Introduction à l'informatique

Cours complet

Jean-Christophe Dubacq

IUT de Villetaneuse

S1 2016

Organisation du module

Remerciements

- Les cours et exercices de ce module sont directement inspirés des documents de **M. Bosc**, **J.-C. Dubacq** et **G. Santini**.
- D'autres intervenants ont participé à l'élaboration des supports.

Les enseignements

- 12 sessions de 4h et du travail personnel …
- 6 sessions pour la présentation générale du système d'exploitation Linux,
- 6 sessions pour la théorie de base du codage informatique

Votre présence est obligatoire

- Contrôle des présences.
- Rapport des absences.

L'évaluation

- Une composition après la sixième session (sur papier ou sur ordinateur).
- Une composition à la fin du module (sur papier ou sur ordinateur).

Plan

Plan

Exercices

Qu'est-ce que l'information

Q1 Proposez différents symbolismes utilisés pour noter un nombre. Donnez l'exemple de leur notation avec le nombre 9. Donnez des inconvénients de votre méthode.

Q2 Travaillez en paire (ou triplettes). Proposez une méthode pour transmettre d'une personne à une autre le résultat d'un lancer de dé (lancer caché par la première personne, la deuxième doit pouvoir énoncer le résultat). Votre méthode fonctionne-t-elle si le dé comporte 20 faces ? Et si le dé est à six faces mais étiqueté par des couleurs ?

Qu'est-ce qu'une information ?

Information

Une information est une donnée que l'on peut interpréter pour se construire une représentation du monde sur laquelle on peut agir.

Claude Shannon a été l'un des premiers à définir l'information comme une quantité mesurable.

L'information de Shannon n'est pas associé au sens ou à la cognition.

- Il s'est intéressé à quantifier des sources aléatoires
- L'information diminue l'incertitude sur une source aléatoire
- On mesure donc la quantité d'information relative à un événement
- Par exemple, si on a six possibilités pour un dé, l'information permet de savoir quelle face ; ou au moins d'éliminer des possibilités

La vision cognitive de l'information

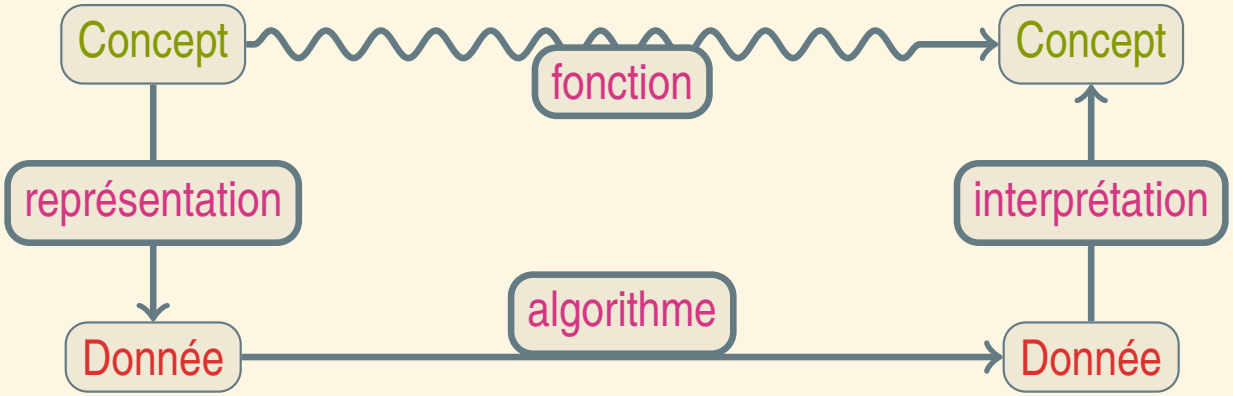

On ne sait pas mesurer le *sens* des choses.


Une même information peut avoir plusieurs représentations très distinctes.


Le nombre quatorze : 14 ou XIV ou IIIIIIIIIIIIII


Une même donnée peut être interprétée de plusieurs façons → informations très distinctes.


Inversement : XIV est un mot ou un nombre



L'information digitale

- Les systèmes d'informations (et les ordinateurs en particulier) ne sont pas équipés pour traiter n'importe quel type de données.
- Toutes les informations sont représentées sous forme de nombres pour être traitées par les ordinateurs.
- Le monde réel est *analogique*, la représentation des ordinateurs est *numérique* (ou *digitale*).
- Nous considérerons que nous avons toujours affaire à des problèmes représentables par des nombres.

Exercices

Digital ou analogique ?

Q3 Est-ce que les données suivantes sont digitales ou analogiques :

- Le fait d'avoir un rendez-vous à une certaine heure un certain jour
- La pression de l'air
- Le résultat (stable) d'un dé
- Votre nom de famille
- Votre nombre de frères et sœurs
- Votre taille
- La couleur de vos yeux

Plan

L'information mesurée

bit

Le bit est la quantité d'information qui permet de choisir complètement entre deux issues distinctes d'un événement.
Le mot de *bit* est l'abréviation de *binary digit*.

Mesure de l'information

Pour exprimer k choix possibles distincts, il faut $\lceil \log_2(k) \rceil$ bits distincts. ($\lceil x \rceil$ est l'arrondi par dessus).
k bits d'information permettent de distinguer 2^k choix.

Exemple (Jeu du fakir)

Je peux deviner n'importe quel nombre entre 0 et 100 par 7 questions à réponse oui ou non ($\lceil \log_2(100) \rceil = 7$).

▶ Tester

Binaire et décimal : unités

- Un groupe de 8 bits est désigné par le terme *octet*.
- Abréviations : bit=b, octet=o ou B (anglais). À éviter !
- Multiples : kilo, mega, giga, tera (voir mémento).

Un gros kilo ou un petit ?


kilo-octet souvent $1\,024 = 2^{10}$ octets et non $10^3 = 1\,000$.


Utilisez le contexte !


Parfois (toujours dans ce cours), préfixe ki ou Mi pour 2^{10} et 2^{20}.

Octet ou byte ?

En anglais, octet=*byte*. Ne pas confondre un Mb, un MB, un Mib et un MiB.

Memento : préfixes et unités

L'échelle décimale

Préfixe	Valeur	Ab.	Préfixe	Valeur	Ab.
kilo	10^3	k	milli	10^{-3}	m
mega	10^6	M	micro	10^{-6}	µ
giga	10^9	G	nano	10^{-9}	n
tera	10^{12}	T	pico	10^{-12}	p
peta	10^{15}	P	exa	10^{18}	E

L'échelle binaire

Préfixe	Valeur		Ab.
kilo	2^{10}	1024	K ou ki
mega	2^{20}	1048576	Mi
giga	2^{30}	1073741824	Gi

Seule exception beaucoup utilisée : kilo-octets souvent 1024 octets. Faux pour kilo-bits (toujours 1000 bits).

Quelques ordres de grandeur

Quantité

- 10^3 bits : carte à bande magnétique
- 10^6 bits : un fax d'une page
- 10^9 bits : Capacité d'un CD ou du génome humain
- 10^{12} bits : Un disque dur moyen en 2008
- 10^{15} bits : 1/10^e taille des serveurs de Google
- 10^{18} bits : Tout ce qui est imprimé dans le monde.

Débit

- 1 b/s : vieille sonde spatiale (9 b/s), morse (40 b/s)
- 10^3 b/s : 2G (9,6 kb/s), modems (56 kb/s)
- 10^6 b/s : ADSL (20 Mb/s)
- 10^9 b/s : Réseau local Gigabit (1 Gb/s), USB (0,48 Gb/s), Infiniband (60 Gb/s)
- 10^{12} b/s : Trafic total USA cumulé sur internet (12 Tb/s)
- 10^{15} b/s : Trafic total international sur internet (0.5 Pb/s)

Exercices

Conversions

Q4 Convertissez 24×10^8 bits en Go.

Q5 Convertissez 2^{16} octets en Mib. Donnez une approximation en Mb. Quel est l'ordre de grandeur de l'approximation faite ?

Q6 Un élément d'ordinateur est capable d'émettre 1024 bits en 0,5 nanosecondes. Quel est le débit (quantité d'information divisée par le temps) de cet élément en bits par secondes ? Quelle est la bonne unité pour ce débit ?

Plan

L'information quantifiée

L'information n'est pas toujours disponible dans la nature sous forme digitale. Il est donc nécessaire, pour la faire traiter par un ordinateur, de la digitaliser.
La digitalisation se fait presque toujours de la même façon :

- Filtrage perceptuel physique
- Découpage (volumique) (pour les phénomènes multidimensionnels)
- Échantillonnage (pour les phénomènes temporels)
- Quantification (réduction à un nombre d'états finis)
- Filtrage perceptuel numérique

Nous reverrons un peu mieux ces notions ultérieurement. Les deux premières étapes forment la *discrétisation* (spatiale ou temporelle), et la troisième la *quantification*.

L'information discrétisée

Découpage spatial ou pixellisation

La résolution d'échantillonnage influe sur la fidélité de l'image

L'information est perdue : on ne peut pas retrouver la précision.

Source de l'image : Image Lena `http://www.cs.cmu.edu/~chuck/lennapg/`

L'information discrétisée

Découpage spatial ou pixellisation

La résolution d'échantillonnage influe sur la fidélité de l'image

L'information est perdue : on ne peut pas retrouver la précision.

Source de l'image : Image Lena `http://www.cs.cmu.edu/~chuck/lennapg/`

L'information discrétisée

Découpage spatial ou pixellisation

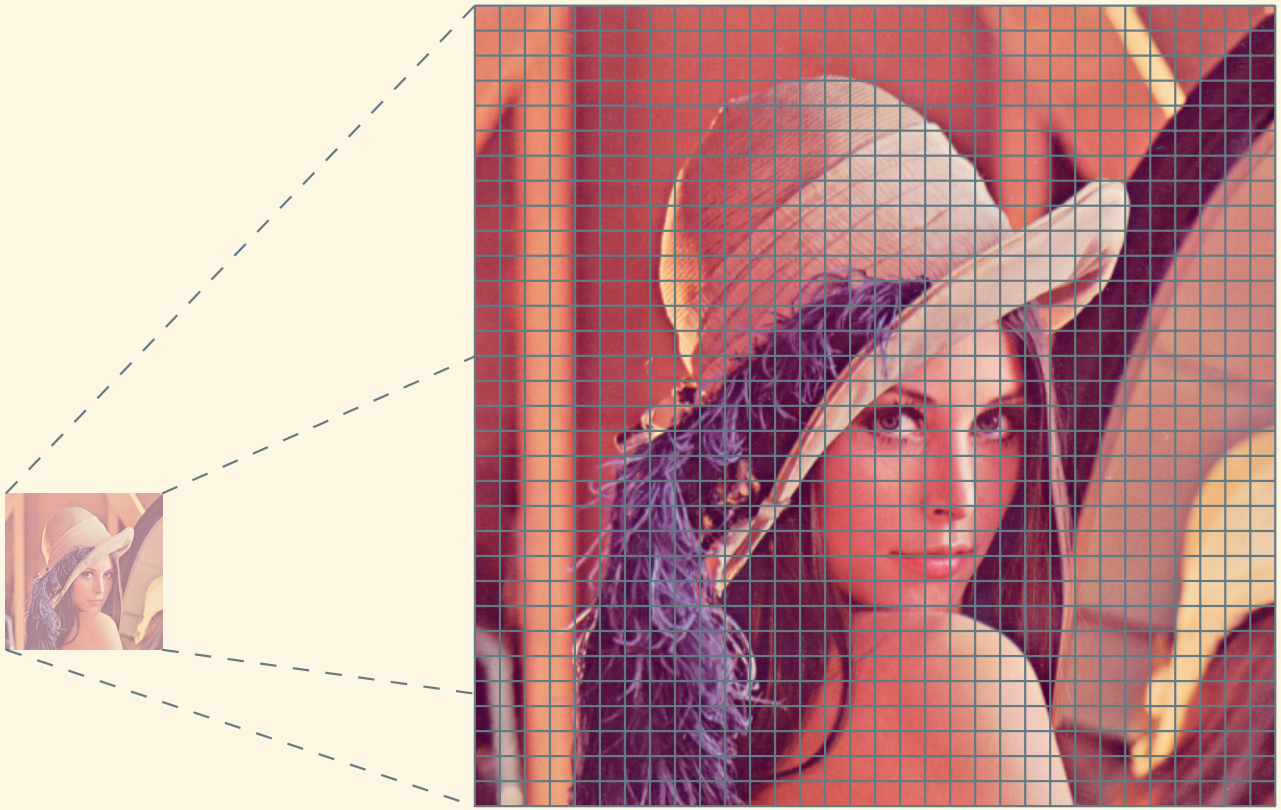
La résolution d'échantillonnage influe sur la fidélité de l'image

L'information est perdue : on ne peut pas retrouver la précision.

Source de l'image : Image Lena `http://www.cs.cmu.edu/~chuck/lennapg/`

L'information discrétisée

Découpage spatial ou pixellisation



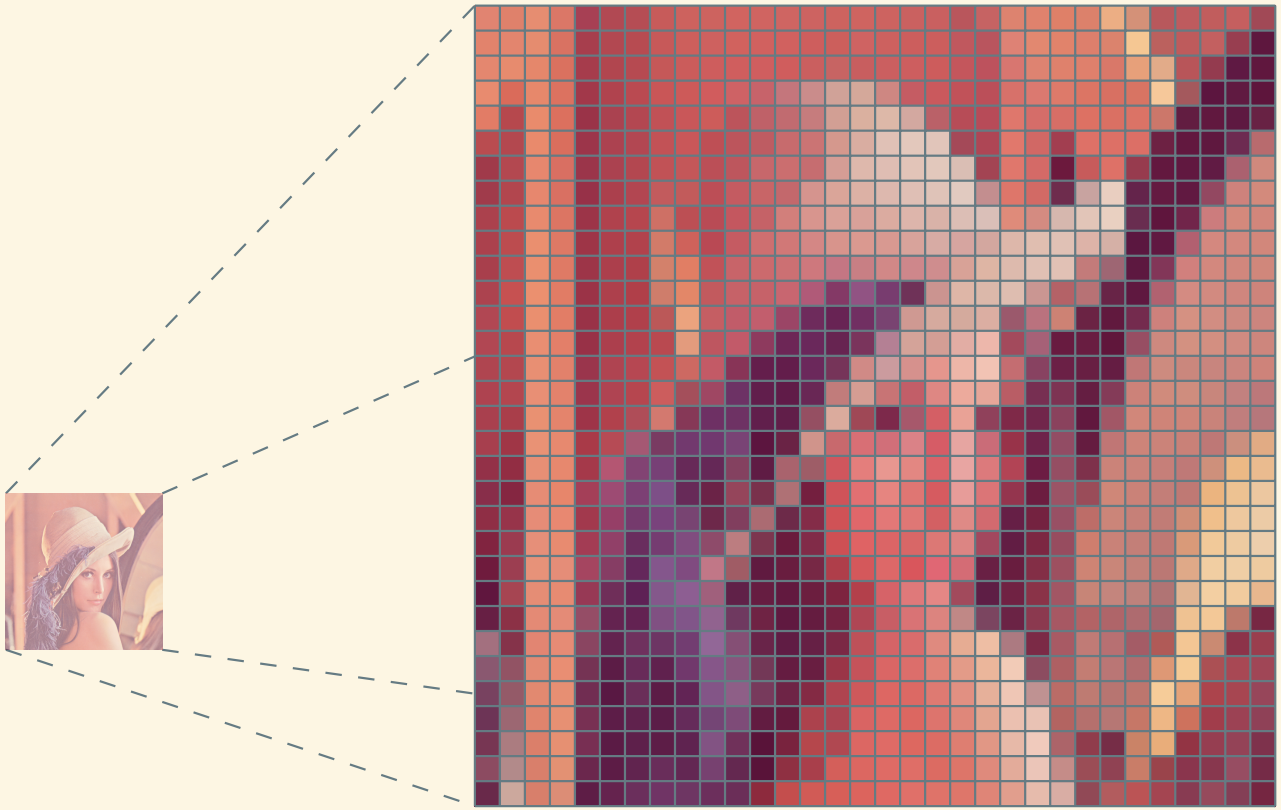
La résolution d'échantillonnage influe sur la fidélité de l'image

L'information est perdue : on ne peut pas retrouver la précision.

Source de l'image : Image Lena `http://www.cs.cmu.edu/~chuck/lennapg/`

L'information discrétisée

Découpage spatial ou pixellisation



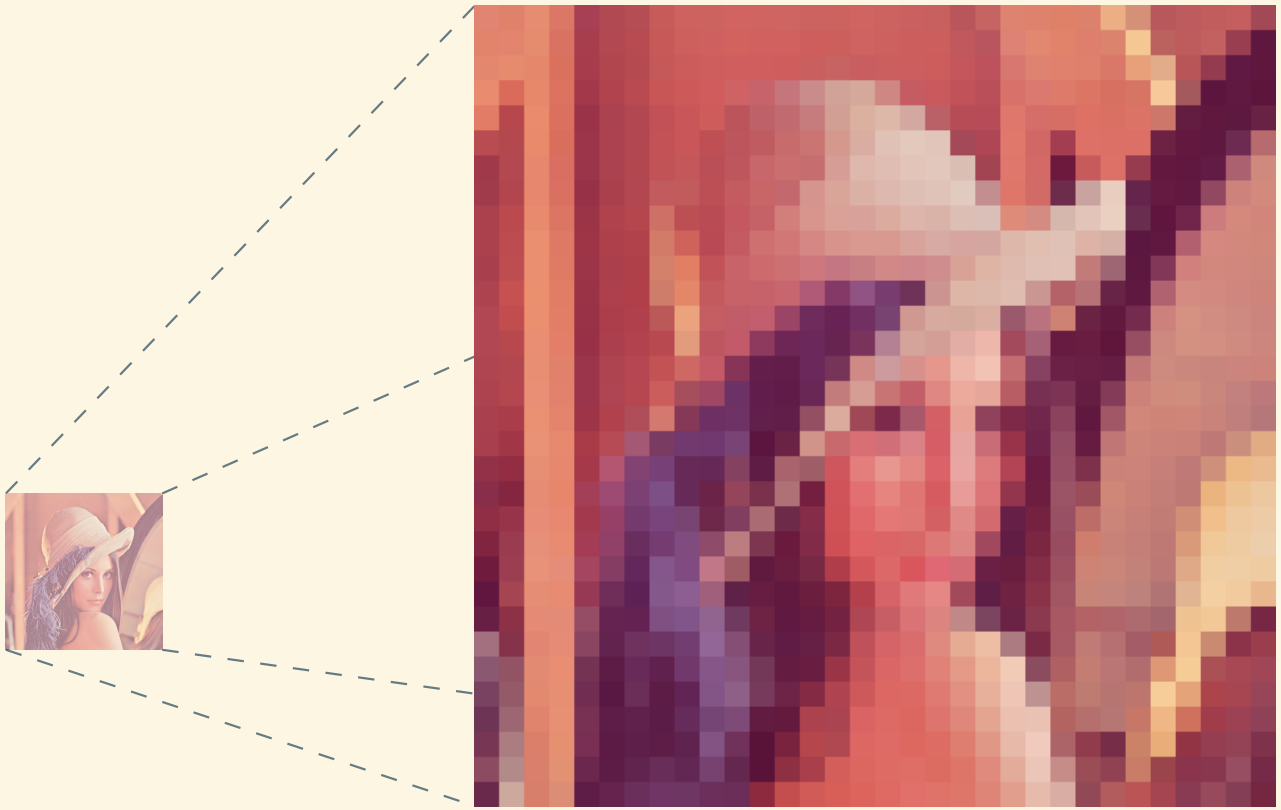
La résolution d'échantillonnage influe sur la fidélité de l'image

L'information est perdue : on ne peut pas retrouver la précision.

Source de l'image : Image Lena `http://www.cs.cmu.edu/~chuck/lennapg/`

L'information discrétisée

Découpage spatial ou pixellisation



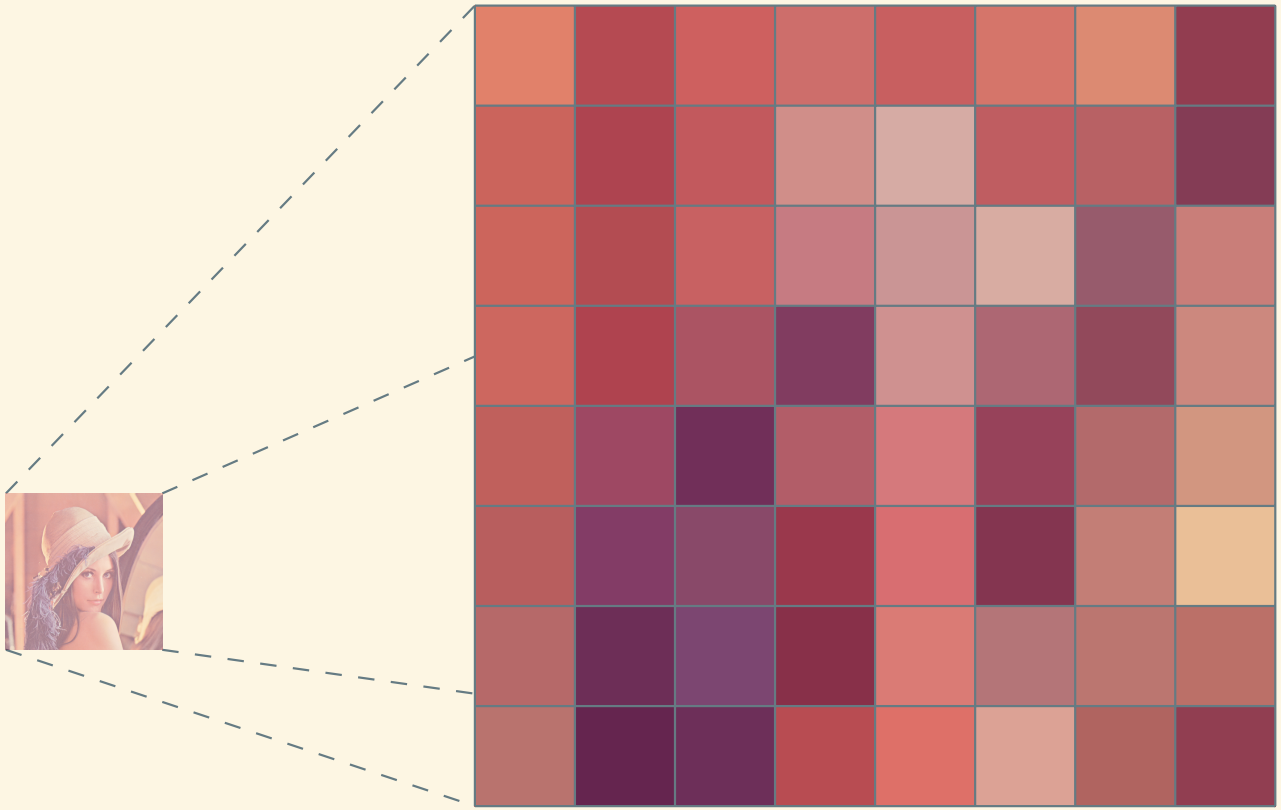
La résolution d'échantillonnage influe sur la fidélité de l'image

L'information est perdue : on ne peut pas retrouver la précision.

Source de l'image : Image Lena `http://www.cs.cmu.edu/~chuck/lennapg/`

L'information discrétisée

Découpage spatial ou pixellisation



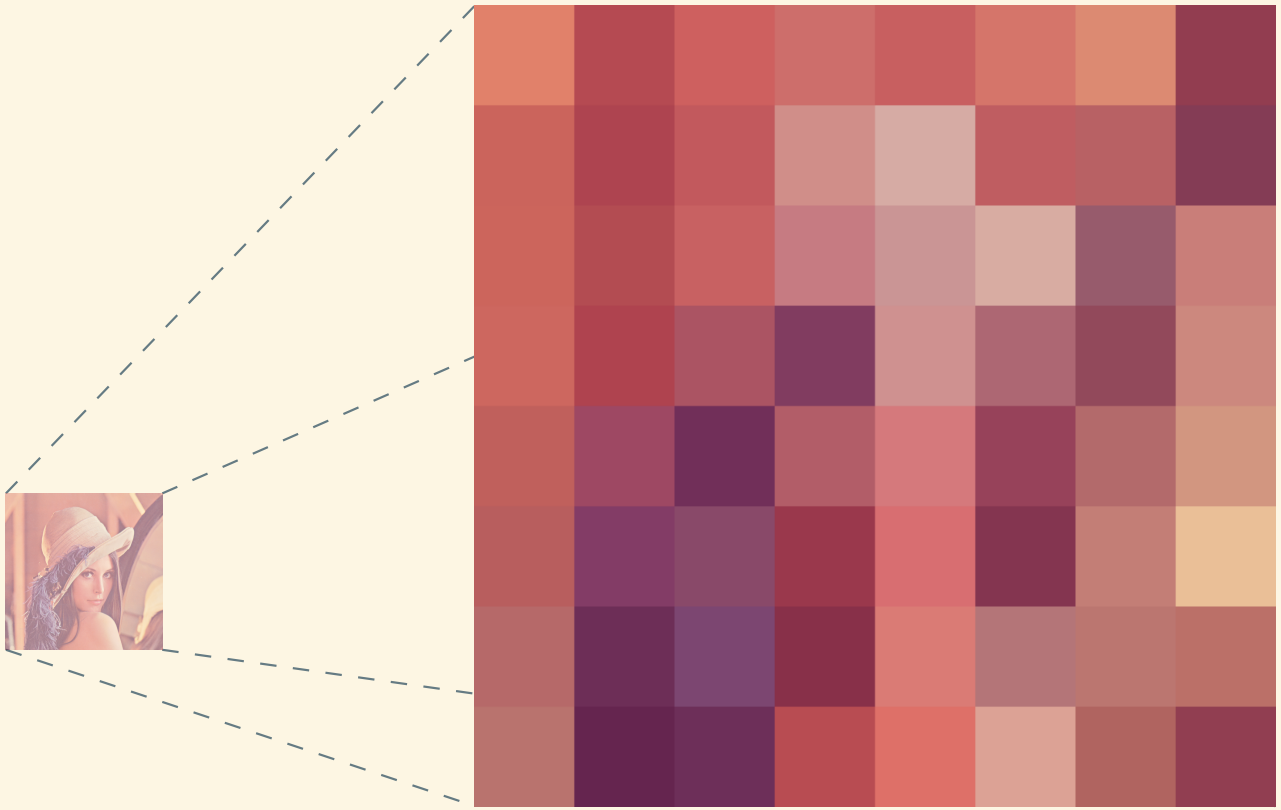
La résolution d'échantillonnage influe sur la fidélité de l'image

L'information est perdue : on ne peut pas retrouver la précision.

Source de l'image : Image Lena `http://www.cs.cmu.edu/~chuck/lennapg/`

L'information discrétisée

Découpage spatial ou pixellisation



La résolution d'échantillonnage influe sur la fidélité de l'image

L'information est perdue : on ne peut pas retrouver la précision.

Source de l'image : Image Lena `http://www.cs.cmu.edu/~chuck/lennapg/`

L'information discrétisée

Découpage spatial ou pixellisation

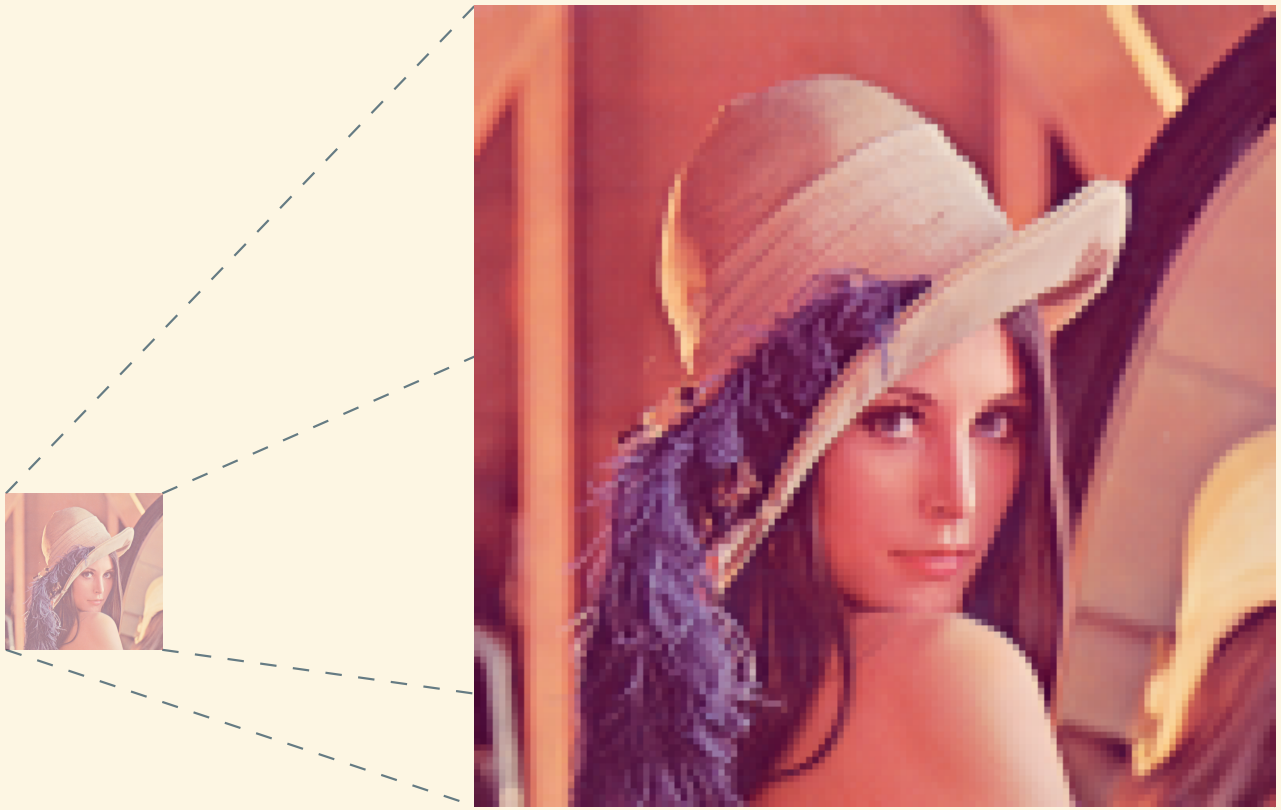
La résolution d'échantillonnage influe sur la fidélité de l'image

L'information est perdue : on ne peut pas retrouver la précision.

Source de l'image : Image Lena `http://www.cs.cmu.edu/~chuck/lennapg/`

L'information discrétisée

Découpage spatial ou pixellisation



La résolution d'échantillonnage influe sur la fidélité de l'image

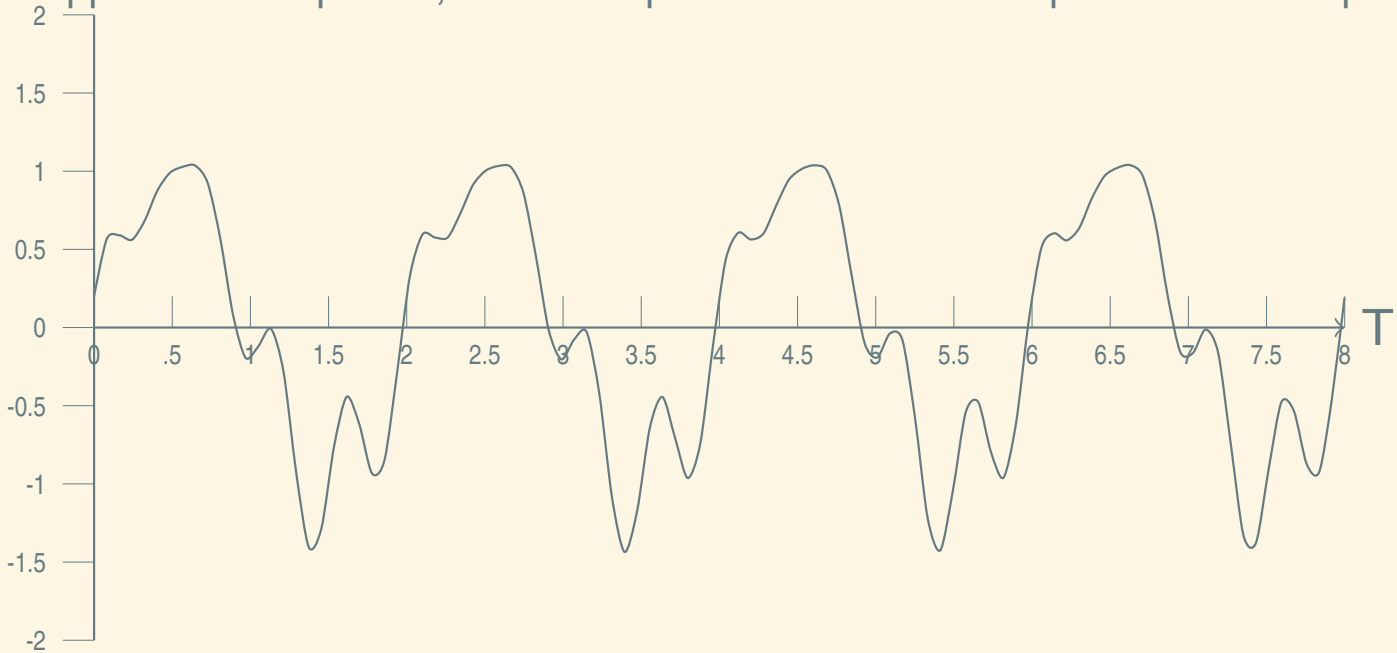
L'information est perdue : on ne peut pas retrouver la précision.

Source de l'image : Image Lena `http://www.cs.cmu.edu/~chuck/lennapg/`

L'information discrétisée

Échantillonnage temporel

Rappel : une fréquence, c'est une quantité d'événements par unité de temps ($1\mathrm{Hz} = 1s^{-1}$).



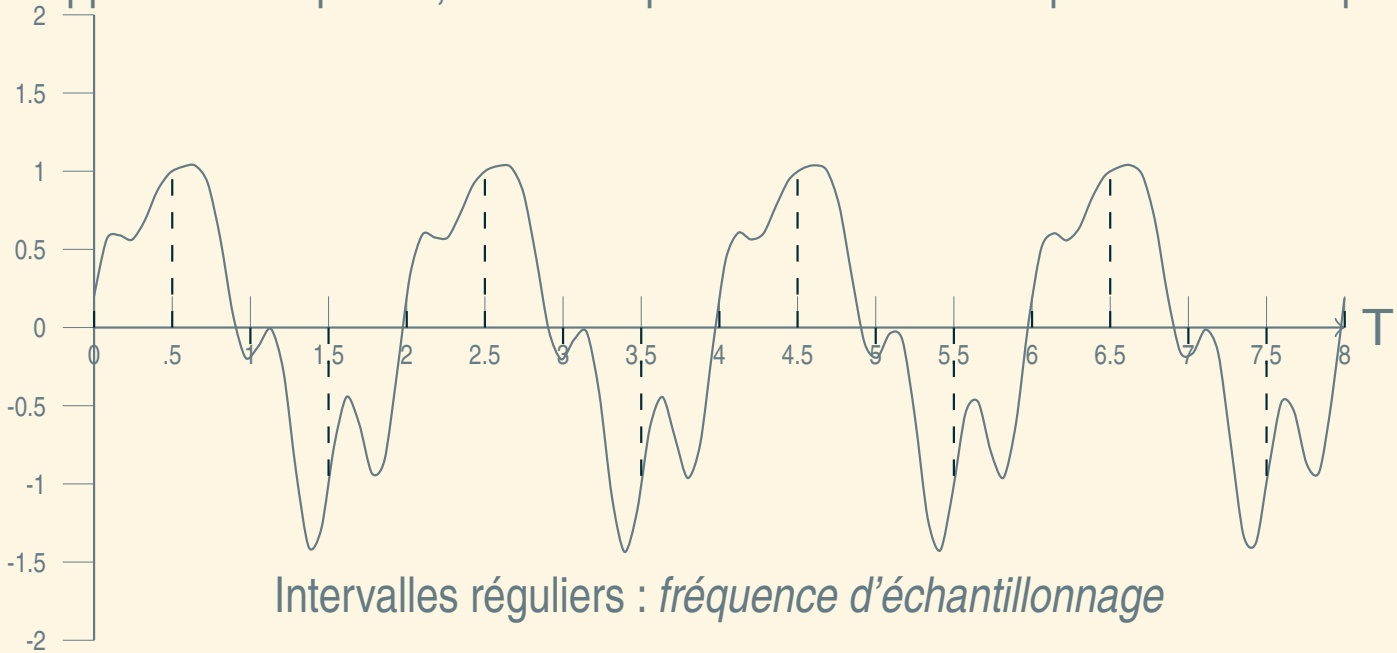
Pour pouvoir reconstruire exactement un signal périodique qui peut être décomposé avec une fréquence maximale, sa fréquence d'échantillonnage f_e doit vérifier *(Théorème d'échantillonnage de Nyquist-Shannon)* :

$$f_e \geqslant 2f_{\mathrm{Max}}$$

L'information discrétisée

Échantillonnage temporel

Rappel : une fréquence, c'est une quantité d'événements par unité de temps ($1\mathrm{Hz} = 1s^{-1}$).



Intervalles réguliers : *fréquence d'échantillonnage*

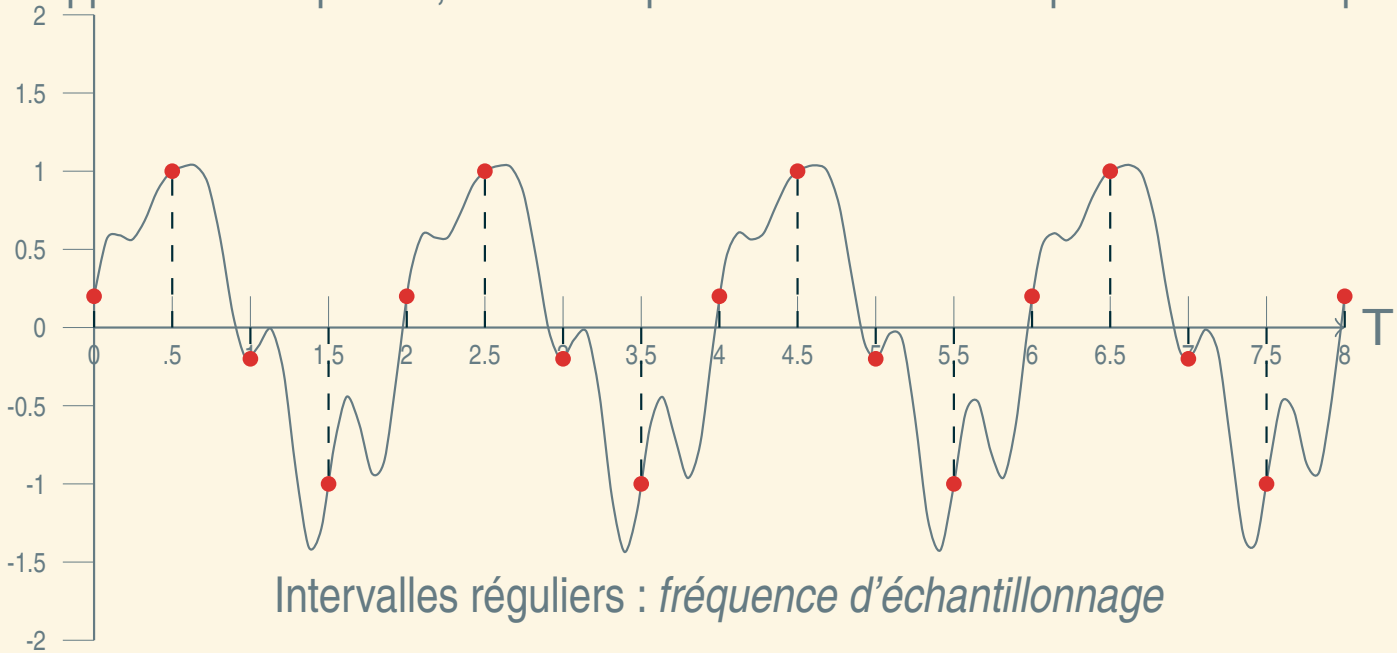
Pour pouvoir reconstruire exactement un signal périodique qui peut être décomposé avec une fréquence maximale, sa fréquence d'échantillonnage f_e doit vérifier *(Théorème d'échantillonnage de Nyquist-Shannon)* :

$$f_e \geqslant 2f_{\mathrm{Max}}$$

L'information discrétisée

Échantillonnage temporel

Rappel : une fréquence, c'est une quantité d'événements par unité de temps ($1\text{Hz} = 1s^{-1}$).



Intervalles réguliers : *fréquence d'échantillonnage*

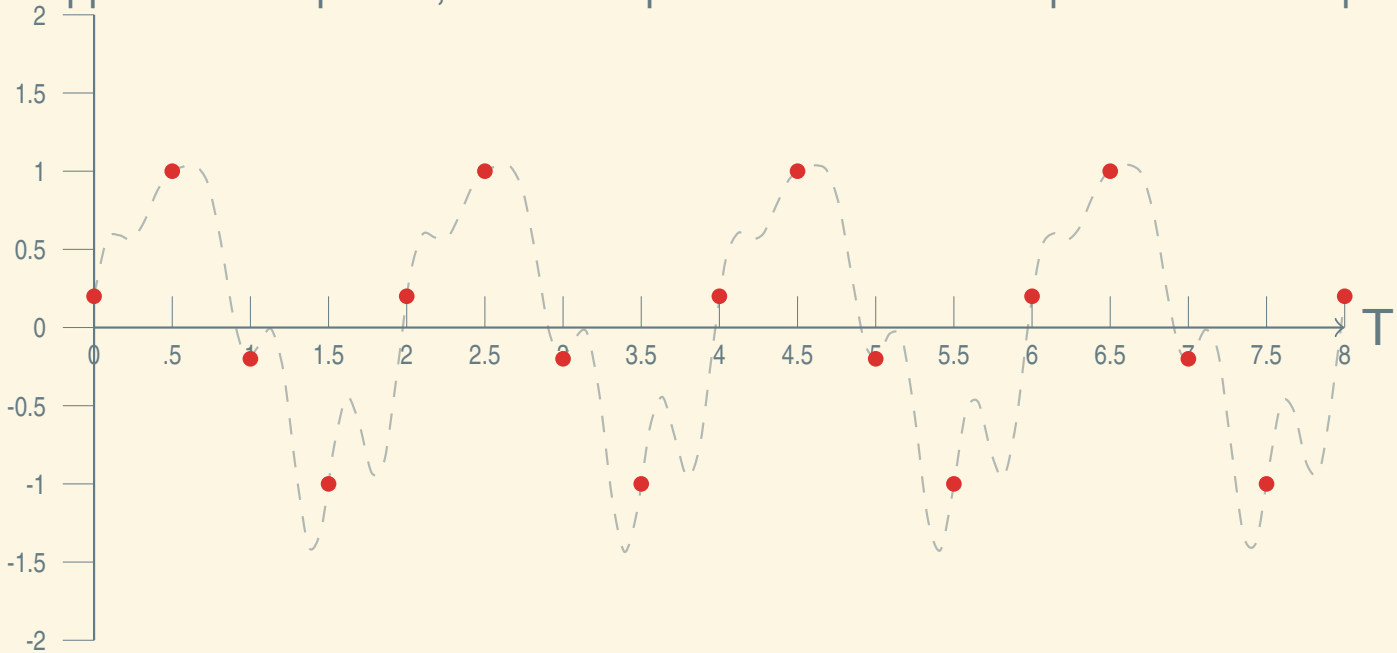
Pour pouvoir reconstruire exactement un signal périodique qui peut être décomposé avec une fréquence maximale, sa fréquence d'échantillonnage f_e doit vérifier *(Théorème d'échantillonnage de Nyquist-Shannon)* :

$$f_e \geqslant 2f_{\text{Max}}$$

L'information discrétisée

Échantillonnage temporel

Rappel : une fréquence, c'est une quantité d'événements par unité de temps ($1\mathrm{Hz} = 1s^{-1}$).



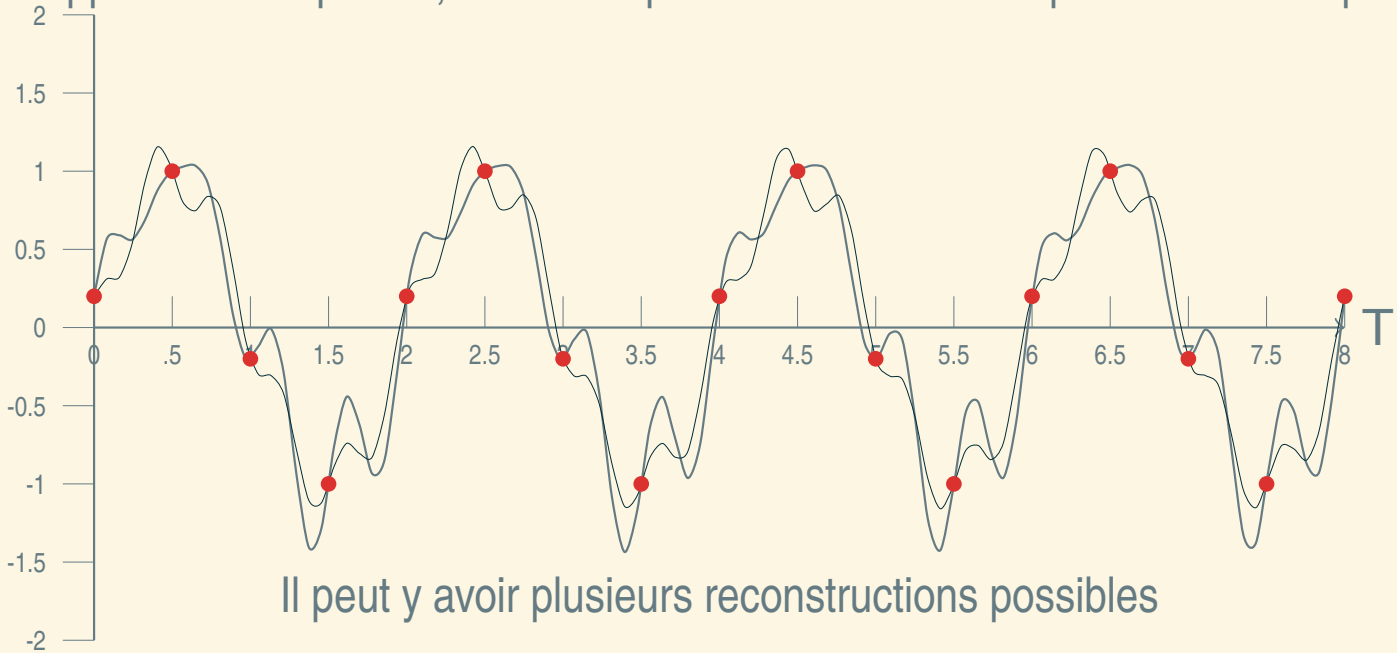
Pour pouvoir reconstruire exactement un signal périodique qui peut être décomposé avec une fréquence maximale, sa fréquence d'échantillonnage f_e doit vérifier *(Théorème d'échantillonnage de Nyquist-Shannon)* :

$$f_e \geqslant 2f_{\mathrm{Max}}$$

L'information discrétisée

Échantillonnage temporel

Rappel : une fréquence, c'est une quantité d'événements par unité de temps ($1\mathrm{Hz} = 1s^{-1}$).



Il peut y avoir plusieurs reconstructions possibles

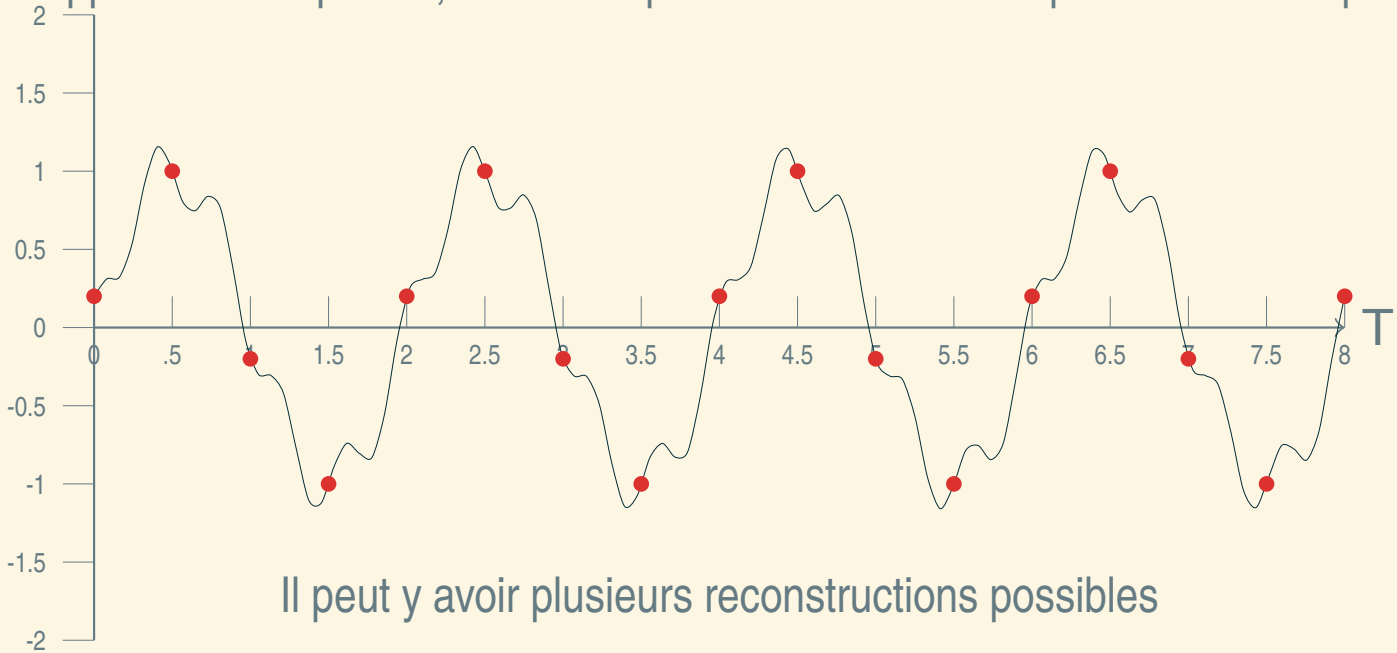
Pour pouvoir reconstruire exactement un signal périodique qui peut être décomposé avec une fréquence maximale, sa fréquence d'échantillonnage f_e doit vérifier *(Théorème d'échantillonnage de Nyquist-Shannon)* :

$$f_e \geqslant 2f_{\mathrm{Max}}$$

L'information discrétisée

Échantillonnage temporel

Rappel : une fréquence, c'est une quantité d'événements par unité de temps ($1\mathrm{Hz} = 1s^{-1}$).



Il peut y avoir plusieurs reconstructions possibles

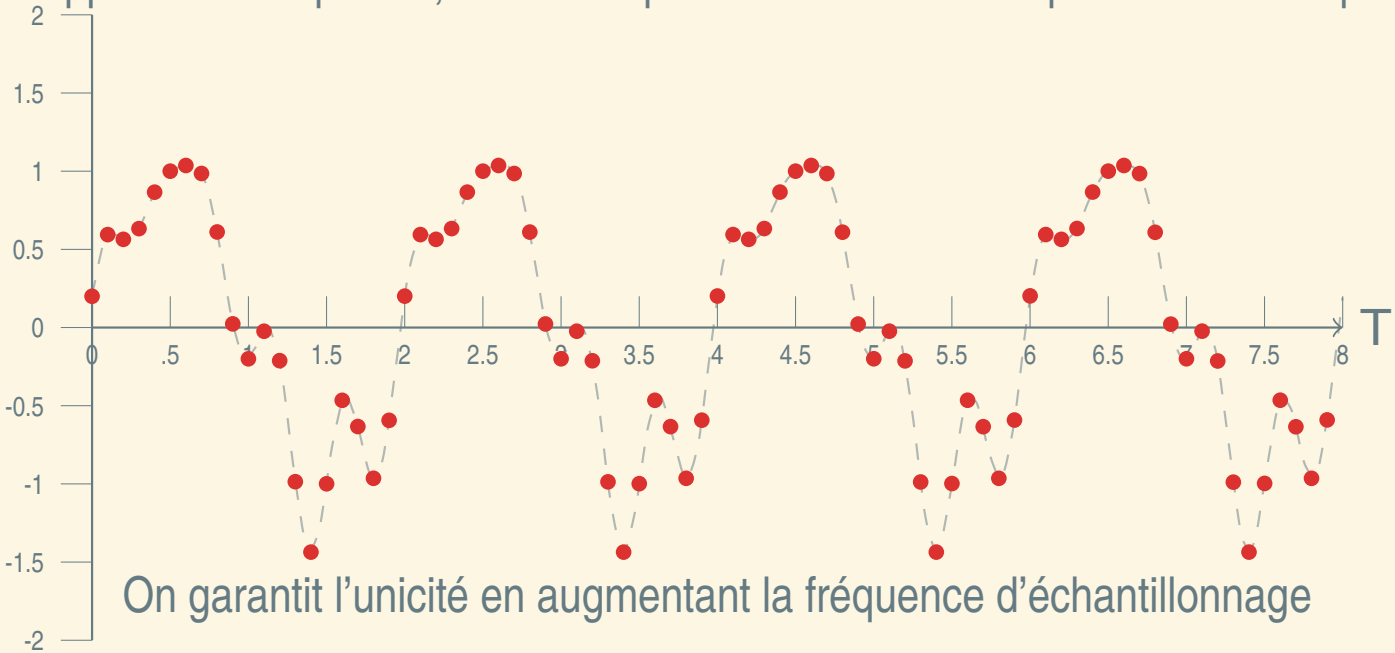
Pour pouvoir reconstruire exactement un signal périodique qui peut être décomposé avec une fréquence maximale, sa fréquence d'échantillonnage f_e doit vérifier *(Théorème d'échantillonnage de Nyquist-Shannon)* :

$$f_e \geqslant 2f_{\mathrm{Max}}$$

L'information discrétisée

Échantillonnage temporel

Rappel : une fréquence, c'est une quantité d'événements par unité de temps ($1\mathrm{Hz} = 1s^{-1}$).



On garantit l'unicité en augmentant la fréquence d'échantillonnage

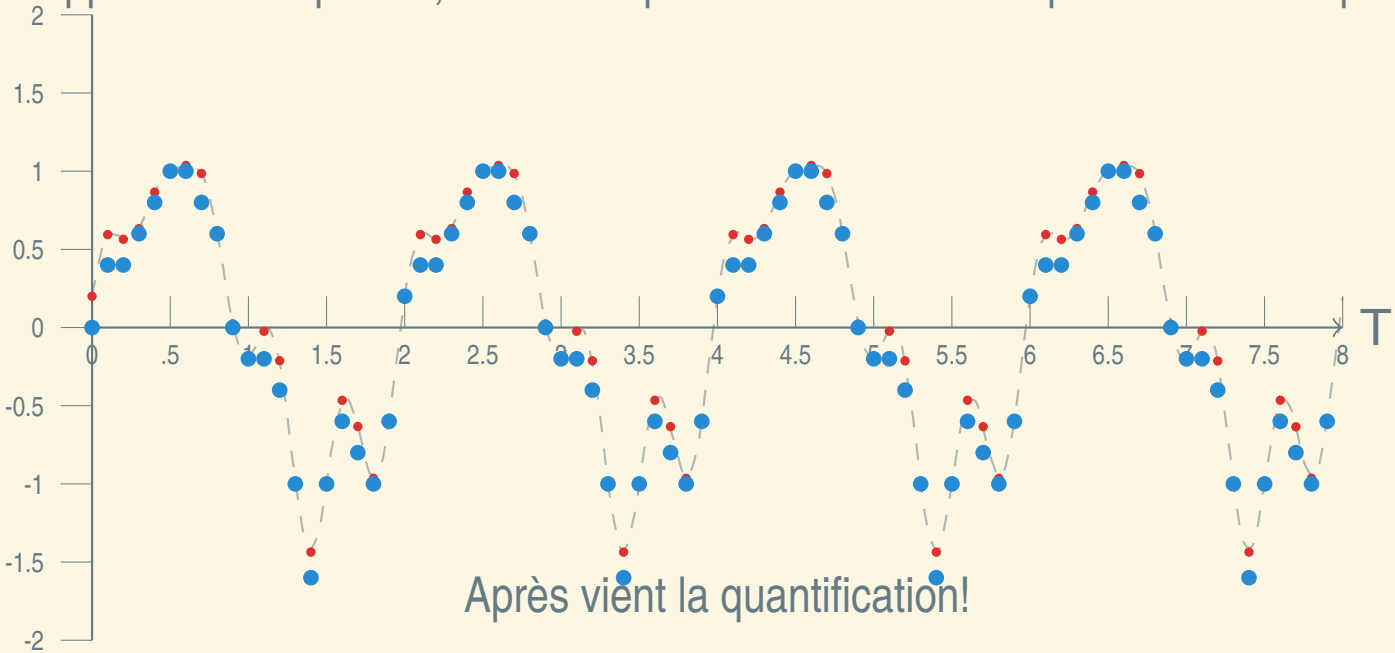
Pour pouvoir reconstruire exactement un signal périodique qui peut être décomposé avec une fréquence maximale, sa fréquence d'échantillonnage f_e doit vérifier *(Théorème d'échantillonnage de Nyquist-Shannon)* :

$$f_e \geqslant 2f_{\mathrm{Max}}$$

L'information discrétisée

Échantillonnage temporel

Rappel : une fréquence, c'est une quantité d'événements par unité de temps ($1\mathrm{Hz} = 1s^{-1}$).



Pour pouvoir reconstruire exactement un signal périodique qui peut être décomposé avec une fréquence maximale, sa fréquence d'échantillonnage f_e doit vérifier *(Théorème d'échantillonnage de Nyquist-Shannon)* :

$$f_e \geqslant 2f_{\mathrm{Max}}$$

L'information quantifiée

Quantification

Cette opération réduit un signal à des *quanta* (singulier *quantum*) en nombre limité. Le nombre de *quanta* possibles s'appelle la *valence*.
La reconstruction exacte du signal n'est plus possible, mais reste souvent proche de l'original.

Exemple

Signal électrique Une tension électrique compris entre 0 (large) et 10 V (strict) peut ainsi être réduit à 10 quanta : 0 V, 1 V,...,9 V.

Le nombre de bits nécessaires pour coder un état du signal peut être exprimé par

$$k = \lceil \log_2 \mathrm{V} \rceil .$$


Beaucoup plus sur la quantification des images plus tard.

Exercices

Signal électrique

Q7 Un signal électrique qui va de 0 à 2,559 V est quantifié sur un quantum de 0,01 V. Quel est le nombre de quanta ? Quelle quantité d'information est transportée par un quantum ?

Q8 Ce signal est périodique, et se décompose avec des fréquences maximales qui vont jusqu'à 10 kHz. Quelle est le débit d'information nécessaire pour reconstituer ce signal à l'identique ?

Q9 Quelle est la taille de l'information nécessaire pour enregistrer ce signal pendant une heure ?

CD audio

Q10 Un CD audio contient de la musique échantillonnée en stéréo sur 16 bits par piste à 44100 Hz (nombre d'échantillons par seconde). Il dure environ 80 minutes. Calculez (de tête) l'ordre de grandeur de la quantité d'information écrite dans un CD audio.

Plan

Plan

Représenter les nombres

- Objet (abstrait) qui admet de nombreuses représentations.
- L'idée de quantité et une représentation visuelle précèdent sans doute l'écriture *(unaire)*.
- Un jeu de règles de représentation des nombres sous forme de signes écrits est un système de numération.

Exemple (Plusieurs représentations)

On représente aussi les nombres sur d'autres *supports* que l'écrit : représentations par sons, par objets (nombre de bougies sur un gâteau). Cela ne change pas le nombre (information), 27 bougies représentent bien 27 éléments (années écoulées, ici) autant que « 2 » collé à « 7 », ou que  (numération babylonienne) ou XXVII (numération romaine).

La numération positionnelle

Définition (Système de numération positionelle)

Un ensemble fini de symboles $\mathcal{B}$ (appelés chiffres) auxquels est associé une valeur entière de 0 à $B-1$, où B est le nombre d'éléments de $\mathcal{B}$. B est la *base*.
La valeur d'une suite finie de k chiffres $\alpha_{k-1}\alpha_{k-2}\ldots\alpha_0$ est la somme :

$$\alpha_{k-1}B^{k-1} + \cdots + \alpha_1 B + \alpha_0 = \sum_{i=0}^{k-1} \alpha_i B^i.$$

- Le mot chiffre vient de l'arabe أَلصِّفر *aṣ-ṣifr* et désignait le zéro.
- Exemple en base 5 : le nombre 132_5 vaut $1 \times 5^2 + 3 \times 5^1 + 2 \times 5^0$, soit $25 + 15 + 2 = 42_{10}$.
- B^i est le *poids* du i-ième chiffre (en comptant de 0 à droite).

Les autres systèmes de numération

un peu de culture générale...

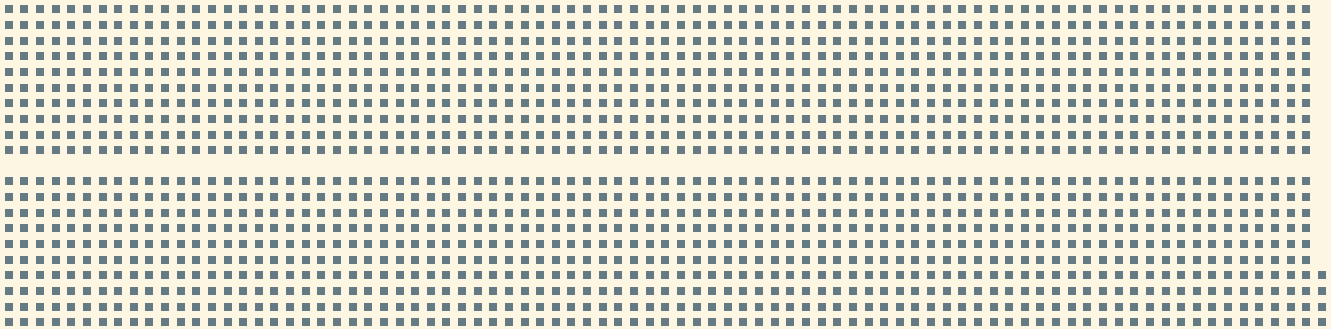
- Systèmes de numération additifs (chiffres grecs, égyptiens) : ∩ ∩ |||||||, par exemple. Chaque poids est représenté par un symbole distinct, la position n'est pas importante. À un détail près, les chiffres romains le sont aussi.
- Systèmes hybrides (numérotation chinoise ou japonaise, français) : on utilise des chiffres fixes, mais on intercale un symbole (écrit) ou un mot (oral) différent pour chaque poids.
- Des systèmes de numération exotiques : les poids ne sont pas 1, B, B^2, B^3, etc. mais les valeurs d'une suite (strictement croissante) : par exemple, numération de Fibonacci.

Cette page est inspirée de Wikipedia *Système de numération*, ainsi que les dessins de chiffres babyloniens.

La base 10

- Système décimal, utilisé depuis le cinquième siècle en Inde, apporté par les Arabes en Europe dans le X^{e} siècle.
- $\mathcal{B} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, et $B = 10$
- Par exemple : mille cinq cent quatre-vingt-quatre se représente par 1684_{10}, qui s'interprète comme

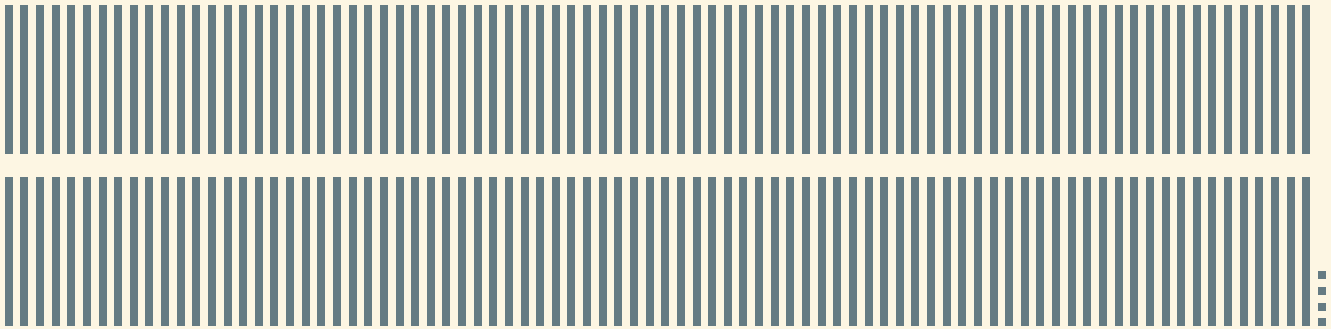
$$1 \times 10^3 + 6 \times 10^2 + 8 \times 10 + 4$$



La base 10

- Système décimal, utilisé depuis le cinquième siècle en Inde, apporté par les Arabes en Europe dans le X^{e} siècle.
- $\mathcal{B} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, et $B = 10$
- Par exemple : mille cinq cent quatre-vingt-quatre se représente par 1684_{10}, qui s'interprète comme

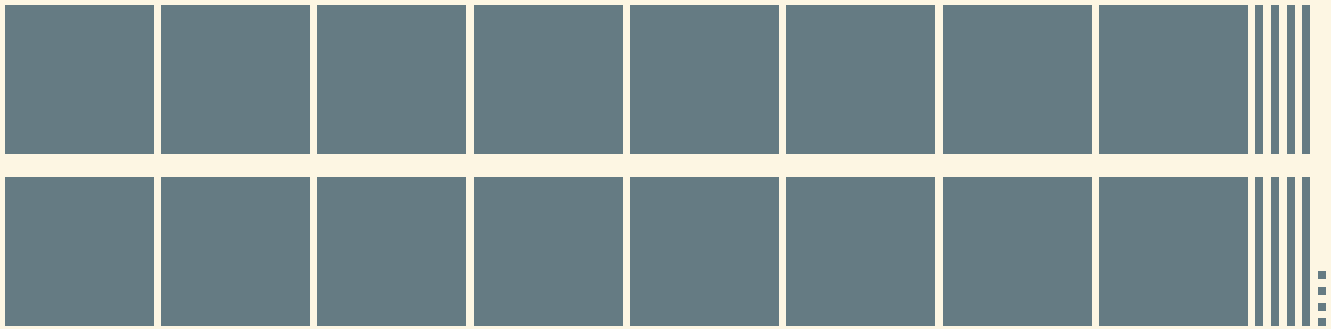
$$1 \times 10^3 + 6 \times 10^2 + 8 \times 10 + 4$$



La base 10

- Système décimal, utilisé depuis le cinquième siècle en Inde, apporté par les Arabes en Europe dans le X^{e} siècle.
- $\mathcal{B} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, et $B = 10$
- Par exemple : mille cinq cent quatre-vingt-quatre se représente par 1684_{10}, qui s'interprète comme

$$1 \times 10^3 + 6 \times 10^2 + 8 \times 10 + 4$$



La base 10

- Système décimal, utilisé depuis le cinquième siècle en Inde, apporté par les Arabes en Europe dans le X^{e} siècle.
- $\mathcal{B} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, et $B = 10$
- Par exemple : mille cinq cent quatre-vingt-quatre se représente par 1684_{10}, qui s'interprète comme

$$1 \times 10^3 + 6 \times 10^2 + 8 \times 10 + 4$$

Représenter les nombres en informatique

Définition (Les bases les plus utilisées sont 2, 8, 10 et 16)

Base	Chiffres		Exemple	Usage
2	{0, 1}	0b	0b10110	Codages bas-niveau
8	{0, 1, 2, 3, 4, 5, 6, 7}	0	026	
16	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}	0x	0x16	
10	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}		22	

- $1 \times 2^4 + 1 \times 2^2 + 1 \times 2 = 22_{10}$
- $2 \times 8 + 6 = 22_{10}$
- $1 \times 16 + 6 = 22_{10}$
- $2 \times 10 + 2 = 22_{10}$
- En binaire, un chiffre est désigné par le terme *bit* (aussi).

Représenter les nombres en informatique

Définition (Les bases les plus utilisées sont 2, 8, 10 et 16)

Base	Chiffres		Exemple	Usage
2	{0, 1}	0b	0b10110	Codages bas-niveau
8	{0, 1, 2, 3, 4, 5, 6, 7}	0	026	peu utilisé
16	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}	0x	0x16	
10	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}		22	

- $1 \times 2^4 + 1 \times 2^2 + 1 \times 2 = 22_{10}$
- $2 \times 8 + 6 = 22_{10}$
- $1 \times 16 + 6 = 22_{10}$
- $2 \times 10 + 2 = 22_{10}$
- En binaire, un chiffre est désigné par le terme *bit* (aussi).

Représenter les nombres en informatique

Définition (Les bases les plus utilisées sont 2, 8, 10 et 16)

Base	Chiffres		Exemple	Usage
2	{0, 1}	0b	0b10110	Codages bas-niveau
8	{0, 1, 2, 3, 4, 5, 6, 7}	0	026	peu utilisé
16	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}	0x	0x16	Écriture compacte d'octets
10	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}		22	

- $1 \times 2^4 + 1 \times 2^2 + 1 \times 2 = 22_{10}$
- $2 \times 8 + 6 = 22_{10}$
- $1 \times 16 + 6 = 22_{10}$
- $2 \times 10 + 2 = 22_{10}$
- En binaire, un chiffre est désigné par le terme *bit* (aussi).

Représenter les nombres en informatique

Définition (Les bases les plus utilisées sont 2, 8, 10 et 16)

Base	Chiffres		Exemple	Usage
2	{0, 1}	0b	0b10110	Codages bas-niveau
8	{0, 1, 2, 3, 4, 5, 6, 7}	0	026	peu utilisé
16	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}	0x	0x16	Écriture compacte d'octets
10	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}		22	Nombres courants

- $1 \times 2^4 + 1 \times 2^2 + 1 \times 2 = 22_{10}$
- $2 \times 8 + 6 = 22_{10}$
- $1 \times 16 + 6 = 22_{10}$
- $2 \times 10 + 2 = 22_{10}$
- En binaire, un chiffre est désigné par le terme *bit* (aussi).

Représenter les nombres en informatique

Définition (Les bases les plus utilisées sont 2, 8, 10 et 16)

Base	Chiffres		Exemple	Usage
2	{0, 1}	0b	0b10110	Codages bas-niveau
8	{0, 1, 2, 3, 4, 5, 6, 7}	0	026	peu utilisé
16	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}	0x	0x16	Écriture compacte d'octets
10	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}		22	Nombres courants

- $1 \times 2^4 + 1 \times 2^2 + 1 \times 2 = 22_{10}$
- $2 \times 8 + 6 = 22_{10}$
- $1 \times 16 + 6 = 22_{10}$
- $2 \times 10 + 2 = 22_{10}$
- En binaire, un chiffre est désigné par le terme *bit* (aussi).

De la base x à la base 10

On peut toujours convertir un nombre de la façon suivante.

Méthode (recalcul)

Si en base x, il s'écrit $\alpha\beta\gamma\delta$, il vaut (par définition) :

$$\alpha \times x^3 + \beta \times x^2 + \gamma \times x + \delta$$

Exemple (conversion de 0x4D7)

Le nombre 0x4D7 (hexadécimal) est égal à $4 \times 16^2 + D \times 16 + 7$, donc à $4 \times 256 + 13 \times 16 + 7 = 1239$ en base 10.

Exemple (puissance de la base)

B^n s'écrit toujours 1 suivi de n zéros (par exemple, 2^6 s'écrit `0b1 000 000`)

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ; $1/2 = 0$, reste 1 ;

$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 = 1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...

216

$216_{10} = 216_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ; $1/2 = 0$, reste 1 ;

$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 = 1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...

216

$216_{10} = 216_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ; $1/2 = 0$, reste 1 ;

$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 = 1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...

216

$216_{10} = 216_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ; $1/2 = 0$, reste 1 ;

$13 = 6 \times 2 + \texttt{0b1} = 3 \times 2 \times 2 + \texttt{0b01} = 1 \times 2 \times 2 \times 2 + \texttt{0b101} = \texttt{0b1101}$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...

216

$216_{10} = 216_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 =$
$1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
216
$216_{10} = 216_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + \texttt{0b1} = 3 \times 2 \times 2 + \texttt{0b01} =$
$1 \times 2 \times 2 \times 2 + \texttt{0b101} = \texttt{0b1101}$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
216
$216_{10} = 216_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + \texttt{0b1} = 3 \times 2 \times 2 + \texttt{0b01} =$
$1 \times 2 \times 2 \times 2 + \texttt{0b101} = \texttt{0b1101}$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
216
$216_{10} = 216_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + \texttt{0b1} = 3 \times 2 \times 2 + \texttt{0b01} =$
$1 \times 2 \times 2 \times 2 + \texttt{0b101} = \texttt{0b1101}$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
216
$216_{10} = 216_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + \texttt{0b1} = 3 \times 2 \times 2 + \texttt{0b01} =$
$1 \times 2 \times 2 \times 2 + \texttt{0b101} = \texttt{0b1101}$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
216
$216_{10} = 216_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 =$
$1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
216
$216_{10} = 216_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 =$
$1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...

$216 - 128 = 88$

$216_{10} = 0b\underbrace{1}_{128}0000000 + 88_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 =$
$1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
$216 - 128 = 88 - 64 = 24$
$216_{10} = 0b1\underbrace{\ 1\ }_{64}000000 + 24_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 =$
$1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
$216 - 128 = 88 - 64 = 24$
$216_{10} = 0b11\underbrace{0}_{32}00000 + 24_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 =$
$1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
$216 - 128 = 88 - 64 = 24 - 16 = 8$
$216_{10} = 0b110\underbrace{1}_{16}0000 + 8_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 =$
$1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
$216 - 128 = 88 - 64 = 24 - 16 = 8 - 8 = 0$
$216_{10} = 0b1101\underbrace{\_1\_}_{8}000 + 0_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 =$
$1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
$216 - 128 = 88 - 64 = 24 - 16 = 8 - 8 = 0$
$216_{10} = 0b11011\underbrace{0}_{4}00 + 0_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 =$
$1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
$216 - 128 = 88 - 64 = 24 - 16 = 8 - 8 = 0$
$216_{10} = 0b110110\underbrace{0}_{2}0 + 0_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 =$
$1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
$216 - 128 = 88 - 64 = 24 - 16 = 8 - 8 = 0$
$216_{10} = 0b1101100\underbrace{0}_{1} + 0_{10}$

De la base 10 à la base 2

Méthode (divisions successives)

On divise le nombre par la base (2). Le reste est le dernier chiffre du nombre dans la base 2, on recommence avec le résultat de la division.

Ceci fonctionne avec toutes les bases, diviser par B *au lieu de 2.*

Exemple (divisions successives)

$13/2 = 6$, reste 1 ; $6/2 = 3$, reste 0 ; $3/2 = 1$, reste 1 ;
$1/2 = 0$, reste 1 ;
$13 = 6 \times 2 + 0b1 = 3 \times 2 \times 2 + 0b01 =$
$1 \times 2 \times 2 \times 2 + 0b101 = 0b1101$

Méthode (soustractions successives, rapide)

Puissances de 2 : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...
$216 - 128 = 88 - 64 = 24 - 16 = 8 - 8 = 0$
$216_{10} = 0b11011000$

De la base 2 à 8 et 16 (et inversement)

- Base 2 vers 8 ou 16 ou inverse : substitution mécanique !
- Compléter par des 0 devant si nécessaire (octal : 3 chiffres, hexadécimal : 4) ;
- Connaître les correspondances pour chaque chiffre ;

Hex./Octal	0	1	2	3	4	5	6	7
Binaire	0	1	10	11	100	101	110	111

Hex.	8	9	A	B	C	D	E	F
Binaire	1000	1001	1010	1011	1100	1101	1110	1111

- Base $8 = 2^3$: `0b11101`
- Base $16 = 2^4$: `0b11101`
- De 16 ou 8, vers 2, procédure inverse : `0x3A` = `0b00111010`
- **APPRENEZ CES TABLES PAR CŒUR !**

De la base 2 à 8 et 16 (et inversement)

- Base 2 vers 8 ou 16 ou inverse : substitution mécanique !
- Compléter par des 0 devant si nécessaire (octal : 3 chiffres, hexadécimal : 4) ;
- Connaître les correspondances pour chaque chiffre ;

Hex./Octal	0	1	2	3	4	5	6	7
Binaire	0	1	10	11	100	101	110	111

Hex.	8	9	A	B	C	D	E	F
Binaire	1000	1001	1010	1011	1100	1101	1110	1111

- Base $8 = 2^3$: `0b011101`
- Base $16 = 2^4$: `0b00011101`
- De 16 ou 8, vers 2, procédure inverse : `0x3A` = `0b00111010`
- **Apprenez ces tables par cœur !**

De la base 2 à 8 et 16 (et inversement)

- Base 2 vers 8 ou 16 ou inverse : substitution mécanique !
- Compléter par des 0 devant si nécessaire (octal : 3 chiffres, hexadécimal : 4) ;
- Connaître les correspondances pour chaque chiffre ;

Hex./Octal	0	1	2	3	4	5	6	7
Binaire	0	1	10	11	100	101	110	111

Hex.	8	9	A	B	C	D	E	F
Binaire	1000	1001	1010	1011	1100	1101	1110	1111

- Base $8 = 2^3$: $0b\underbrace{011}_{3}\underbrace{101}_{5}$
- Base $16 = 2^4$: $0b\underbrace{0001}_{1}\underbrace{1101}_{D}$
- De 16 ou 8, vers 2, procédure inverse : $0x3A = 0b00111010$
- **APPRENEZ CES TABLES PAR CŒUR !**

De la base 2 à 8 et 16 (et inversement)

- Base 2 vers 8 ou 16 ou inverse : substitution mécanique !
- Compléter par des 0 devant si nécessaire (octal : 3 chiffres, hexadécimal : 4) ;
- Connaître les correspondances pour chaque chiffre ;

Hex./Octal	0	1	2	3	4	5	6	7
Binaire	0	1	10	11	100	101	110	111

Hex.	8	9	A	B	C	D	E	F
Binaire	1000	1001	1010	1011	1100	1101	1110	1111

- Base $8 = 2^3$: $0b\underbrace{011}_{3}\underbrace{101}_{5} = 035$
- Base $16 = 2^4$: $0b\underbrace{0001}_{1}\underbrace{1101}_{D} = 0x1D$
- De 16 ou 8, vers 2, procédure inverse : $0x3A = 0b00111010$
- **APPRENEZ CES TABLES PAR CŒUR !**

De la base 2 à 8 et 16 (et inversement)

- Base 2 vers 8 ou 16 ou inverse : substitution mécanique !
- Compléter par des 0 devant si nécessaire (octal : 3 chiffres, hexadécimal : 4) ;
- Connaître les correspondances pour chaque chiffre ;

Hex./Octal	0	1	2	3	4	5	6	7
Binaire	0	1	10	11	100	101	110	111

Hex.	8	9	A	B	C	D	E	F
Binaire	1000	1001	1010	1011	1100	1101	1110	1111

- Base $8 = 2^3$: $\texttt{0b}\underbrace{\texttt{011}}_{3}\underbrace{\texttt{101}}_{5} = \texttt{035}$
- Base $16 = 2^4$: $\texttt{0b}\underbrace{\texttt{0001}}_{1}\underbrace{\texttt{1101}}_{\texttt{D}} = \texttt{0x1D}$
- De 16 ou 8, vers 2, procédure inverse : $\texttt{0x3A} = \texttt{0b00111010}$
- **Apprenez ces tables par cœur !**

De la base 2 à 8 et 16 (et inversement)

- Base 2 vers 8 ou 16 ou inverse : substitution mécanique !
- Compléter par des 0 devant si nécessaire (octal : 3 chiffres, hexadécimal : 4) ;
- Connaître les correspondances pour chaque chiffre ;

Hex./Octal	0	1	2	3	4	5	6	7
Binaire	0	1	10	11	100	101	110	111

Hex.	8	9	A	B	C	D	E	F
Binaire	1000	1001	1010	1011	1100	1101	1110	1111

- Base $8 = 2^3$: $0b\underbrace{011}_{3}\underbrace{101}_{5} = 035$
- Base $16 = 2^4$: $0b\underbrace{0001}_{1}\underbrace{1101}_{D} = 0x1D$
- De 16 ou 8, vers 2, procédure inverse : $0x3A = 0b00111010$
- **Apprenez ces tables par cœur !**

Exercices

Puissances de 2

Q11 Écrivez la liste de toutes les puissances de 2, de 2^{-4} à 2^{16}.

Q12 Écrivez une table de conversion des chiffres hexadécimaux et octaux vers le codage naturel écrit en binaire (4 bits ou 3 bits).

Conversions

Q13 Écrivez en binaire et en hexadécimal les nombres décimaux suivants : 28 ; 149 ; 1285.

Q14 Convertissez en décimal les nombres suivants : 0x48 ; 0xA1C ; 0b1010010010011111.

Q15 Comment trouver midi à quatorze heures ?

Plan

Bases, entiers relatifs et réels

- Pour les entiers relatifs, il faut une information supplémentaire : le signe ;
- Représentation classique : un signe – pour les négatifs.
- Réels : la virgule (séparateur décimal) est à droite du chiffre de poids 1 (exposant 0) *(représentation en virgule fixe)*.

Exemple

$$\begin{aligned} -10,11_2 &= -\left(1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-2}\right) \\ &= -2,75_{10} \\ -47,2_8 &= -\left(4 \times 8^1 + 7 \times 8^0 + 2 \times 8^{-1}\right) \\ &= -39,25_{10} \end{aligned}$$

Convertir un réel d'une base dans une autre

Théorème

On peut toujours convertir d'un côté la partie fractionnaire d'un nombre, de l'autre sa partie entière.

Méthode (multiplications successives)

On multiplie par B *la partie fractionnaire, la partie entière du résultat donne le premier chiffre après la virgule.* $0,375 \times 2 = 0,75$; $0,75 \times 2 = 1,5$; $0,5 \times 2 = 1$;...*développement fini ! Pas toujours !*

$0,375 =$

Méthode (soustractions successives, rapide)

Puissances de 2 : 0,5, 0,25, 0,125, 0,0625,...

$0,8125$

$0,8125_{10} =$

Convertir un réel d'une base dans une autre

Théorème

On peut toujours convertir d'un côté la partie fractionnaire d'un nombre, de l'autre sa partie entière.

Méthode (multiplications successives)

On multiplie par B *la partie fractionnaire, la partie entière du résultat donne le premier chiffre après la virgule.* $0,375 \times 2 = 0,75$; $0,75 \times 2 = 1,5$; $0,5 \times 2 = 1$;*...développement fini ! Pas toujours !*

$0,375 =$

Méthode (soustractions successives, rapide)

Puissances de 2 : 0,5, 0,25, 0,125, 0,0625,...

$0,8125$

$0,8125_{10} =$

Convertir un réel d'une base dans une autre

Théorème

On peut toujours convertir d'un côté la partie fractionnaire d'un nombre, de l'autre sa partie entière.

Méthode (multiplications successives)

On multiplie par B *la partie fractionnaire, la partie entière du résultat donne le premier chiffre après la virgule.* $0,375 \times 2 = 0,75$; $0,75 \times 2 = 1,5$; $0,5 \times 2 = 1$;*...développement fini ! Pas toujours !*

$0,375 = 0b0,0$

Méthode (soustractions successives, rapide)

Puissances de 2 : 0,5, 0,25, 0,125, 0,0625,...

$0,8125$

$0,8125_{10} =$

Convertir un réel d'une base dans une autre

Théorème

On peut toujours convertir d'un côté la partie fractionnaire d'un nombre, de l'autre sa partie entière.

Méthode (multiplications successives)

On multiplie par B *la partie fractionnaire, la partie entière du résultat donne le premier chiffre après la virgule.* $0,375 \times 2 = 0,75$; $0,75 \times 2 = 1,5$; $0,5 \times 2 = 1$;*...développement fini ! Pas toujours !*

$0,375 = 0b0,01$

Méthode (soustractions successives, rapide)

Puissances de 2 : 0,5, 0,25, 0,125, 0,0625,...

$0,8125$

$0,8125_{10} =$

Convertir un réel d'une base dans une autre

Théorème

On peut toujours convertir d'un côté la partie fractionnaire d'un nombre, de l'autre sa partie entière.

Méthode (multiplications successives)

On multiplie par B *la partie fractionnaire, la partie entière du résultat donne le premier chiffre après la virgule.* $0,375 \times 2 = 0,75$; $0,75 \times 2 = 1,5$; $0,5 \times 2 = 1$; *...développement fini ! Pas toujours !*

$0,375 = 0b0,011$

Méthode (soustractions successives, rapide)

Puissances de 2 : 0,5, 0,25, 0,125, 0,0625,...

$0,8125$

$0,8125_{10} =$

Convertir un réel d'une base dans une autre

Théorème

On peut toujours convertir d'un côté la partie fractionnaire d'un nombre, de l'autre sa partie entière.

Méthode (multiplications successives)

On multiplie par B *la partie fractionnaire, la partie entière du résultat donne le premier chiffre après la virgule.* $0,375 \times 2 = 0,75$; $0,75 \times 2 = 1,5$; $0,5 \times 2 = 1$;*...développement fini ! Pas toujours !*

$0,375 = 0b0,011$

Méthode (soustractions successives, rapide)

Puissances de 2 : 0,5, 0,25, 0,125, 0,0625,...

$0,8125$

$0,8125_{10} =$

Convertir un réel d'une base dans une autre

Théorème

On peut toujours convertir d'un côté la partie fractionnaire d'un nombre, de l'autre sa partie entière.

Méthode (multiplications successives)

On multiplie par B *la partie fractionnaire, la partie entière du résultat donne le premier chiffre après la virgule.* $0,375 \times 2 = 0,75$; $0,75 \times 2 = 1,5$; $0,5 \times 2 = 1$;*...développement fini ! Pas toujours !*

$0,375 = 0b0,011$

Méthode (soustractions successives, rapide)

Puissances de 2 : 0,5, 0,25, 0,125, 0,0625,...

$0,8125$

$0,8125_{10} =$

Convertir un réel d'une base dans une autre

Théorème

On peut toujours convertir d'un côté la partie fractionnaire d'un nombre, de l'autre sa partie entière.

Méthode (multiplications successives)

On multiplie par B *la partie fractionnaire, la partie entière du résultat donne le premier chiffre après la virgule.* $0,375 \times 2 = 0,75$; $0,75 \times 2 = 1,5$; $0,5 \times 2 = 1$;*...développement fini ! Pas toujours !*

$0,375 = 0b0,011$

Méthode (soustractions successives, rapide)

Puissances de 2 : 0,5, 0,25, 0,125, 0,0625,...

$0,8125 - 0,5 = 0,3125$

$0,8125_{10} = 0b0,\underbrace{1}_{1/2} + 0,3125_{10}$

Convertir un réel d'une base dans une autre

Théorème

On peut toujours convertir d'un côté la partie fractionnaire d'un nombre, de l'autre sa partie entière.

Méthode (multiplications successives)

On multiplie par B *la partie fractionnaire, la partie entière du résultat donne le premier chiffre après la virgule.* $0,375 \times 2 = 0,75$; $0,75 \times 2 = 1,5$; $0,5 \times 2 = 1$;*...développement fini ! Pas toujours !*

$0,375 = 0b0,011$

Méthode (soustractions successives, rapide)

Puissances de 2 : 0,5, 0,25, 0,125, 0,0625,...

$0,8125 - 0,5 = 0,3125 - 0,25 = 0,0625$

$0,8125_{10} = 0b0,1\underbrace{1}_{1/4} + 0,0625_{10}$

Convertir un réel d'une base dans une autre

Théorème

On peut toujours convertir d'un côté la partie fractionnaire d'un nombre, de l'autre sa partie entière.

Méthode (multiplications successives)

On multiplie par B *la partie fractionnaire, la partie entière du résultat donne le premier chiffre après la virgule.* $0,375 \times 2 = 0,75$; $0,75 \times 2 = 1,5$; $0,5 \times 2 = 1$;*...développement fini ! Pas toujours !*

$0,375 = 0b0,011$

Méthode (soustractions successives, rapide)

Puissances de 2 : 0,5, 0,25, 0,125, 0,0625,...

$0,8125 - 0,5 = 0,3125 - 0,25 = 0,0625$

$0,8125_{10} = 0b0,11\underbrace{0}_{1/8} + 0,0625_{10}$

Convertir un réel d'une base dans une autre

Théorème

On peut toujours convertir d'un côté la partie fractionnaire d'un nombre, de l'autre sa partie entière.

Méthode (multiplications successives)

On multiplie par B *la partie fractionnaire, la partie entière du résultat donne le premier chiffre après la virgule.* $0,375 \times 2 = 0,75$; $0,75 \times 2 = 1,5$; $0,5 \times 2 = 1$;*...développement fini ! Pas toujours !*

$0,375 = 0b0,011$

Méthode (soustractions successives, rapide)

Puissances de 2 : 0,5, 0,25, 0,125, 0,0625,...

$0,8125 - 0,5 = 0,3125 - 0,25 = 0,0625 - 0,0625 = 0$

$0,8125_{10} = 0b0,110\underbrace{1}_{1/16} + 0_{10}$

Convertir un réel d'une base dans une autre

Théorème

On peut toujours convertir d'un côté la partie fractionnaire d'un nombre, de l'autre sa partie entière.

Méthode (multiplications successives)

On multiplie par B *la partie fractionnaire, la partie entière du résultat donne le premier chiffre après la virgule.* $0,375 \times 2 = 0,75$; $0,75 \times 2 = 1,5$; $0,5 \times 2 = 1$;*...développement fini ! Pas toujours !*

$0,375 = 0b0,011$

Méthode (soustractions successives, rapide)

Puissances de 2 : 0,5, 0,25, 0,125, 0,0625,...

$0,8125 - 0,5 = 0,3125 - 0,25 = 0,0625 - 0,0625 = 0$

$0,8125_{10} = 0b0,1101$

Exercices

Changements de base

Q16 Écrivez en binaire et en hexadécimal les nombres décimaux suivants : 0,3125 ; 164,3125.

Q17 Convertissez en décimal le nombre suivant : 0b1010,0011.

Plan

Le codage

- Plutôt que d'écrire des nombres, on est souvent amené à les *coder*, c'est-à-dire à les écrire sur une taille fixe.
- On écrit ces codes en binaire (ou en hexadécimal pour gagner de la place) avec un nombre déterminé à l'avance de bits.
- Avec un nombre fixé k de bits, on peut écrire uniquement un nombre fixé de nombres (2^k).

Définition (Codage naturel des entiers — NAT)

Le codage naturel consiste à écrire l'entier en base 2 et à compléter l'écriture par des 0 à gauche jusqu'à atteindre la taille désirée. Exemple : 27_{10}=0b11011 se code 0001 1011 en NAT 8 bits.
Avec n bits, on code les entiers de 0 à $2^n - 1$. Usuellement, on utilise des tailles multiples de 8.

Le codage des entiers relatifs (1)

VA+S et C1, peu usités

Définition (Codage valeur absolue+signe — VA+S)

On écrit l'entier en base 2 et on complète l'écriture par des 0 à gauche jusqu'à atteindre la taille désirée *moins* 1, et de coder le signe devant par 0 (positif) ou 1 (négatif).
Exemple : -12_{10}=0b1100 se code 1000 1100 en VA+S 8 bits.

Définition (Codage complément à 1 — C1)

L'entier écrit en base 2 est complété par des 0 à gauche jusqu'à la taille désirée. *Si le nombre est négatif,* on **complémente** alors chacun des chiffres ($0 \longleftrightarrow 1$).
Exemple : -12_{10}=0b1100 se code 1111 0011 en C1 8 bits.

Avec n bits, on code les entiers de $-2^{n-1}+1$ à $2^{n-1}-1$ (pour VA+S et C1).

Le codage des entiers relatifs (2)

C2, le plus utilisé

Définition (Codage complément à 2 — C2)

Si le nombre est positif, on complète son écriture binaire par des 0 à gauche jusqu'à la taille désirée. *Si le nombre est négatif,* on écrit *sa valeur absolue moins* 1 en base 2, on le complète par des 0 à gauche jusqu'à la taille désirée, et on **complémente** alors chacun des chiffres ($0 \longleftrightarrow 1$).
Exemple : -12_{10}=0b1100 se code 1111 0100 en C2 8 bits.

Le codage des entiers relatifs (3)

C2 (deuxième étape)

- Avec n bits, on code les entiers de -2^{n-1} à $2^{n-1}-1$.
- Dans l'autre sens (de codage C2 vers valeur binaire), il faut faire les opérations dans l'ordre inverse.
- Pour les positifs, les quatre codages sont identiques.

Exemple (Codage C2 sur 8 bits)

$28 = 0b1\,1100$ donc codage $001\,1100 \rightarrow 0001\,1100$
$-28 = 0b1\,1100$ donc codage $001\,1011 \rightarrow 110\,0100 \rightarrow 1110\,0100$
Codage C2 : 10100111, donc valeur négative : $0100111 \rightarrow 1011000 \rightarrow 1011001$, soit 89 en décimal ; donc -89.
Codage C2 : 00100111, donc valeur positive : $0b100111$, soit 39 en décimal ; donc 39.

Exercices

Codage d'entiers

Q18 Ce tableau comporte des cases inutilisées. Complétez-le :

Décimal	**Écriture Binaire**	**Type de codage**	**Codage** (binaire)	Codage (hexa)
−18	−10010	VA+S (8 bits)	10010010	0x92
424		NAT (16 bits)		
−138		C2 (16 bits)		
	−1110011	C1 (8 bits)		
−4197		VA+S (24 bits)		
−84				0xAB
341		NAT (8 bits)		

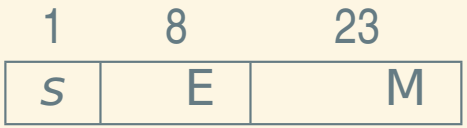
Plan

Représentation en virgule flottante

- Décomposition en quatre parties d'un réel : signe s, valeur v, base B et exposant e ;
- Ex : $-325 = -3,25 \times 10^2, -0b101,1 = -0b1,011 \times 2^2$;
- Contrainte : valeur=réel x, tq $1 \leq x < \mathrm{B}$;
- Un seul chiffre avant la virgule !
- e entier, signe usuel ;
- $x = (-1)^s \times v \times \mathrm{B}^e$
- Choix de B donne une décomposition unique si $x \neq 0$;
- En binaire, le premier chiffre de v est forcément 1 ;
- Exception pour 0.

Normalisation IEEE 754

- Réduire la quantité d'info redondante ;
- format 32 bits pour simple précision, 64 et 80 pour double et étendue à partir de codages simples côte-à-côte ;
- Stockage de s, $\mathrm{E} = e + 127$, M (partie fractionnaire de v) ;
- Tout ceci en codage NAT car tout positif !
- Format sur 32 bits :



;

- Exception : pour 0, $\mathrm{E} = 00000000$, pour ∞, $\mathrm{E} = 11111111$;
- Intervalle de valeurs (32 bits) : 2^{-126} à $(2 - 2^{-23}) \times 2^{127}$, soit de $1,8 \times 10^{-38}$ à $3,4 \times 10^{38}$.

Exercices

Codage IEEE754

Q19 Ce tableau comporte des cases inutilisées. Complétez-le :

Décimal	Binaire	Virgule flottante	E	Codage IEEE754			Hexa
				S	E (8b)	M (23b)	
19,5	10011,1	$1{,}00111 \times 2^4$	131	0	10000011	00111 $\underbrace{0\ldots0}_{18\text{ fois}}$	419C0000
-7,5							
-46,25							
0,3125							
							BE400000
							7F800000
0							
-26,375 $\times 2^{40}$							

Plan

Plan

Exercices

De la fonction à l'algorithme

La numération grecque (simple) est proche de la numération romaine que vous connaissez : on note les nombres comme suit :

1	5	10	50	100	500	1 000	5 000	10 000	50 000
I	Γ	Δ	$\Gamma\Delta$	H	ΓH	X	ΓX	M	ΓM

C'est à la différence près que l'on a pas de règle soustractive : le nombre 4 s'écrit IIII, pas IΓ. La position des chiffres n'a théoriquement aucune importance, mais on les classait dans l'ordre décroissant de valeur.

Q20 Ce système est-il un système de numération positionnelle ?

Q21 Écrivez votre âge et votre date de naissance en numération grecque.

Q22 Écrivez un algorithme d'addition des nombres représentés en numération grecque. Est-ce que cet algorithme est le même qu'en décimal ?

Q23 Faites l'addition de votre âge et de votre année de naissance avec votre algorithme (vous devriez obtenir XXΔIIII ou XXΔIII). De quelle représentations partez-vous ?

Q24 Faites la même chose en décimal. De quelles représentations partez-vous ? Est-ce que l'algorithme est le même ? Est-ce que la fonction calculée est la même ?

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + Br$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

$$\begin{array}{cccccc} & & & 1 & 0 & 0 \\ + & 1 & 0 & 1 & 1 & 0 \\ + & & & 1 & 1 & 1 \\ \hline \end{array}$$

On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

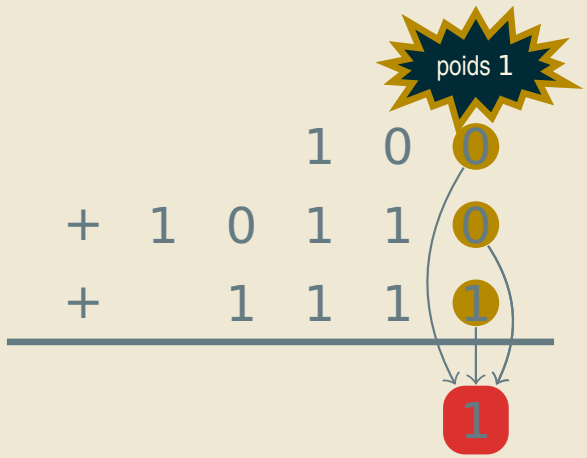
Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + Br$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)



On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

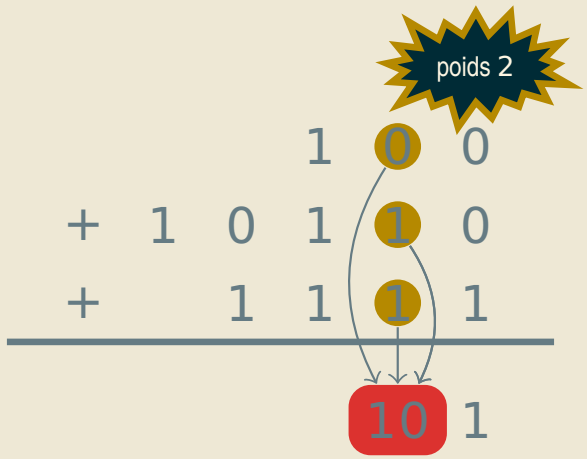
Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + Br$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)



On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

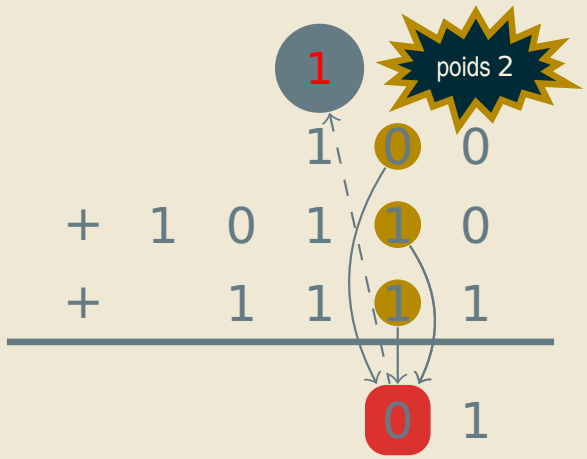
Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + Br$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)



On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

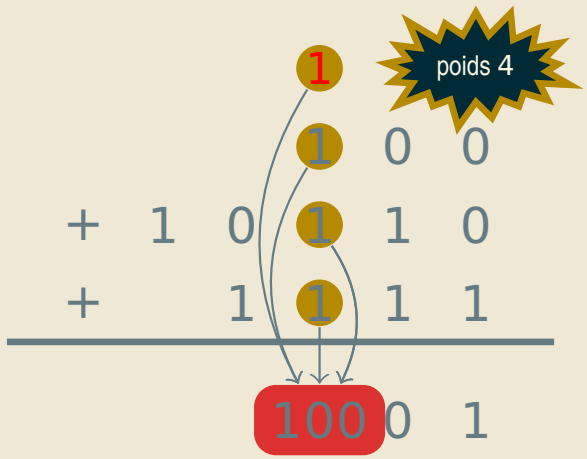
Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + Br$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)



On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

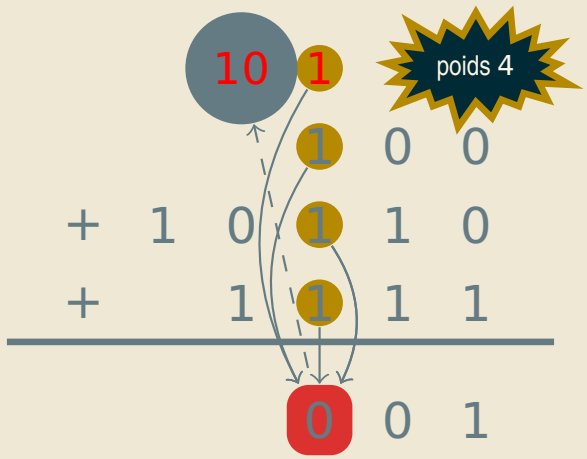
Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + Br$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)



On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

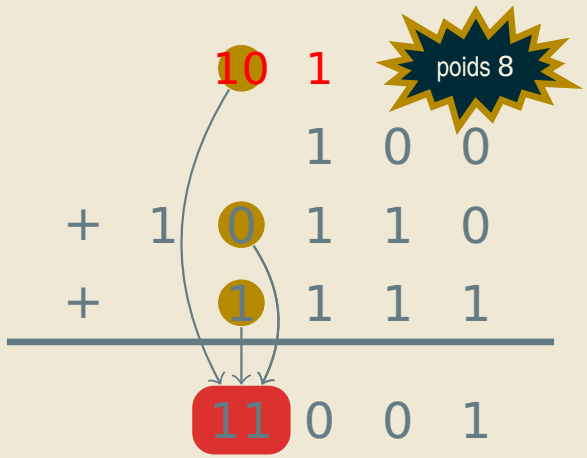
Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + Br$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)



On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

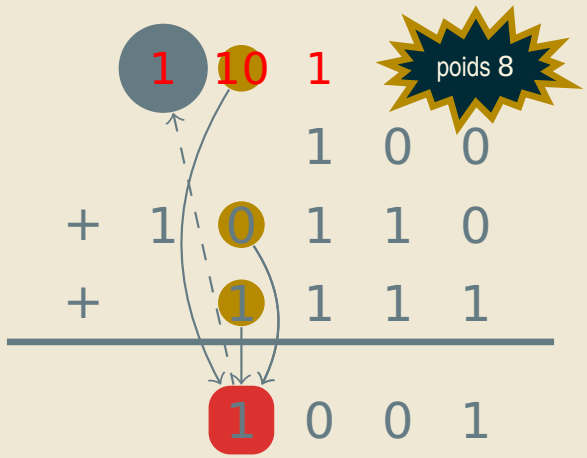
Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + Br$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)



On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

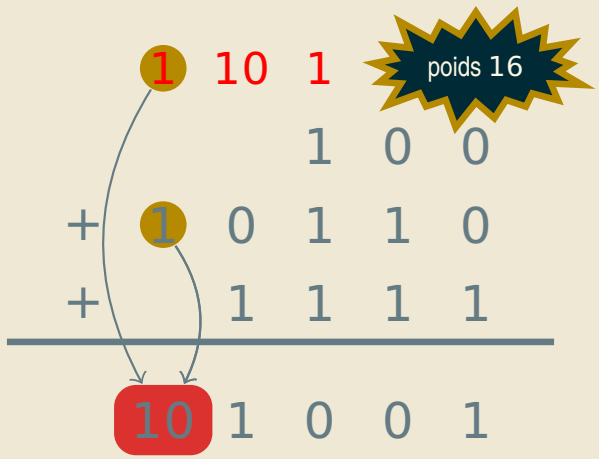
Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + Br$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)



On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

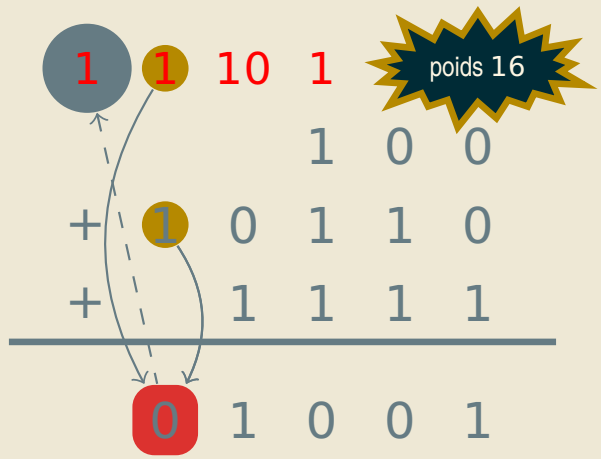
Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + Br$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)



On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + Br$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + Br$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

	1	1	10	1		
				1	0	0
+	1	0	1	1	0	
+			1	1	1	1
	1	0	1	0	0	1

On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Multiplication

Méthode (Multiplication)

À l'instar de l'addition, la multiplication se fait comme pour les nombres décimaux. Les tables sont justes différentes (il faut les écrire dans la bonne base !).
En binaire, c'est très facile : on multiplie par 0 *ou par* 1*, donc on se contente d'additionner des copies du multiplicande décalées là où le multiplicateur a des* 1*.*
Très important : *décaler à gauche de* n *colonnes un nombre, c'est le multiplier par* B^n*. Le décaler à droite, c'est le diviser par* B^n*.*

			1	1	0
×			1	0	1
			1	1	0
(+				0	)
+	1	1	0		
	1	1	1	1	0

Négatifs et réels

Par analogie avec les techniques maîtrisées en base 10, il est possible de faire également :

- Des soustractions : lorsque le chiffre duquel on soustrait n'est pas suffisant, on ajoute 1 au chiffre à soustraire dans la colonne d'ordre immédiatement supérieur, et en compensation, on ajoute la base dans la colonne courante.
- Des divisions : en fait, on fait plein de multiplications enchaînées avec des soustractions.
- Additionner un négatif, c'est soustraire un positif.
- Des opérations en virgule fixe : les règles de placement de la virgule sont les mêmes qu'en base 10.
- Des décalages qui sont des multiplications ou divisions par une puissance de la base.
- Pour la virgule flottante, les multiplications sont simples ; les additions nécessitent de recoder en virgule fixe.

Exercices

Calcul en binaire et hexadécimal

Q25 Faites les additions en binaire : `0b11010101+0b11100101` ; `0b1,1+0b110+0b100,1+0b111,1+0b1010,1+0b100,1`.

Q26 Faites les opérations suivantes en hexadécimal : $0x122 + 0x233$; $0x87 + 0x54$; $0x18 + 0x9$; $0xED + 0xED$; $0x100 - 0x3$.

Q27 Faites la multiplication suivante : 17×129 à la fois en décimal et en binaire.

Q28 Faites la multiplication suivante en binaire : 110110×1101.

Plan

Les algorithmes d'addition

- L'addition de codage se fait par un algorithme (représentation/information). Si l'algorithme fait ce qu'on attend de lui (identique à la fonction), il est *correct*.
- Tous les nombres ne sont pas représentables. Si le résultat de l'addition (fonction) donne un nombre non représentable, l'algorithme ne peut pas être correct.
- L'algorithme d'addition pour NAT et C2 est très similaire à la méthode usuelle (décrite avant) ; on coupe juste le résultat à la taille du codage.

Théorème

En C2 et NAT, si le résultat est représentable, l'addition (classique) est correcte.

Addition correcte ou pas ?

Exemple (Addition en NAT)

$$131 + 85 \xrightarrow{\text{code}} 1000\,0011_{\text{NAT}} + 0101\,0101_{\text{NAT}}$$

$$\overset{\text{algo}}{=}$$

$$216 \xleftarrow{\text{decode}} 1101\,1000_{\text{NAT}}$$

$$187 + 101 \xrightarrow{\text{code}} 1011\,1011_{\text{NAT}} + 0110\,0101_{\text{NAT}}$$

$$\overset{\text{algo}}{=}$$

$$32 (\neq 288) \xleftarrow{\text{decode}} 0010\,0000_{\text{NAT}}$$

- En C1 et en VA+S, il suffit de mettre un négatif pour opération incorrecte
- En plus, problème du double zéro pour C1 et VA+S.
- C2 : comme si le bit de poids fort était de poids -2^{n-1} au lieu de 2^{n-1}, arithmétique modulo 2^n.

Addition correcte ou pas ?

Exemple (Addition en NAT)

$$131 + 85 \xrightarrow{\text{code}} 1000\,0011_{\text{NAT}} + 0101\,0101_{\text{NAT}}$$

$$\overset{\text{algo}}{=}$$

Correct !

$$216 \xleftarrow{\text{decode}} 1101\,1000_{\text{NAT}}$$

$$187 + 101 \xrightarrow{\text{code}} 1011\,1011_{\text{NAT}} + 0110\,0101_{\text{NAT}}$$

$$\overset{\text{algo}}{=}$$

$$32 (\neq 288) \xleftarrow{\text{decode}} 0010\,0000_{\text{NAT}}$$

- En C1 et en VA+S, il suffit de mettre un négatif pour opération incorrecte
- En plus, problème du double zéro pour C1 et VA+S.
- C2 : comme si le bit de poids fort était de poids -2^{n-1} au lieu de 2^{n-1}, arithmétique modulo 2^n.

Addition correcte ou pas ?

Exemple (Addition en NAT)

$$131 + 85 \xrightarrow{\text{code}} 1000\,0011_{\text{NAT}} + 0101\,0101_{\text{NAT}}$$

Correct !

$$\overset{\text{algo}}{=}$$

$$216 \xleftarrow{\text{decode}} 1101\,1000_{\text{NAT}}$$

$$187 + 101 \xrightarrow{\text{code}} 1011\,1011_{\text{NAT}} + 0110\,0101_{\text{NAT}}$$

$$\overset{\text{algo}}{=}$$

$$32 (\neq 288) \xleftarrow{\text{decode}} 0010\,0000_{\text{NAT}}$$

- En C1 et en VA+S, il suffit de mettre un négatif pour opération incorrecte
- En plus, problème du double zéro pour C1 et VA+S.
- C2 : comme si le bit de poids fort était de poids -2^{n-1} au lieu de 2^{n-1}, arithmétique modulo 2^n.

Addition correcte ou pas ?

Exemple (Addition en NAT)

$$131 + 85 \xrightarrow{\text{code}} 1000\,0011_{\text{NAT}} + 0101\,0101_{\text{NAT}}$$

Correct !

$$\xrightarrow{\text{algo}} 1101\,1000_{\text{NAT}}$$

$$216 \xleftarrow{\text{decode}} 1101\,1000_{\text{NAT}}$$

$$187 + 101 \xrightarrow{\text{code}} 1011\,1011_{\text{NAT}} + 0110\,0101_{\text{NAT}}$$

Incorrect !

$$\xrightarrow{\text{algo}} 0010\,0000_{\text{NAT}}$$

$$32 (\neq 288) \xleftarrow{\text{decode}} 0010\,0000_{\text{NAT}}$$

- En C1 et en VA+S, il suffit de mettre un négatif pour opération incorrecte
- En plus, problème du double zéro pour C1 et VA+S.
- C2 : comme si le bit de poids fort était de poids -2^{n-1} au lieu de 2^{n-1}, arithmétique modulo 2^n.

Notes sur la multiplication et l'hexadécimal

- La multiplication est extrêmement simple en binaire. Elle se comporte comme une série d'additions.
- La multiplication est rarement correcte si on a autant de chiffres en entrée qu'en sortie. La plupart des ordinateurs multiplient en mettant le résultat dans deux mots de code distincts (partie haute et partie basse).

Exercices

Limites de la multiplication

Expliquez pourquoi le résultat d'une multiplication de deux nombres représentés dans l'un des 4 codes classiques est toujours représentable à condition de doubler la taille du code.

Addition en C2

Q29 Faites les opérations suivantes en transformant les nombres au préalable en codage C2 sur 8 bits (résultat aussi en C2 sur 8 bits) :

- 45+17
- 45-17 (soit 45+(-17))
- -17-17
- 17-45
- 221+45

Dites aussi si le résultat obtenu est correct et s'il est représentable.

Plan

La logique booléenne

- Il est possible d'interpréter les deux valeurs binaires comme représentant respectivement *vrai* (1) ou *faux* (0).
- On peut définir des opérations correspondantes à la conjonction (et), la disjonction (ou) et la négation (opposé de).

Ce ne sont pas des opérations arithmétiques classiques.

$c = ab$ ou $c = a \wedge b$ pour le *et* (en C : $a = b\&c$)

$c = a + b$ ou $c = a \vee b$ pour le *ou* (en C : $a = b|c$)

$b = \overline{a}$ ou $b = \neg a$ pour le *non* (en C : $a =\sim c$)

Exemple (chemin sous UNIX)

Soit $\mathcal{A}(p)$ la propriété « p est un chemin existant » et $\mathcal{B}(p)$ la propriété « p est un chemin qui désigne un répertoire ». Si on suppose qu'il n'y a que deux type d'objets (répertoires et fichiers), la propriété $\mathcal{C}(p)$ « p est un chemin qui désigne un fichier » s'exprime par $\mathcal{A}(p)\overline{\mathcal{B}(p)}$.

Les opérateurs booléens

AND et OR

AND

L'opérateur binaire AND, noté $a \times b$, renvoie 1 si et seulement si ses deux arguments sont égaux à 1.
L'opérateur général AND renvoie 1 si et seulement si tous ses arguments sont égaux à 1.
Ils sont équivalents à la fonction *minimum*.

OR

L'opérateur binaire OR, noté $a + b$, renvoie 0 si et seulement si ses deux arguments sont égaux à 0.
L'opérateur général OR renvoie 0 si et seulement si tous ses arguments sont égaux à 0.
Ils sont équivalents à la fonction *maximum*.

Les opérateurs booléens

XOR et NOT

XOR

L'opérateur binaire XOR, noté $a \oplus b$, renvoie 1 si et seulement si ses deux arguments sont différents.
Ils sont équivalents à la fonction *différent*.

NOT

L'opérateur unaire NOT, noté $\overline{a}$, renvoie 0 si et seulement si son argument est égal à 1.
Il est équivalent à la fonction *complémentation*.
Il y a aussi les opérateurs généraux NOR et NAND qui sont en fait NOT(OR(...)) et NOT(AND(...)). Ils sont rarement implémentés dans les langages de programmation.

Exercices

Tables de vérité

Q30 Faites une table qui montre toutes les paires d'arguments possibles pour les opérateurs AND, OR, XOR et qui montre le résultat à côté.

A	B	A×B
0	0	
0	1	
1	0	
1	1	

A	B	A+B
0	0	
0	1	
1	0	
1	1	

A	B	A⊕B
0	0	
0	1	
1	0	
1	1	

Les champs de bits

- La notion de variable booléenne stockant une valeur vraie ou faux est très souvent intégrée directement dans les langages

Ce n'est pas le cas dans le langage C

- On appelle parfois ces variables des *flags* (drapeaux).
- Quand on réunit plusieurs de ces variables dans une même entité, on appelle le résultat un champ de bits (anglais *bit field*).
- Ces champs de bits peuvent être stockés dans une seule variable (selon leur nombre). On les considère comme un entier codé en NAT ou C2.
- On définit des opérations sur les champs de bits : le *et bit-à-bit*, le *ou bit-à-bit*, le *not bit-à-bit* et le *xor bit-à-bit*.

Il s'agit de faire sur les bits de même position dans deux champs de bits (un pour la négation) l'opération booléenne correspondante.

Exercices

Opérations booléennes

Q31 Que vaut 0b10000110 AND 0b11101001 ?

Q32 Que vaut 0b10000110 OR 0b11101001 ?

Q33 Que vaut 0b10000110 XOR 0b11101001 ?

Q34 Que se passe-t-il si on calcule (a est une variable booléenne) : $a + 0$? $a + 1$? $a \times 0$? $a \times 1$? $a + a$? $a + a + a + a + a + a$?

Q35 Démontrez que $a + ab = a$;

Q36 Démontrez que $a + bc = (a + b)(a + c)$;

Q37 Démontrez que $a + \overline{a}b = a + b$;

Masquage

Lorsqu'un champ de bits est représenté par un entier, on peut accéder à un bit particulier en procédant à un ET :

$$b_x = (\mathrm{B}\&(1 << x)) >> x$$

On obtient 1 si le bit numéro x est à 1, 0 sinon.
On peut aussi mettre à 1 le bit numéro x :

$$\mathrm{B} = \mathrm{B}|(1 << x)$$

ou à zéro :

$$\mathrm{B} = \mathrm{B}\&(\sim (1 << x))$$

On peut aussi inverser le bit numéro x :

$$\mathrm{B} = \mathrm{B} \oplus (1 << x)$$

On peut tester si par exemple le bit 1 ou 3 sont à 1 : `if (a&0b1010 != 0) ...`
L'ensemble de ces techniques pour manipuler un champ de bits sous la forme d'un entier est appelé *masquage*.
Souvent, les valeurs $(1 << x)$ sont nommées pour qu'on puisse simplement utiliser leur nom au lieu de se souvenir de leur position.

Exercices

Analyse d'un masquage

Dans un champ de bits qui contient $a = 0b11001001$, on veut faire les choses suivantes :

Q38 On veut vérifier si le bit 0 est actif ou non. Décomposez l'opération.

Q39 On veut changer le bit 1 en 1 et le bit 3 en 0. Décomposez les opérations qui permettent de le faire.

Q40 Changez le bit 5, en expliquant les valeurs intermédiaires.

Analyse de touches

Dans un système, la fonction `keyEvent()` renvoie une valeur entière sur 16 bits (dont 5 ignorés) :

- Les 8 premiers bits correspondent au numéro de la touche sur le clavier (pour les touches ordinaires)
- Le 9e bit correspond à la touche SHIFT (1 : pressée, 0 : pas pressée)
- Le 10e bit correspond à la touche CONTROL (1 : pressée, 0 : pas pressée)
- Le 11e bit correspond au fait d'appuyer sur une touche (1) ou de l'avoir juste relachée (0)

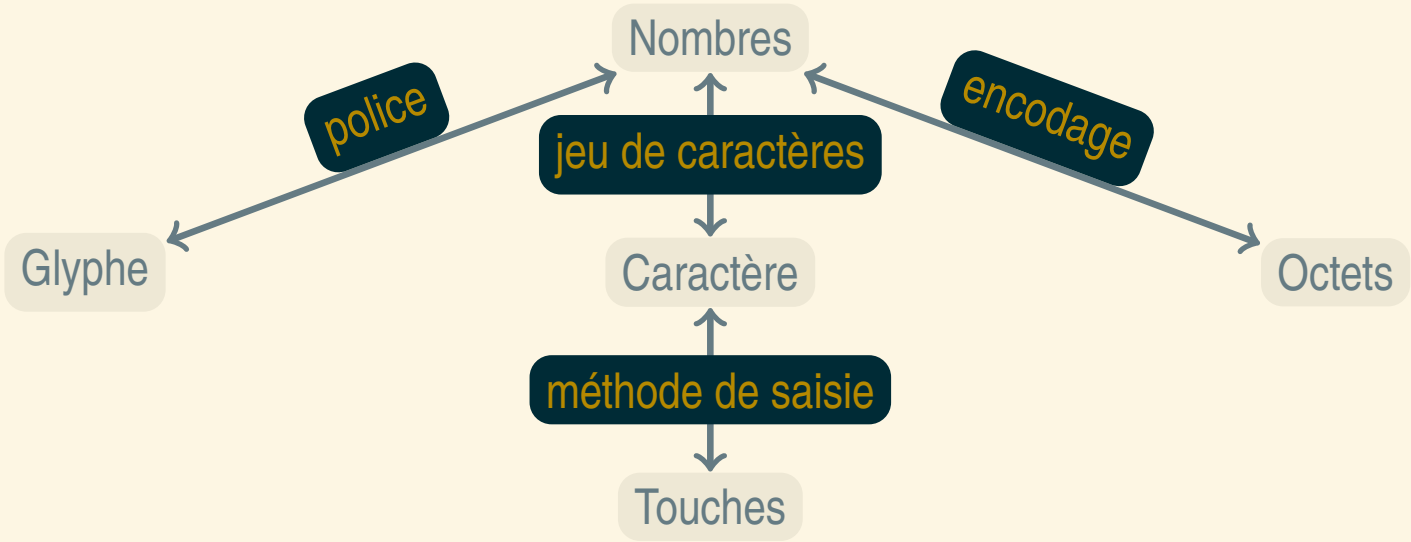
Q41 Écrivez un programme qui appelle cette fonction (`a=keyEvent()`) puis qui en fonction de `a` affiche un texte du genre : « Vous venez de lâcher la touche 27 en ayant SHIFT appuyé et CONTROL lâché »

Plan

Plan

Du texte au(x) glyphe(s)

- Les écrits sous forme d'images ne sont pas exploitables ;
- L'écriture est donc simplifiée pour ne retenir que les *caractères* les uns à la suite des autres (≠ *lettres*) ;



- Les glyphes sont les dessins des lettres, différents selon les polices

Du caractère au glyphe : la police

- Les polices supportent souvent plusieurs jeux de caractères. Le dessin n'y est stocké qu'une fois.
- Une même police peut comporter plusieurs glyphes pour le même caractère (formes décoratives)
- Une police comporte une partie programme pour sélectionner le dessin le mieux adapté

Différence de glyphes

La lettre $\mathcal{A}$ et A représentent le même caractère mais pas le même que A.
De même le *a* de *Abba*, de Abba ou Abba ou Abba sont les mêmes caractères.

Ligatures esthétiques ou linguistiques

La lettre Œ (ligature linguistique) est différente de OE. La lettre fi représente deux caractères, avec affichage fi (ligature esthétique pour éviter le fi).
En arabe ou sanskrit, la ligature est obligatoire mais esthétique : تونس contre ت و ن س.
La ligature esthétique apparaît au niveau des polices, la ligature linguistique au niveau des caractères.

Qu'est-ce qu'un caractère ?

- Au début : lettres, chiffres, ponctuation simplifiée.

Correspondait grossièrement à une touche de machine à écrire (+Majuscule/Minuscule)

- Au fur et à mesure, de très nombreux caractères ont été rajoutés.
- Jeu de caractères universel : Unicode.

Quelques caractères dont vous ne connaissez peut-être pas les noms

C	Usuel	Français	Anglais
#	dièse	croisillon, octothorpe	hash, number sign
&	et	esperluette, et commercial	ampersand, and
|	ou, *païpe*	barre verticale	pipe
/	slash	barre oblique	slash
@	*arobasse*	arobase	at, at sign
\	backslash	contre-oblique	backslash
_	underscore	(blanc) souligné	underscore
[ ]	crochets	crochets	(square) brackets
{ }	accolades	accolades	(curly) braces

Jeux de caractères

- Plusieurs jeux de caractères primitifs sur 7 ou 8 bits par caractère.
- Un seul a vraiment survécu : ASCII
- Création de jeux de caractères nationaux
- Normes ISO-8859-* : caractères 0 à 127 = ASCII ; caractères 128 à 255 = caractères locaux
- Autres méthodes : KOI-8R (russe), JIS (Japonais), BIG5 (Chinois)... collections de caractères
- Universalisation : Unicode : plus de 100 000 caractères.

Voir la table

Certains caractères sont dupliqués pour des raisons historiques

Exercices

La table ASCII

Trouvez dans la table ASCII :

1. Le caractère de code 0x41
2. Le caractère de code 0x30
3. Le caractère *a* et *A*. Comparez l'écriture binaire des codes numériques correspondants.
4. Le caractère de code 0x20. Quel est-il ?
5. Le caractère *retour chariot* (son nom est NEWLINE ou NL).

Comment passe-t-on d'une lettre à la suivante ? D'une majuscule à une minuscule ?

Plan

Au début était le *byte*

- Premiers codages : un caractère = un *byte* = 6 à 8 bits
- Rapidement *byte* = octet = 8 bits. ASCII sur 8 bits avait un bit inutilisé.
- Langues asiatiques : pas suffisant.
- Codage à décalage : certaines séquences (non rencontrées habituellement) permettent de changer de « zone » de caractères.
- Certaines séquences déclenchent du codage où 1 caractère est codé par 2 octets.
- Rupture de l'égalité 1 octet = 1 caractère
- Autres codages : BIG5 est un codage à 2 octets par caractères pour le chinois.

Le Mojibake

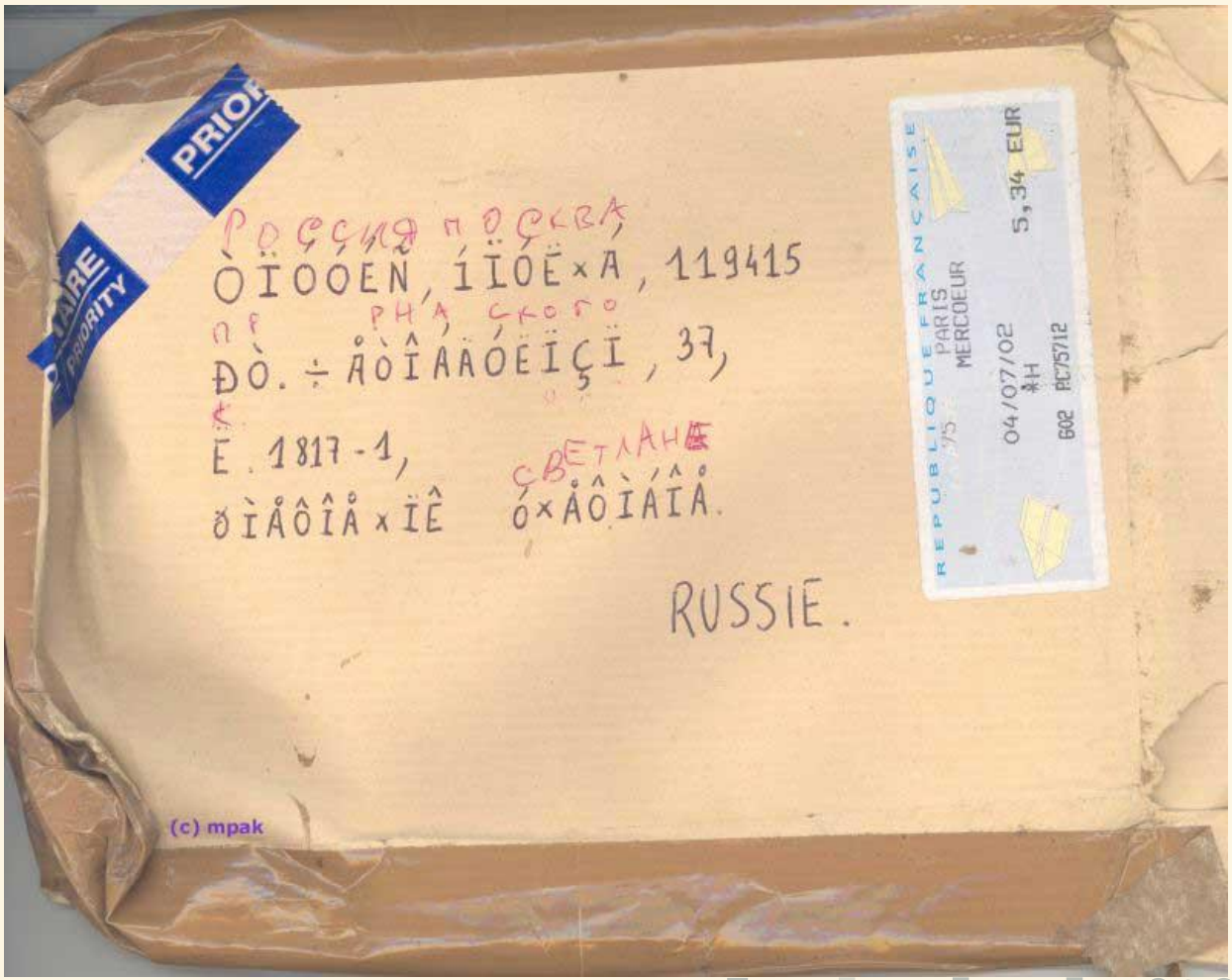
L'enveloppe était envoyée à un étudiant russe par une amie française qui a recopié son adresse reçue par e-mail. Le logiciel ne savait pas lire les caractères cyrilliques (page de code KOI8-R) et les a remplacés par les caractères du code ISO-8859-1.

En KOI8-R :
Россия Москва, 119415
пр.Вернадского, 37,
к.1817-1,
Плетневой Светлане

En ISO-8859-1 :
òÏÓÓÉÑ íÏÓË×Á, 119415
ÐÒ.÷ÁÒÎÁÄÓËÏÇÏ, 37,
Ë.1817-1,
ðÌÅÔÎÅ×ÏÊ ó×ÅÔÌÁÎÅ

Le postier a réussi à faire la transformation inverse ! (en rouge)

Une enveloppe en krakozyabry (кракозябры) (aussi Mojibake).



Exercices

Codage nationaux et Mojibake

Soit le texte : Coefficient marée trop fort pour livraison tomates cœur-de-bœuf

1. Identifiez dans ce texte les ligatures linguistiques et les ligatures esthétiques
2. Est-il possible de représenter ce texte dans le jeu de caractères ASCII ?
3. Dans le jeu de caractère ISO-8859-15 (dit *latin-9*), il est possible de coder ce texte. Chaque caractère est alors codé par un octet unique. Quelle est la taille du fichier qui contient uniquement ce texte ?
4. Un polonais lit sur son vieil ordinateur le texte précédent. Il voit qu'une des lettres a été remplacée par ˝ (c'est un double accent aigu, comme dans Erdős, et pas un tréma comme dans Gwenaël). Laquelle et pourquoi ? S'il renvoie le texte tel quel a son correspondant français du début, que verra le français et pourquoi ?

Unicode et UTF-8

- Unicode est une collection de plus de 100 000 caractères qui ne spécifie pas la façon de le représenter par une séquence d'octets. La taille maximale est de 17×2^{16} et le code maximal 0x10FFFF
- UTF-8 est une façon de transformer un numéro en une séquence d'octets

Valeurs	Écriture binaire	Codage UTF-8 (binaire)	octets
0x0–0x7F	abc defg	0abc defg	1
0x80–0x7FF	abc defg hijk	110a bcde 10fg hijk	2
0x800–0xFFFF	abcd efgh ijkl mnop	1110 abcd 10ef ghij 10kl mnop	3
0x10000–0x1FFFFF	a bcde fghi jklm nopq rstu	1111 0abc 10de fghi 10jk lmno 10pq rstu	4

- UCS-2 est un codage partiel sur 2 octets par caractères (représente les 2^{16} premiers caractères)
- UTF-16 est un codage plus simple qu'UTF-8 utilisant 2 ou 4 octets par caractères : 2 pour les premiers, 4 pour les autres (10 octets par paire de 2 octets).

Avantage de l'UTF-8 : économe en place pour l'ASCII (1 octet par caractère)

Inconvénient de l'UTF-8 : impossible de dire facilement à quel octet est le n^{e} caractère.

Exercices

UTF8

1. Le caractère de numéro 0x0041 (A) est codé par quel(s) octet(s) en UTF-8 ?
2. Le caractère de numéro 0x00E9 (é) est codé par quel(s) octet(s) en UTF-8 ?
3. Le caractère de numéro 0x0F03 () est codé par quel(s) octet(s) en UTF-8 ?
4. Le caractère de numéro 0x12084 () est codé par quel(s) octet(s) en UTF-8 ?
5. Dans un fichier codé en UTF-8, on trouve les six octets suivants. Combien de caractères sont réellements codés dans ce texte ?

   `0xE6 0x9D 0x8c 0xDE 0xBC 0x43`

6. L'anglais n'utilise que des caractères dont le numéro est dans la première ligne, et est codé traditionnellement en ISO-8859-1 (1 caractère = 1 octet). Le français utilise 5% de caractères de la deuxième ligne (le reste de la première), et est codé pareil (1 caractère = 1 octet). L'arabe (le russe, l'hébreu, le grec) sont aussi codés traditionnellement par 1 caractère = 1 octet, et comportent 95% de caractères de la deuxième ligne (le reste de la première ligne). Le chinois, en revanche est traditionnellement codé en BIG5 (1 caractère = 2 octets). Les textes chinois sont à 99% des caractères de la troisième ligne (le reste de la première ligne).
   Pour un texte de 1000 caractères codé en UTF-8, combien d'octets seront utilisés en moyenne pour un texte anglais, français, russe et chinois ?
7. Quel est en chinois l'augmentation de la taille du texte par rapport au codage traditionnel ?

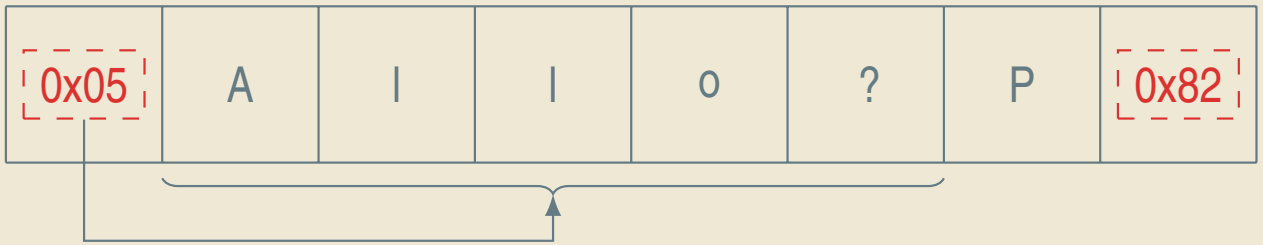
Plan

Les chaînes avec longueur spécifiée

Les chaînes de caractères sont des listes ordonnées de caractères.
Lorsqu'une chaîne de caractères est stockée en mémoire, elle occupe plusieurs positions consécutives dans la mémoire. On désigne souvent la chaîne par la première position occupée.
Certains langages résolvent le problème de savoir où la chaîne s'arrête en stockant aussi la longueur.
Problème avec certains codages/jeux de caractères pour trouver le n^e élément d'une chaîne (et en particulier, la longueur en nombre de caractères).
Avantage : le calcul de la place mémoire occupée est instantané.

Exemple

On stocke ici la chaîne « Allo ? » (le P et la valeur 0x82 sont des éléments qui sont dans la mémoire mais ne font pas partie de la chaîne).



Est-ce que la longueur est en caractères ou en octets ?

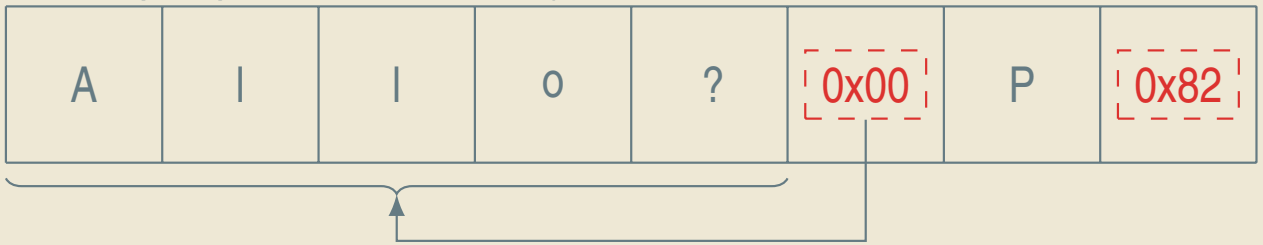
En **octets**, le plus souvent, ou les deux. le plus important est de savoir trouver la fin de la chaîne (pour pouvoir la copier).

Les chaînes avec marqueurs de fin

Une autre possibilité est de marquer la fin de la chaîne avec un octet particulier ou une séquence d'octets particulière. C'est le cas du langage C (et de beaucoup d'autres langages dérivés) qui utilise le caractère nul.

Exemple

On stocke ici la chaîne « Allo ? » (le P et la valeur 0x82 sont des éléments qui sont dans la mémoire mais ne font pas partie de la chaîne).



Est-ce que le marqueur fait partie de la chaîne ?

En pratique, oui. Mais il ne fait pas partie du texte codé par la chaîne.
À l'intérieur d'un langage il n'y a en général qu'une seule sorte de chaîne.

Le problème de l'échappement

La fin de chaîne

Quand la longueur n'est pas spécifiée à côté d'une chaîne, la fin de la chaîne est forcément indiquée par une séquence spécifique de bits.

- S'il existe une séquence spécifique invalide dans le codage pour la représentation de caractères, alors on peut la choisir comme représentant la fin de chaîne.
- Sinon, il faut choisir un caractère qui va coder la fin de la chaîne

Comment coder une chaîne qui comporte ce caractère ?

Les séquences signifiantes

Parfois, on veut pouvoir utiliser dans des chaînes des séquences qui ont un sens spécial. Par exemple, on pourrait vouloir que `0x0F03` représente le caractère ྃ qu'on ne peut pas rentrer facilement au clavier. Mais dans ce cas, comment écrire la chaîne `0x0F03` (comme par exemple pour la phrase « Si on met `0x0F03` dans une chaîne on obtient le caractère ྃ » ?

Il faut donc utiliser une procédure d'échappement !

Les séquences d'échappement

- On utilise une séquence (parfois codante) d'échappement qui permet de modifier le sens des caractères qui suivent
- Si la séquence d'échappement est codante, on doit prévoir au moins une combinaison qui permet de redonner le caractère d'échappement
- Avoir des chaînes interprétables complique énormément les opérations élémentaires, comme calculer le nombre de caractères dans la chaîne, ou savoir si un caractère est présent dans la chaîne.

On se retrouve souvent à « empiler » les modes d'échappement identiques ou différents.

Exemple

En langage C et dérivés, le caractère \ est utilisé pour introduire des séquences d'échappement.
`\0` est le caractère nul.
`\n` est le caractère 10 (NEWLINE).
`\t` est le caractère 9 (TAB).
`\xxx` est le caractère de numéro octal xxx.
`\Uxxxx` est le caractère unicode de numéro hexadécimal xxxx. Ce caractère unicode peut représenter plusieurs octets. Le codage choisi dépend du compilateur et du type de la chaîne.

Exercices

Les échappements en C

Dessinez quelle est la structure en mémoire des chaînes C suivantes ? Comment sont elles affichés ?

1. `"Toto"`
2. `"Bonjour le monde\n"`
3. `"Acheter:\n\tponey\n\tporte-avions\n"`
4. `"\303\251\n"`
5. `"\U20AC"` (symbole euro)
6. `"\0"`

Une bizarrerie historique du C/C++ fait que certaines séquences sont remplacées avant compilation par d'autres caractères :

Trigraphe	??(	??)	??<	??>	??=	??/	??'	??!	??-
Remplacement	[	]	{	}	#	\	^	|	~

7. `"Hello??!"`
8. `"Bye??/n"`

Plan

Plan

L'assemblage de types

Presque tous les langages de programmation utilisent une notion de *types de base* et de *composition de types*.

Types de base

Les langages ont presque tous un type *entier* et un type *flottant* (certains n'ont pas le premier). Il y a aussi souvent un type *booléen* et un type *caractère* qui désigne un caractère.
Ces types de base sont les éléments les plus

Assemblage

Les assemblages typiques sont les **assemblages par répétition** (structures de *tableau* (taille fixe), de vecteur ou de liste (taille arbitraire), mais aussi des **assemblages hétérogènes** de taille fixe.
Une dernière option est l'assemblage de type *variante* qui consiste à pouvoir mettre dans un même emplacement un type ou un autre. Certains langages ne spécialisent même pas les variables.

Exercices

Types simples ou composés

Q42 Identifiez dans les types suivants lesquels sont susceptibles d'êtres des types de base et lesquels sont plutôt des types construits par assemblage :

- Un nombre entier positif ou nul
- Un nombre complexe
- Un point dans l'espace
- Un nombre avec un très grand nombre de chiffres non fixé à l'avance
- Un intervalle
- Une date
- Un étudiant (nom, prénom, date de naissance)
- Un caractère
- Une chaîne de caractères

Une date

Q43 Décrivez à partir de quels éléments on peut composer une donnée qui représente un moment précis de la journée.

Q44 Discutez les éléments précis selon que l'on considère qu'un moment est pris à la seconde près ou beaucoup plus précis.

Éléments de taille variable

Il arrive que l'on veuille représenter une structure complexe qui compte un nombre variable d'autres structures.

Par exemple une chaîne de caractères !

- Problème de détection de la fin.
- Codage du nombre d'éléments, ou séquence de fin.

Lors de la construction de structures complexes à partir de structures élémentaires, on peut externaliser la structure de taille variable.

Une structure de taille fixe est plus facilement manipulable (notamment pour en faire des vecteurs ou des listes).

Représentation en mémoire

Représentation dans un fichier

- L'unité de base est l'octet
- Les données peuvent être indexées (on connaît le début de chaque donnée)
- Les données peuvent être typées (on connait le type de chaque donnée)
- Compromis entre taille occupée et résistance aux erreurs ou déchiffrabilité

Représentation en mémoire vive

- Les données simples ont souvent une représentation en mémoire qui occupe un nombre d'octets contigus fixe.
- Les données composites sont stockées par juxtaposition des données élémentaires qui les composent.
- Les données de taille variables sont *externalisées*. On les connaît alors par leur **adresse**, l'emplacement mémoire où elles sont stockées.

Plan

Du nombre de bits d'un processeur

- Un processeur manipule des données avec une taille fixe

Il y a parfois plusieurs tailles manipulables

- Historique :
  - Ordinateurs 8 bits : 1972–1985
  - Ordinateurs 16 bits : 1975–1990
  - Ordinateurs 32 bits : 1986–maintenant
  - Ordinateurs 64 bits : 1992/2003–maintenant
- Pour les flottants ou les données graphiques, il y a des circuits spécialisés qui ont des tailles différentes (plus grandes)
- Les données de taille supérieure doivent être manipulées en plusieurs opérations et ne sont pas des données simples

C'est la taille du *mot-machine*.

Données simples du C

- `char` (C2 8 bits)
- `short int` (C2 $\geq$16 bits)
- `int` (C2 $\geq$16 bits, usuellement 32)
- `long int` (C2 $\geq$32 bits)
- `long long int` (C2 $\geq$64 bits)
- `float` est l'IEEE754 simple précision (32 bits), `double` est la double précision (64 bits), `long double` est la précision étendue ou quadruple selon les processeurs, $\geq$80 bits).
- une *adresse* permet de désigner une autre donnée dans la mémoire. Leur taille a varié dans l'histoire de l'informatique.
- N'oubliez pas qu'un nombre peut cacher un champ de bits
- Des versions `unsigned` de toutes les longueurs d'entiers existent et permettent de choisir le codage NAT au lieu de C2.

Les modèles de programmation 32 et 64 bits

Besoin de garder la compatibilité des codes sources.
En C et en C++ les types changent de taille.

Modèle	Taille `short int`	Taille `int`	Taille `long`	Taille adresse	Taille `long long`	Exemples de Compilateurs
32 bits	16	32	32	32	—	
LP64	16	32	64	64	—	Tous sauf...
ILP64	16	64	64	64	—	Exceptions...
LLP64	16	32	32	64	64	Microsoft...

Alignement

Qu'est-ce que l'alignement ?

Les processeurs présentent des contraintes techniques pour l'adressage des données. Une opération sur une donnée de type simple en mémoire doit se faire avec une adresse multiple de sa taille.
Les types de taille supérieure au mot-machine sont de toute façon manipulée en morceaux indépendants.

Exemple

Sur une machine 32 bits, un `int` (4 octets) ne peut pas commencer à l'adresse `0x00000002`. Il commence soit à l'adresse `0x00000000`, soit `0x00000004`. Par contre, un `short int` de 16 bits pourra commencer à cette adresse.

L'alignement dans les données composées

Sauf exception, l'alignement doit être respecté pour toutes les composantes de la donnée composée. L'adresse de la donnée composée est l'adresse de la première composante, et chaque composante doit être alignée correctement vis-à-vis de sa taille.

Exercices

Stockage d'une date (suite)

Une date est composée des éléments suivants :

- Une année (disons de -2 milliards à +2 milliards)
- Un mois, un jour du mois
- Un fuseau horaire qui est une « adresse »
- Une heure, une minute (entiers)
- Un nombre de secondes qui est un flottant simple précision

Q45 Dites quels sont les types de base du C à utiliser pour coder cette information, d'après les limites connues de stockage pour chaque type. Utilisez les tailles les plus petites possibles.

Q46 Donnez leur noms à la fois dans un modèle 32 bits et un modèle 64 bits.

Q47 Si on avait voulu aller de -5 milliards à +5 milliards d'années, quel type aurait-on dû utiliser ?

Q48 Si les données sont dans l'ordre indiqué dans un type composé, précisez à quel moment les contraintes d'alignement provoquent des « trous » dans la structure.

Q49 Quelle est la taille totale de la structure (avec les trous) ?

Grand et petit-boutien (little/big-endian)

- Valeur `0x4A3B2C1D`, adresse `0x00000000` ?

Plusieurs représentations possibles :

- Contenu | 4A | 3B | 2C | 1D | petit-boutien
  Adresse | 03 | 02 | 01 | 00 | *(little-endian)*
- Contenu | 1D | 2C | 3B | 4A | grand-boutien
  Adresse | 03 | 02 | 01 | 00 | *(big-endian)*

- little-endian : 6502, x86, VAX.
- big-endian : Motorola 68000, SPARC, System/370.
- bi-endian : ARM, PowerPC (sauf G5), MIPS.
- Les bi-boutiens ont un mode par défaut (big-endian pour PowerPC, little-endian pour IA-64).

Grand et petit-boutien (little/big-endian)

- Valeur `0x4A3B2C1D`, adresse `0x00000000` ?

Plusieurs représentations possibles :

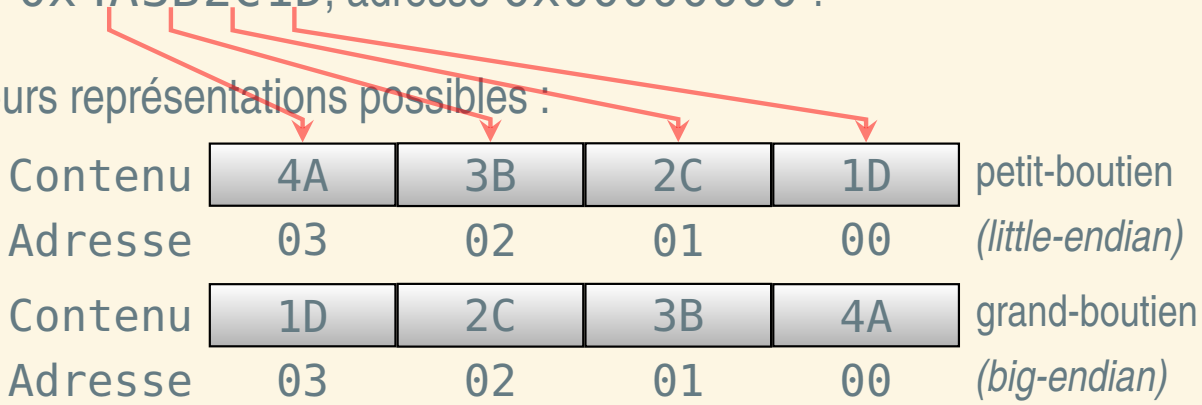
- `Contenu` | `4A` | `3B` | `2C` | `1D` | petit-boutien
  `Adresse` | `03` | `02` | `01` | `00` | *(little-endian)*
- `Contenu` | `1D` | `2C` | `3B` | `4A` | grand-boutien
  `Adresse` | `03` | `02` | `01` | `00` | *(big-endian)*

- little-endian : 6502, x86, VAX.
- big-endian : Motorola 68000, SPARC, System/370.
- bi-endian : ARM, PowerPC (sauf G5), MIPS.
- Les bi-boutiens ont un mode par défaut (big-endian pour PowerPC, little-endian pour IA-64).

Grand et petit-boutien (little/big-endian)

- Valeur `0x4A3B2C1D`, adresse `0x00000000` ?

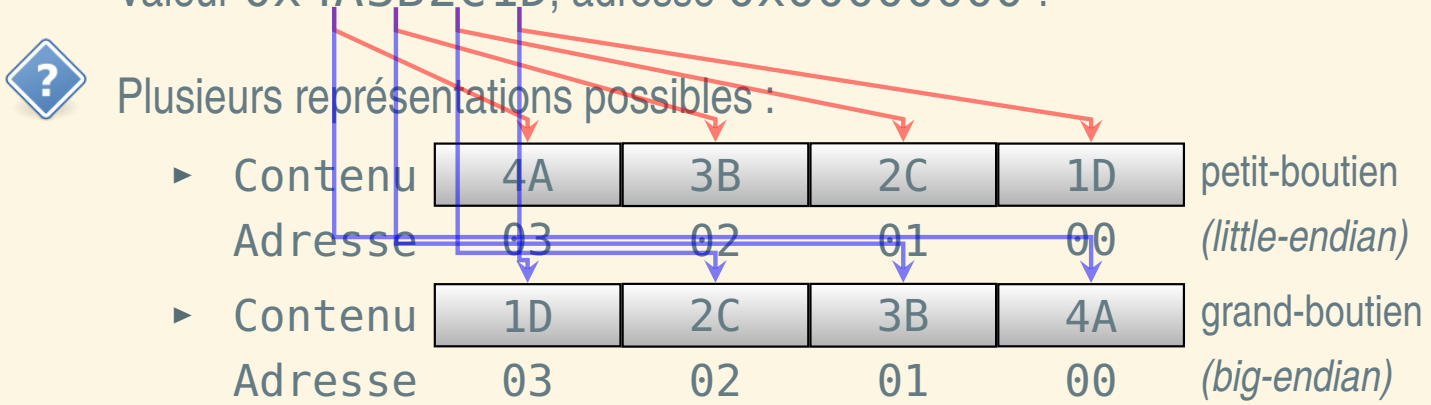
Plusieurs représentations possibles :



- little-endian : 6502, x86, VAX.
- big-endian : Motorola 68000, SPARC, System/370.
- bi-endian : ARM, PowerPC (sauf G5), MIPS.
- Les bi-boutiens ont un mode par défaut (big-endian pour PowerPC, little-endian pour IA-64).

Grand et petit-boutien (little/big-endian)

- Valeur `0x4A3B2C1D`, adresse `0x00000000` ?



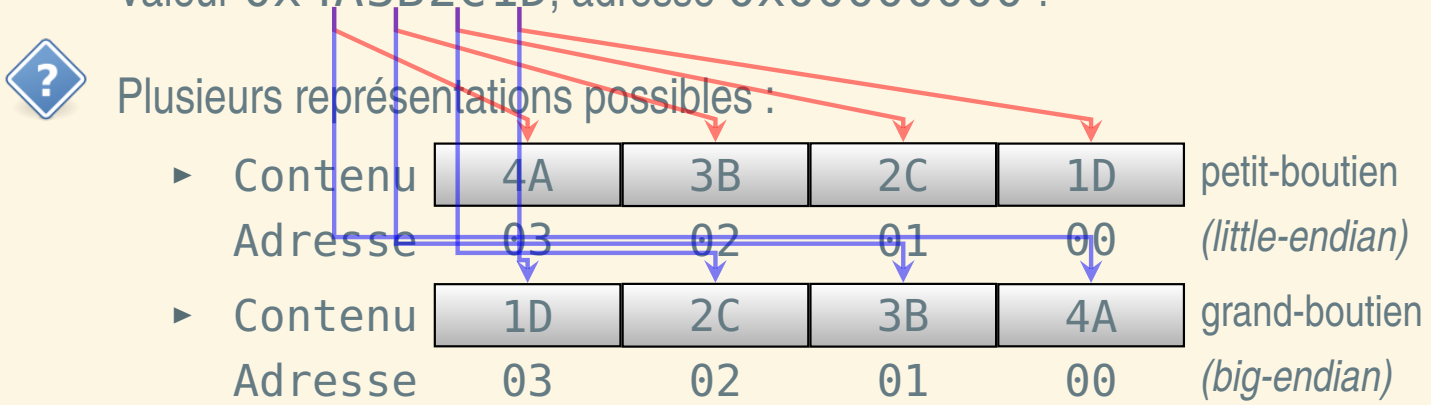
petit-boutien *(little-endian)*

grand-boutien *(big-endian)*

- little-endian : 6502, x86, VAX.
- big-endian : Motorola 68000, SPARC, System/370.
- bi-endian : ARM, PowerPC (sauf G5), MIPS.
- Les bi-boutiens ont un mode par défaut (big-endian pour PowerPC, little-endian pour IA-64).

Grand et petit-boutien (little/big-endian)

- Valeur `0x4A3B2C1D`, adresse `0x00000000` ?



- little-endian : 6502, x86, VAX.
- big-endian : Motorola 68000, SPARC, System/370.
- bi-endian : ARM, PowerPC (sauf G5), MIPS.
- Les bi-boutiens ont un mode par défaut (big-endian pour PowerPC, little-endian pour IA-64).

Exercices

Stockage d'une date (suite)

Q50 On veut stocker la date du 24 décembre -4 à 21h45 dans un système 32 bits. Calculez les valeurs à stocker en hexadécimal (hormis l'adresse du fuseau horaire). Vous prendrez comme valeur d'adresse pour le fuseau horaire 0x12345678.

Q51 Écrivez, les uns après les autres, les octets qui composent cette date si on est dans un système 32 bits *big-endian*

Q52 Écrivez, les uns après les autres, les octets qui composent cette date si on est dans un système 32 bits *little-endian*

Plan

Exercices

Information intrinsèque

Q53 Si vous lancez une pièce de monnaie 20 fois en l'air, est-ce que vous avez plus de chance de tomber sur 20 fois face (F), 10 fois face-pile (FP), ou sur FPFPPFFFPPPPFPFFPPFP ?

Q54 Quelle est la quantité d'information contenue dans la suite binaire 110111001001 ? Et dans la suite binaire 000000000000 ? Et dans la suite binaire 010101010101 ?

Q55 Est-ce qu'on pourrait écrire certaines de ces suites de façons plus courtes ? Proposez-en.

La compression de données sans pertes

Complexité de Kolmogorov

La complexité d'une suite binaire est le programme le plus court qui permet de l'écrire (dans un langage adéquat).

Pour certaines suites, le programme le plus court est celui qui les contient. C'est lié à la notion d'aléatoire en *calculabilité*.

Cette complexité ne peut pas être calculée par un programme. Par contre, il est possible de trouver des programmes plus courts pour la plupart des données usuelles. C'est la **compression** !

Compression

La compression permet d'économiser de la place sur les disques et en mémoire.
Elle pénalise l'accès direct aux données. Elle se paye par des calculs de *décompression* pour y accéder.
En gagnant de la place sur le disque dur, elle permet d'accélérer le traitement des données, la lecture depuis le disque étant beaucoup plus lente que les calculs pour décompresser en général.

Les utilitaires classiques

L'archivage

L'archivage consiste à transformer toute une hiérarchie de fichiers en un fichier unique.

Ce n'est pas de la compression.

- Souvent couplé à une ou plusieurs méthodes de compression.
- Utilitaire `tar` sous Unix, programmes commerciaux `zip` ou `rar`.

Les formats classiques

- `gzip` utilise le codage de Lempel-Ziv (extension usuelle : `gz`)
- `zip` utilise le codage de Lempel-Ziv et de Huffman par dessus (extension usuelle : `zip`, mais aussi d'autres formats : `jar`, `odt`)
- `rar` (logiciel commercial) utilise le codage de Lempel-Ziv avec des techniques de prédiction PPM par dessus (extension usuelle : `rar`)

La compression RLE

Codage par plages

La compression RLE (pour *Run Length Encoding*) est une des compressions les plus simples : on transforme une suite de symboles en une suite de paires (nombre de symboles à répéter – symbole). Par exemple 00000011110011 se code **6**0**4**1**2**0**2**1.
Il peut mener à une chaîne plus grande que l'originale (par exemple pour 010101 : **1**01**1**10**1**11**1**1).
Le compte est souvent codé sur une taille fixe (un octet). Lorsqu'on dépasse, on peut découper en plusieurs *runs* de longueur maximale (sauf le dernier). Exemple : **255**0**45**0 pour un chaîne de 300 fois 0.
Lorsqu'on a seulement deux symboles, on peut ne noter que le compte, pas le symbole. On autorise alors les *runs* de 0 caractères, et on indique le caractère initial. Exemple : 0**255 0 45 1** pour 300 zéros suivis d'un 1.

Exercices

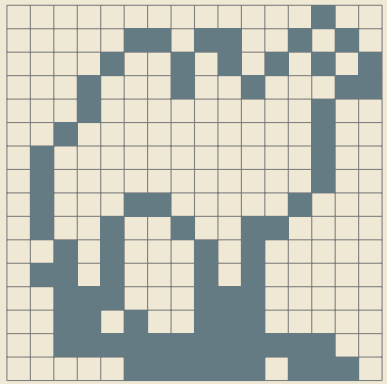
Compression RLE

Q56 Voilà des suites à compresser avec la compression RLE :

- 000011110010100001111111
- 11121121111122131221113112221111321321 1
- 0110100110010110

Protocole de fax

Q57 Un fax est une suite de points noirs et blancs. En utilisant les notations du RLE binaire données dans le cours, décrivez l'image à droite. On part en haut à gauche de l'image, on part à droite et on reprend à la fin de chaque ligne à la ligne suivante à gauche.



Plan

Plan

Les images matricielles (bitmap)

Les images issues du processus de discrétisation sont décomposés en *pixels* (souvent rectangulaires).

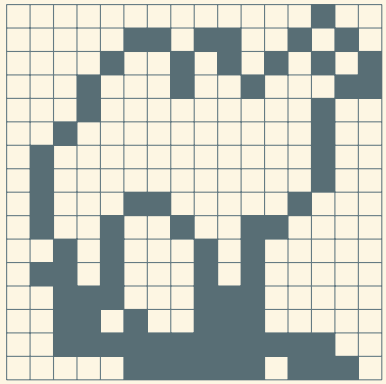
les formats RAW (qui contiennent les données brutes issues des capteurs d'appareils photo) ont des grilles qui sont plutôt des pavages par des triangles ou des hexagones.

Une image est un format *composite*. Il contient entre autres :

- la largeur et la hauteur en pixels (obligatoire),
- les données d'intensité de chaque pixel (obligatoire),
- des informations sur le rendu des couleurs (voir plus loin, le gamma et le gamut),
- des informations sur le rendu physique (par exemple, dimensions en cm),
- des méta-données (sujet, auteur, lieu de prise).

Le format PBM

- Le format PBM est un format matriciel qui est composé des éléments suivants :
  - une entête en mode texte qui commence par P4 en ASCII, du blanc,
  - la largeur écrite en ASCII, du blanc,
  - la hauteur écrite en ASCII, **un** blanc,
- Les données binaires ligne par ligne de haut en bas.
- Chaque ligne est découpée en paquet de 8 pixels noirs ou blancs consécutifs, codés en binaire (1=noir). Si la largeur n'est pas un multiple de 8, on complète par des 0 à droite.
- Les blancs sont les caractères ASCII espace et TAB, CR, NL (0x09, 0x0A, 0x0D)



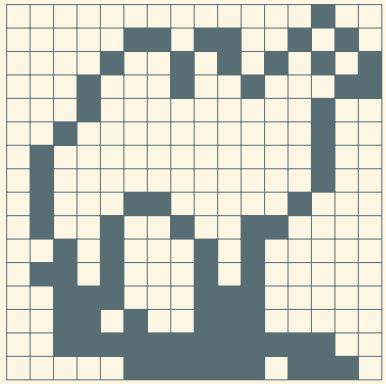
L'entête en ASCII :

```
P4\n
16 16\n
```

```
ADRESSE   OCTETS EN HEXADECIMAL
00000000  50 34 0a 31 36 20 31 36 0a 00 04 06 ca 09 55 11
00000010  23 10 04 20 04 40 04 40 04 46 08 49 30 28 a0 68
00000020  a0 38 e0 34 e0 3f fc 07 ee
```

Le format PBM

- Le format PBM est un format matriciel qui est composé des éléments suivants :
  - une entête en mode texte qui commence par P4 en ASCII, du blanc,
  - la largeur écrite en ASCII, du blanc,
  - la hauteur écrite en ASCII, **un** blanc,
- Les données binaires ligne par ligne de haut en bas.
- Chaque ligne est découpée en paquet de 8 pixels noirs ou blancs consécutifs, codés en binaire (1=noir). Si la largeur n'est pas un multiple de 8, on complète par des 0 à droite.
- Les blancs sont les caractères ASCII espace et TAB, CR, NL (0x09, 0x0A, 0x0D)



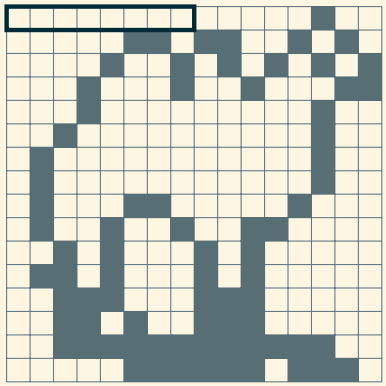
L'entête en ASCII :

```
P4\n
16 16\n
```

```
ADRESSE   OCTETS EN HEXADECIMAL
00000000  50 34 0a 31 36 20 31 36 0a 00 04 06 ca 09 55 11
00000010  23 10 04 20 04 40 04 40 04 46 08 49 30 28 a0 68
00000020  a0 38 e0 34 e0 3f fc 07 ee
```

Le format PBM

- Le format PBM est un format matriciel qui est composé des éléments suivants :
  - une entête en mode texte qui commence par P4 en ASCII, du blanc,
  - la largeur écrite en ASCII, du blanc,
  - la hauteur écrite en ASCII, **un** blanc,
- Les données binaires ligne par ligne de haut en bas.
- Chaque ligne est découpée en paquet de 8 pixels noirs ou blancs consécutifs, codés en binaire (1=noir). Si la largeur n'est pas un multiple de 8, on complète par des 0 à droite.
- Les blancs sont les caractères ASCII espace et TAB, CR, NL (0x09, 0x0A, 0x0D)



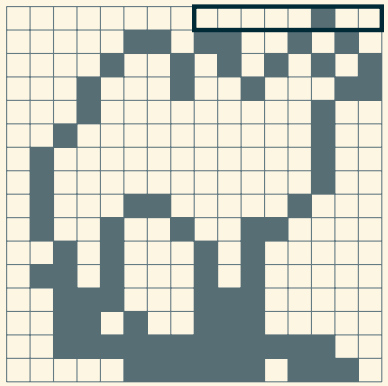
L'entête en ASCII :

```
P4\n
16 16\n
```

```
ADRESSE   OCTETS EN HEXADECIMAL
00000000  50 34 0a 31 36 20 31 36 0a 00 04 06 ca 09 55 11
00000010  23 10 04 20 04 40 04 40 04 46 08 49 30 28 a0 68
00000020  a0 38 e0 34 e0 3f fc 07 ee
```

Le format PBM

- Le format PBM est un format matriciel qui est composé des éléments suivants :
  - une entête en mode texte qui commence par P4 en ASCII, du blanc,
  - la largeur écrite en ASCII, du blanc,
  - la hauteur écrite en ASCII, **un** blanc,
- Les données binaires ligne par ligne de haut en bas.
- Chaque ligne est découpée en paquet de 8 pixels noirs ou blancs consécutifs, codés en binaire (1=noir). Si la largeur n'est pas un multiple de 8, on complète par des 0 à droite.
- Les blancs sont les caractères ASCII espace et TAB, CR, NL (0x09, 0x0A, 0x0D)



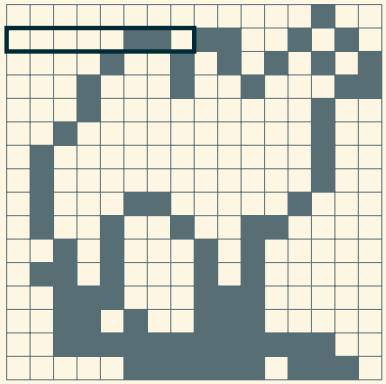
L'entête en ASCII :

```
P4\n
16 16\n
```

```
ADRESSE   OCTETS EN HEXADECIMAL
00000000  50 34 0a 31 36 20 31 36 0a 00 04 06 ca 09 55 11
00000010  23 10 04 20 04 40 04 40 04 46 08 49 30 28 a0 68
00000020  a0 38 e0 34 e0 3f fc 07 ee
```

Le format PBM

- Le format PBM est un format matriciel qui est composé des éléments suivants :
  - une entête en mode texte qui commence par P4 en ASCII, du blanc,
  - la largeur écrite en ASCII, du blanc,
  - la hauteur écrite en ASCII, **un** blanc,
- Les données binaires ligne par ligne de haut en bas.
- Chaque ligne est découpée en paquet de 8 pixels noirs ou blancs consécutifs, codés en binaire (1=noir). Si la largeur n'est pas un multiple de 8, on complète par des 0 à droite.
- Les blancs sont les caractères ASCII espace et TAB, CR, NL (0x09, 0x0A, 0x0D)



L'entête en ASCII :

```
P4\n
16 16\n
```

```
ADRESSE   OCTETS EN HEXADECIMAL
00000000  50 34 0a 31 36 20 31 36 0a 00 04 06 ca 09 55 11
00000010  23 10 04 20 04 40 04 40 04 46 08 49 30 28 a0 68
00000020  a0 38 e0 34 e0 3f fc 07 ee
```

Exercices

Décodage d'un fichier PBM

Voici la séquence d'octets qui compose un fichier PBM :

```
50 34 0a 38 20 31 30 0a 41 41 3e 55 41 49 bf 12 24 22
```

Q58 Repérez l'entête du fichier, et traduisez-là en ASCII.

Q59 Quelle est la taille de cette image (en pixels) ?

Q60 Dessinez le fichier résultant.

Q61 Quelle est la taille (minimale) de l'entête et la taille des données ?

Q62 Même question pour une image 8000 par 8000.

Le format JPG

- Format développé par comité *(Joint Picture Expert Group)*
- Adapté au stockage de photographies
- Méta-données riches (EXIF, XMP, ITPC)
- Filtrage perceptuel numérique paramétrable : réduire les couleurs (filtrage hautes-fréquences)
- Compression sur un filtrage plus ou moins intense *(qualité)*
- Stockage avec perte de qualité *(artefacts)*

Lena, 90%, 90 ko

Lena, 50%, 30 ko

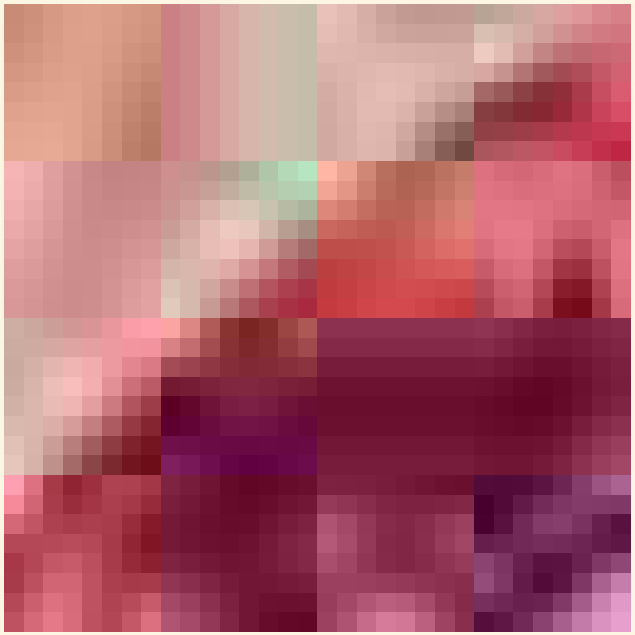
Lena, 10%, 10 ko

Le format JPG

- Format développé par comité *(Joint Picture Expert Group)*
- Adapté au stockage de photographies
- Méta-données riches (EXIF, XMP, ITPC)
- Filtrage perceptuel numérique paramétrable : réduire les couleurs (filtrage hautes-fréquences)
- Compression sur un filtrage plus ou moins intense *(qualité)*
- Stockage avec perte de qualité *(artefacts)*

Lena, 90%, 90 ko

Lena, 50%, 30 ko



Lena, 10%, 10 ko

Le format PNG

- Format *Portable Network Graphics*
- Méta-données liées aux couleurs
- Pas de filtrage perceptuel numérique
- Compression tenant compte de l'aspect 2D
- Support de la transparence et de la translucidité (transparence partielle)
- Support d'une palette de couleurs : les données colorimétriques de chaque point sont regroupées dans une palette unique, et on ne note que le numéro de couleur dans la palette pour chaque point.

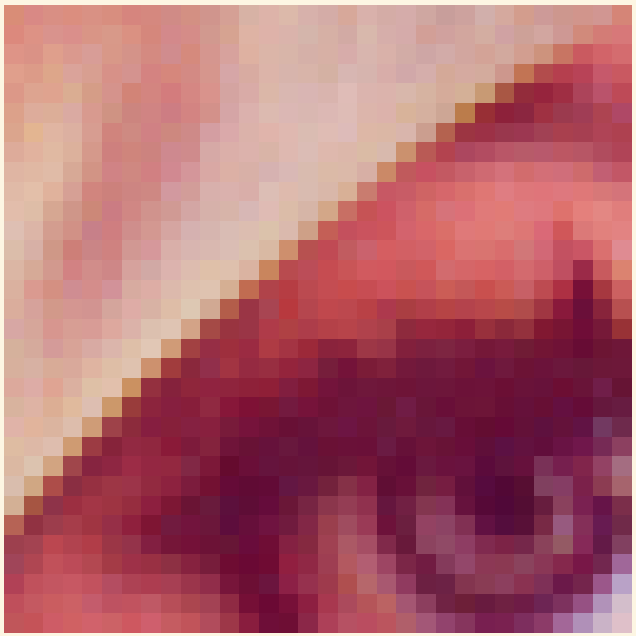
Lena, 475 ko

Lena, palette 256c, 189 ko

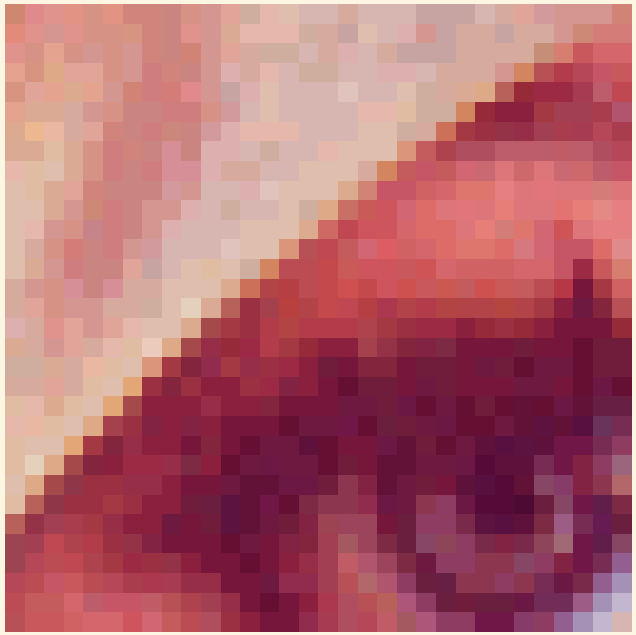
Lena, palette 4c, 50 ko

Le format PNG

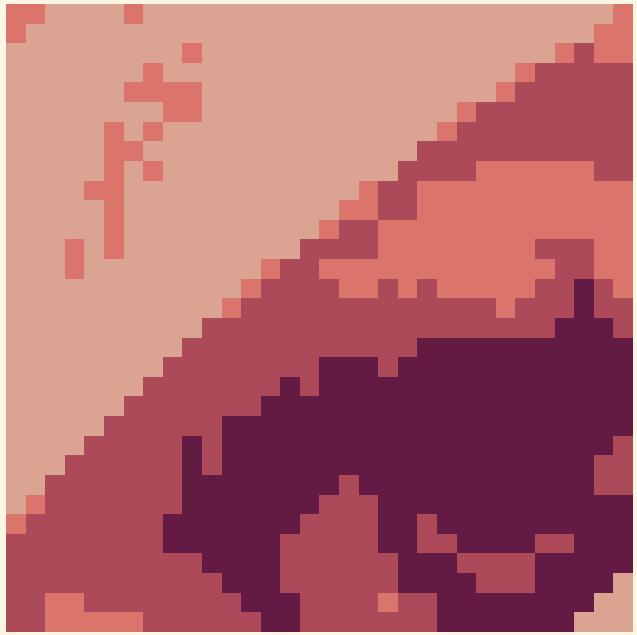
- Format *Portable Network Graphics*
- Méta-données liées aux couleurs
- Pas de filtrage perceptuel numérique
- Compression tenant compte de l'aspect 2D
- Support de la transparence et de la translucidité (transparence partielle)
- Support d'une palette de couleurs : les données colorimétriques de chaque point sont regroupées dans une palette unique, et on ne note que le numéro de couleur dans la palette pour chaque point.



Lena, 475 ko



Lena, palette 256c, 189 ko



Lena, palette 4c, 50 ko

Les images vectorielles

- Les images vectorielles sont stockées sous forme de courbes mathématiques : points, traits, courbes.
- Courbes de Bézier : équations polynomiales de degré 2 ou 3, déterminées par des points d'ancrage (1+degré) qui « passe entre les points »
  `http://en.wikipedia.org/wiki/Bezier_curve`
- Précision arbitraire
- Gestion du texte
- Adapté aux images synthétiques (dessins avec aplats)

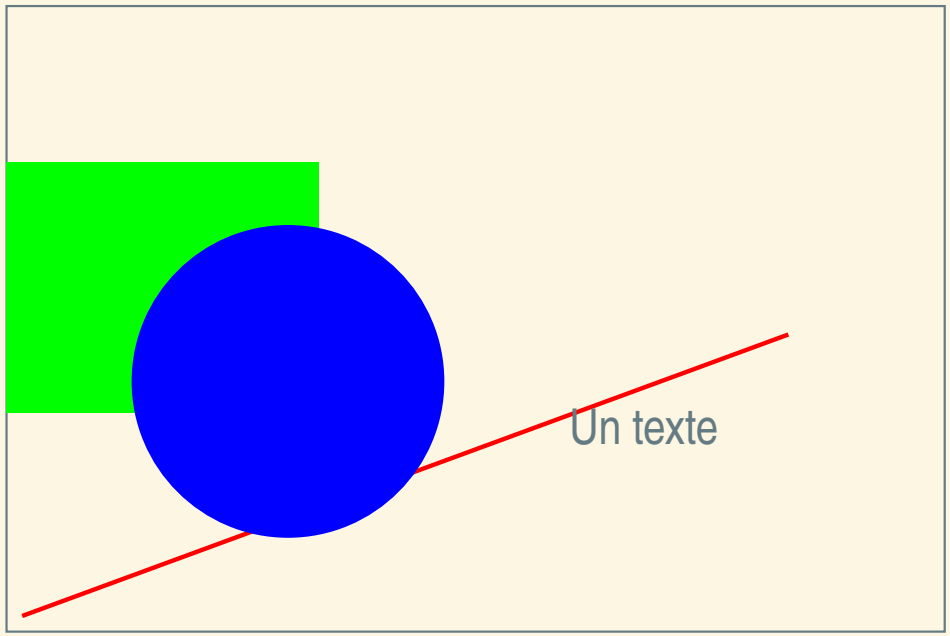
Tigre, zoom ×1

Tigre, zoom ×64

Tigre, zoom ×4096

Le format SVG

- Format en mode texte qui décrit des images vectorielles.
- Des balises écrites avec < et > délimitent les éléments
- On décrit les courbes par leurs coordonnées
- Métadonnées en XML possibles
- Visualisables directement dans les navigateurs

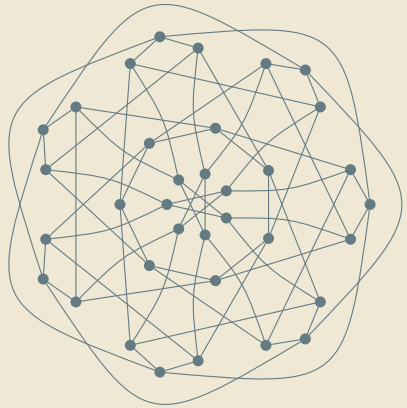



```
1  <?xml version="1.0" encoding="utf—8"?>
2  <svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="300" height="200">
3     <title>Exemple simple de figure SVG</title>
4     <desc>Cette figure comporte un rectangle,
5       un segment de droite et un cercle.</desc>
6
7     <rect width="100" height="80" x="0" y="70" fill="green" />
8     <line x1="5" y1="5" x2="250" y2="95" stroke="red" />
9     <circle cx="90" cy="80" r="50" fill="blue" />
10       <text x="180" y="60">Un texte</text>
11 </svg>
```

Exercices

Choix de format d'image

Q63 Voici quatre images. Imaginez le format le plus adapté à chacune d'entre elles. Expliquez votre choix.



Palette

Q64 Une image 1000×1000 utilise 3 octets pour décrire la couleur de chaque pixel. Calculez la taille occupée par les données de cette image en PNG.

Q65 Cette image n'a que 256 couleurs au total. On peut utiliser une palette de couleurs. Calculez la taille de la palette et la taille des données de l'image utilisant la palette.

Plan

Qu'est-ce qu'une couleur ?

- Réaction du cerveau à l'intensité des longueurs d'ondes de la lumière
- Certaines longueurs d'onde ne sont pas visibles
- Le mélange de longueurs d'onde est vu comme une autre couleur.
- On n'obtient certaines couleurs que par mélange (rose, marron)
- On distingue la couleur d'une source lumineuse, et la couleur d'un objet éclairé.

Un objet absorbe une partie de la lumière et recrache le reste. Par exemple, la chlorophylle absorbe essentiellement tout sauf le vert.

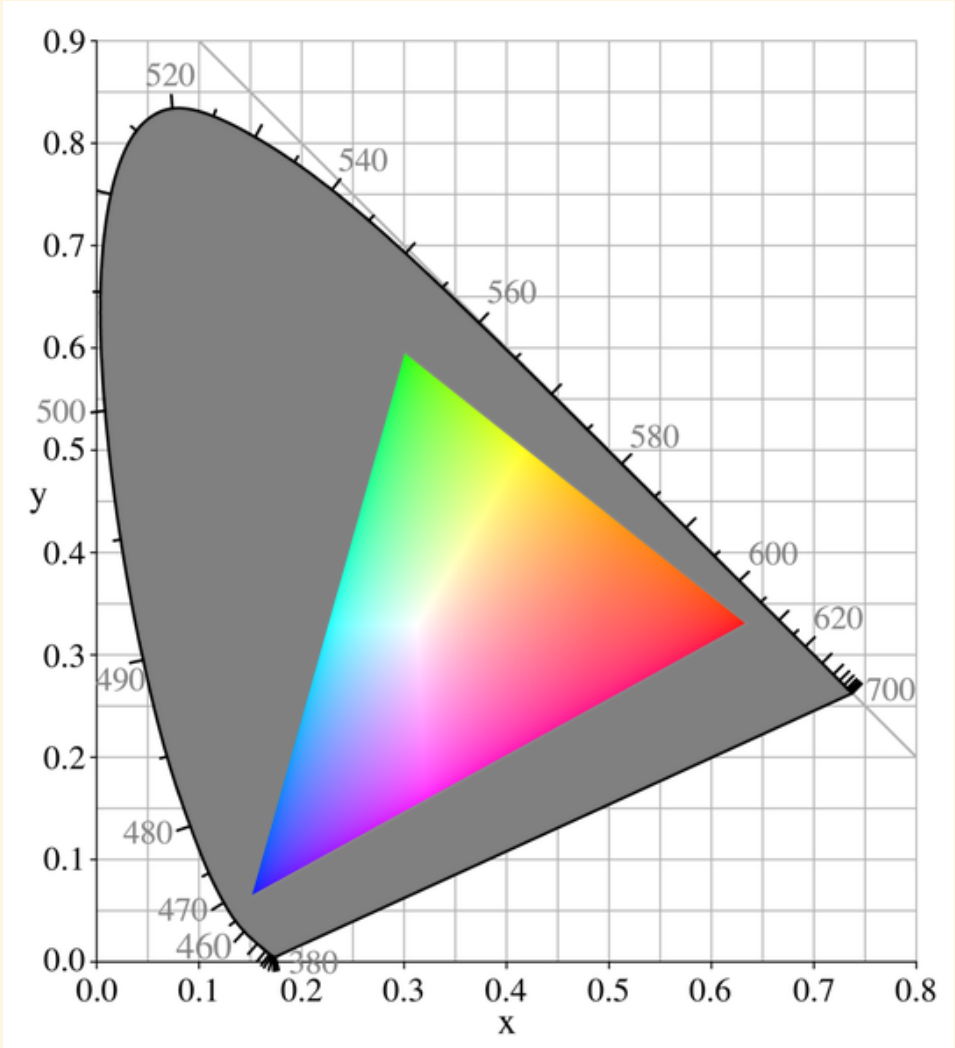
- Les couleurs d'une source s'additionnent : synthèse additive
- Les couleurs de pigments se masquent mutuellement : synthèse soustractive

Le gamut : teinte et saturation

- Une couleur peut être définie par sa teinte (ou ton), sa saturation (intensité de la teinte) et sa luminosité.
- Le gamut représente l'étendue des couleurs qui peuvent être reproduites par un moniteur ou une imprimante à luminosité fixée
- Le polygone représente les couleurs que l'on peut reproduire ; les extrémités sont les tons des couleurs de base que l'on mélange.

Certaines couleurs ne peuvent pas être obtenues. On perd de l'information lorsqu'on passe d'un système à un autre.

On utilise des profils couleurs ICC pour représenter le gamut et assurer les meilleures conversions possibles.



Le gamma : luminance

- La luminance est l'intensité de la couleur produite
- Le facteur γ caractérise la réponse lumineuse au stimulus électrique : $I = kV^{\gamma}$.

C'est donc un coefficient d'une réponse exponentielle.

Intensité	0%	39%	52%	61%	69%	75%	81%	86%	91%	95%	100%
Codage	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%

Signification : lorsque le signal d'entrée est de 10% de la puissance maximale, la perception du gris est de 39% environ. On compense donc le signal entré par le γ inverse pour donner l'impression d'une progression linéaire.

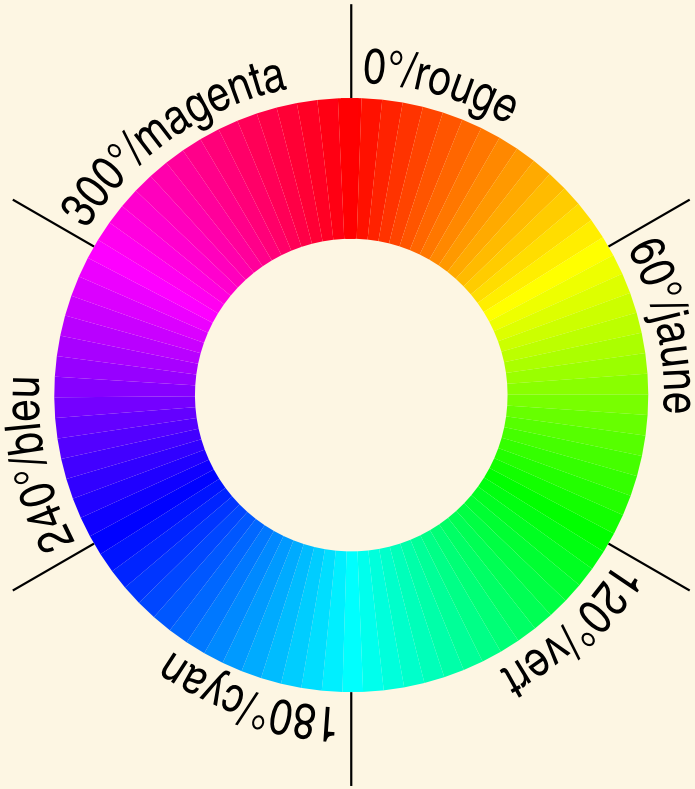
- Gamma normalisé des moniteurs : 2,5

Mais... tous les moniteurs n'ont pas le même γ.

- Nécessité de corriger la correction (on fait le produit des γ).

Images RVB (RGB en anglais)

- Le standard est de recomposer la couleur par synthèse additive de 3 couleurs.
- On mesure les couleurs par l'intensité de chacune des couleurs primaires : rouge, vert et bleu
- On obtient des couleurs par mélange de différentes intensités de RVB
- Un système équivalent permet de désigner les couleurs par teinte, saturation et luminance relative (valeur) : TSV (en anglais HSB, Hue/Saturation/Brightness).
- On utilise très souvent un octet d'information pour chaque composante.
- Notation usuelle : #RRVVBB (avec chaque paire de lettre qui est un octet noté en hexadécimal).



La roue des couleurs permet d'identifier la teinte à saturation maximale.

Exercices

Décomposition de couleurs

Donnez des composantes couleur plausibles RGB des couleurs suivantes. Utilisez la notation HTML.

- Rouge, vert, bleu
- Cyan, magenta, jaune
- Blanc, noir
- Gris 50%
- Marron foncé, rose pâle, orange vif

Scanner

Un scanner scanne en RGB à une résolution de 1200 points par pouce (dans les deux directions). Pour simplifier, on considérera qu'il y a une surface de 10 pouces $\times$ 6 pouces scannable. Chaque couleur est scannée en 12 bits. Quelle est la quantité d'information résultant de chaque scan ?

Images CMJN et polychromes

- L'impression utilise un standard de synthèse soustractive
- Deux pigments ensemble absorbent tous les deux la lumière et dans l'absolu le mélange complet fait du noir.

Le noir par addition n'est pas assez noir, on utilise donc une encre noire pure.

- Utilisation des couleurs complémentaires cyan, magenta et jaune
- Standard de l'impression : la *quadrichromie* CMJN (CMYK en anglais).

Certaines teintes ne sont pas possibles : rose vif, oranges vifs.

- Possibilité d'impression pentachrome ou hexachrome

possibilité de faire des couleurs garanties avec une encre par couleur sans mélanges : gamme Pantone® ou Focoltone®.

Exercices

Conversion HTML-CMJ

La trichromie consiste à n'utiliser que trois couleurs et faire le noire par mélange des autres couleurs. Dans ce cas, la formule est simple : la proportion d'une encre est 100% - la proportion de la couleur complémentaire.
Convertissez la couleur suivante en CM : #FA0140. Quel genre de teinte est-ce ? Est-elle très saturée ?

Vitesse d'impression

Une imprimante en quadrichromie est capable d'imprimer 6 pages par minutes, en 1200 points par pouce en mode RVB 8 bits par composante. Pour simplifier, on considérera qu'il y a une surface de 10 pouces × 6 pouces imprimable. Quelle est la quantité d'information qu'on doit fournir à l'imprimante pour une page ? Pour une minute d'impression ?

Plan

Qu'est-ce qu'un son ?

Un son est une vibration de l'air transportant un signal. C'est aussi le signal véhiculé par vibration.
Le son est donc numérisé comme vu au chapitre 1. Il est caractérisé par son spectre de fréquence instantané (les sons purs et périodiques qui le constituent à un moment donné). Un son est venu, à un instant donné, comme la somme de plusieurs sons « purs ».
La reproduction du son se fait en reproduisant une vibration qui a les mêmes caractéristiques fréquentielles. Les données audio sont donc des variations de pression (ou plutôt d'intensité électrique dans les capteurs et émetteurs).

Les formats

- Il faut distinguer les formats de fichiers des codecs (méthode de compression et filtrage des données).
- On distingue trois types de formats :
  1. Des formats non compressés qui rajoutent quelques méta-données (ou pas) à des données brutes (WAV, PCM, AIFF)
  2. Des formats compressés qui utilisent un algorithme (le *codec*) pour compresser et éventuellement réduire la quantité d'information avec une perte acceptable de qualité (FLAC, MP3, AAC, OGG)
  3. Des formats synthétiques qui contiennent des données d'instruments pour reproduire de la musique à base d'une partition (MIDI, SID)

Dans le domaine de la musique, le codec sert rarement à plusieurs formats (aucun obstacle théorique) à part PCM (pas de compression)

Caractéristiques

Le débit d'information va dépendre :

- Du nombre de voies (émetteurs indépendants pour reconstituer l'aspect spatial)
- Des fréquences reproduites (théorème d'échantillonage)
- De la quantification désirée (en nombre de bits)

Exercices

Compression audio MP3

Le codec MP3 permet de compresser le signal sonore dans une grande variété de débits finaux (après compression), le plus commun étant 128 kb/s. La fréquence d'échantillonage est quasi-toujours 44,1 kHz. **Calculatrice autorisée**.

Q66 Quel est le débit non compressé pour de l'audio stéréo 16 bits ?

Q67 Quel est le taux de compression du format MP3 le plus classique (débit final 128 kb/s) ?

Q68 Et avec le format plus généreux à 320 kb/s au final ?

Plan

Qu'est-ce qu'un film ?

- Un film est toute sorte d'image animée synchronisée ou non avec du son ou du texte.

Ils représentent plus de la moitié du trafic nord-américain sur internet.

- Les images successives s'appellent des trames (anglais *frames*).
- La synchronisation avec le son doit être précise et résistante aux erreurs.

Les containers et les codecs

- Comme pour l'audio, on distingue les formats (AVI, MP4, MPEG, MOV, MKS) des codecs (DIVX, x264, Theora, FFMPEG, Sorenson)
- Un certain nombre de formats n'acceptent qu'un nombre restreint de codecs vidéos ou audios (MP4 par exemple).
- Le processus administratif de normalisation pèse très lourd, car les fabriquants doivent faire du matériel conforme
- Les DRM sont des protections rajoutées qui empêchent dans certaines (nombreuses) circonstances d'accéder aux données. Elles sont dépendantes d'une inviolabilité du matériel et du logiciel.
- Citons aussi les GIF animés (et APNG) qui sont des formats d'images permettant une animation simple
- Ces formats peuvent contenir des méta-données plus ou moins riches (titre, auteurs, DRM,…).

L'encodage

- La phase d'encodage d'une vidéo ou d'un fichier audio consiste à repérer les similarités entre plusieurs trames successives.
- On peut par exemple décider s'il vaut mieux décrire les différences ou envoyer une nouvelle image.
- Deux images très similaires peuvent être compressées par exemple en faisant un XOR binaire entre les deux et en compression RLE après.
- Les codecs font aussi du filtrage perceptuel pour réduire la quantité de données. La compression peut-être plus agressive lorsque l'autre méthode donne de mauvais résultats.
- Il est important de garder des trames complètes périodiquement pour gérer les erreurs.
- Encodage souvent en deux passes (estimation des débits binaires voulus, puis calcul définitif)
- Actuellement, l'une des opérations les plus coûteuses en temps de calcul.
- Certains formats spéciaux multi-débits permettent de s'adapter à la vitesse de communication de deux ordinateurs pour proposer la meilleure qualité *(streaming)*.

Annexe

Le jeu du fakir
La table ASCII

Le jeu du fakir (1)

Est-ce que le nombre choisi est impair ?

Le jeu du fakir (2)

Est-ce que le nombre choisi fait partie de ceux-ci :

2	3	6	7	10	11	14	15
18	19	22	23	26	27	30	31
34	35	38	39	42	43	46	47
50	51	54	55	58	59	62	63
66	67	70	71	74	75	78	79
82	83	86	87	90	91	94	95
98	99						

Le jeu du fakir (3)

Est-ce que le nombre choisi fait partie de ceux-ci :

4	5	6	7	12	13	14	15
20	21	22	23	28	29	30	31
36	37	38	39	44	45	46	47
52	53	54	55	60	61	62	63
68	69	70	71	76	77	78	79
84	85	86	87	92	93	94	95
100							

Le jeu du fakir (4)

Est-ce que le nombre choisi fait partie de ceux-ci :

8	9	10	11	12	13	14	15
24	25	26	27	28	29	30	31
40	41	42	43	44	45	46	47
56	57	58	59	60	61	62	63
72	73	74	75	76	77	78	79
88	89	90	91	92	93	94	95

Le jeu du fakir (5)

Est-ce que le nombre choisi fait partie de ceux-ci :

16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63
80	81	82	83	84	85	86	87
88	89	90	91	92	93	94	95

Le jeu du fakir (6)

Est-ce que le nombre choisi fait partie de ceux-ci :

32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63
96	97	98	99	100			

Le jeu du fakir (7)

Est-ce que le nombre choisi est strictement plus grand que 63 ?

Le nombre est... Retour

Le jeu du fakir (7)

Est-ce que le nombre choisi est strictement plus grand que 63 ?
Le nombre est... Retour

La table ASCII

- 32 caractères de « contrôle », 96 « affichables » ;
- Unicode, ISO-8859 compatibles avec ASCII.

00 NUL	01 SOH	02 STX	03 ETX	04 EOT	05 ENQ	06 ACK	07 BEL
08 BS	09 HT	0A NL	0B VT	0C NP	0D CR	0E SO	0F SI
10 DLE	11 DC1	12 DC2	13 DC3	14 DC4	15 NAK	16 SYN	17 ETB
18 CAN	19 EM	1A SUB	1B ESC	1C FS	1D GS	1E RS	1F US
20 SP	21 !	22 "	23 #	24 $	25 %	26 &	27 ’
28 (	29)	2A *	2B +	2C ,	2D -	2E .	2F /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5A Z	5B [	5C \	5D]	5E ^	5F _
60 ‘	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7A z	7B {	7C |	7D }	7E ~	7F DEL

Retour