, \\
3 &= 1 \times 2 + 1, \\
1 &= 0 \times 2 + 1.
\end{aligned}$$

On en déduit que $30 = (11110)_2$. Le même algorithme peut s'utiliser en remplaçant **deux** par n'importe quelle base.

Conversion entre des bases qui sont l'une la puissance de l'autre

La conversion entre la base deux et les bases quatre, huit, seize, etc. peut se faire de la même façon que le cas général, cependant il est possible d'effectuer le changement avec beaucoup moins de travail en observant, par exemple, que $16 = 2^4$.

En effet, si on considère le nombre $(1B3)_{16}$, on voit que

$$(1B3)_{16} = 1 \times 16^2 + 11 \times 16 + 3 = 1 \times 2^8 + 11 \times 2^4 + 3.$$

Si maintenant on convertit chaque chiffre de la base seize en base deux, on a, par exemple, que

$$11 = (1011)_2 = 2^3 + 2^1 + 2^0$$

et donc

$$11 \times 2^4 = (1011)_2 \times 2^4 = 2^7 + 2^5 + 2^4.$$

En conclusion, on a

$$(1B3)_{16} = (1)_2 \times 2^8 + (1011)_2 \times 2^4 + (11)_2 = 2^8 + (2^7 + 2^5 + 2^4) + (2+1) = (110110011)_2,$$

où on a laissé des parenthèses pour mettre en évidence les groupes provenant de chacun des chiffres hexadécimaux.

De façon générale, comme $16 = 2^4$, les chiffres hexadécimaux s'écrivent sur au plus quatre chiffres binaires. Par conséquent, pour convertir de la base seize à la base deux on écrit chacun des chiffres en base deux, on ajoute suffisamment de 0 pour que ça tienne sur quatre chiffres (par exemple, $3 = (11)_2 = (0011)_2$) et on met le tout bout à bout.

La conversion inverse est aussi aisée. On part d'un nombre en base deux, on le découpe en blocs de quatre chiffres binaires, et on convertit chaque bloc en un chiffre hexadécimal. Par exemple

$$(10100110011)_2 = (101)_2 \times 2^8 + (0011)_2 \times 2^4 + (0011)_2 = (533)_{16}.$$

Le même principe s'applique aux conversions entre la base deux et les bases quatre et huit (avec des blocs de longueur deux et trois, respectivement). Voici un exemple de passage de la base deux à la base huit, où il suffit de remarquer que $8 = 2^3$:

$$\begin{aligned}(1011010101)_2 &= (1)_2 \times 2^9 + (011)_2 \times 2^6 + (010)_2 \times 2^3 + (101)_2 \\ &= (1)_8 \times 8^3 + (3)_8 \times 8^2 + (2)_8 \times 8 + (5)_8 \\ &= (1325)_8\end{aligned}$$

Inversement, pour convertir par exemple $(625)_8$ en base deux il suffit de remplacer chacun des trois chiffres dans la représentation octale, par leur représentation binaire sur 3 bits: $(6)_8 = (110)_2$, $(2)_8 = (010)_2$, $(5)_8 = (101)_2$. Il suffit ensuite de placer les trois suites bout à bout dans le bon ordre pour obtenir finalement

$$(625)_8 = (110010101)_2.$$

Le même principe s'applique aussi aux conversions entre la base quatre et la base seize (avec des blocs de longueur deux). Pour passer de la base huit à la base seize, on peut d'abord aller de la base huit à la base deux, et ensuite de la base deux à la base seize, et inversement. Enfin, les même principes s'appliquent aussi aux bases 3, 9, 27, … aux bases 10, 100, …, etc.
