

TD2: Encodage des nombres

MP2I Lycée Pierre de Fermat
guillaume.rousseau@ac-toulouse.fr

La calculatrice est **interdite** pour ce TD !

Exercice 1.

Q1. Combien valent les nombres suivants ?

- a) $\overline{10011}^2$ b) $\overline{30}^5$ c) $\overline{66}^7$ d) $\overline{01111111}^2$ e) $\overline{13}^{11}$

Q2. Calculez la décomposition de n en base B pour :

- a) $B = 2, n = 37$ b) $B = 7, n = 82$ c) $B = 16, n = 35$

Q3. Soit B une base quelconque. Écrivez n en base B pour :

- a) $n = B^k$ avec $k \in \mathbb{N}$ b) $n = B^k - 1$ avec $k \in \mathbb{N}$

Q4. Écrire les nombres suivant en base 4 :

- a) 11 b) 11×4
c) 11×4^2 d) 11×4^k pour $k \in \mathbb{N}$
e) 73 f) $\lfloor \frac{73}{4} \rfloor$ g) $\lfloor \frac{73}{4^2} \rfloor$
h) $\lfloor \frac{73}{4^3} \rfloor$ i) $73 \bmod 4$.
j) 86 k) $\lfloor \frac{86}{4} \rfloor$ l) $\lfloor \frac{86}{4^2} \rfloor$
m) $\lfloor \frac{86}{4^3} \rfloor$ n) $86 \bmod 4$.

Q5. Soit B une base, et $u_{l-1} \dots u_0$ l'écriture en base B d'un entier $n \in \mathbb{N}$. Donner l'écriture en base B de :

- a) $n \bmod B$
b) le quotient euclidien de n par B , c'est à dire $\lfloor \frac{n}{B} \rfloor$
c) le quotient euclidien de n par B^2 , c'est à dire $\lfloor \frac{n}{B^2} \rfloor$
d) $\lfloor \frac{n}{B^k} \rfloor$, pour $k \in \mathbb{N}$.
e) $n \bmod B^k$, pour $k \in \mathbb{N}$.

Q6. Étant donné B une base, $n \in \mathbb{N}$ et $k \in \mathbb{N}$, donner une formule mathématique permettant d'obtenir le k -ème chiffre de n en base B (celui associé à B^k).

A retenir En base B , les opérations triviales à calculer sont les multiplications et divisions par des puissances de B , qui reviennent simplement à décaler les chiffres. Lorsque l'on pense en base B , il faut visualiser le nombre comme une suite de chiffres que l'on peut manipuler via ces opérations, et toujours garder en tête qu'une base B quelconque se comporte presque exactement comme la base 10 !

Exercice 2.

Hexadécimal

En plus de la base 2 et de la base 10, une base importante en informatique est la base 16, ou **hexadécimale**. En effet, $16 = 2^4$, et on peut montrer que cela permet de compacter la base 2 : un chiffre en base 16 correspondra à une suite d'exactly 4 chiffres en base 2.

Pour écrire des nombres en base 16, il nous faut 16 chiffres. En plus des chiffres 0, 1, ..., 9, on rajoute les lettres a, b, ..., f afin d'avoir 16 symboles distincts.

On peut donc étendre la table des nombres de 0 à 15 vue plus haut pour y rajouter les chiffres hexadécimaux :

Dec	Bin	Hex	Dec	Bin	Hex	Dec	Bin	Hex	Dec	Bin	Hex
0	0000	0	4	0100	4	8	1000	8	12	1100	c
1	0001	1	5	0101	5	9	1001	9	13	1101	d
2	0010	2	6	0110	6	10	1010	a	14	1110	e
3	0011	3	7	0111	7	11	1011	b	15	1111	f

Par exemple $\overline{b2e}^{16}$ vaut $11 \times 16^2 + 2 \times 16 + 14 \times 1$.

Q1. Donner l'écriture décimale de $\overline{5a}^{16}$.

Q2. Donner l'écriture hexadécimale de 172.

Soit $n = \overline{1101\ 0010\ 1111\ 0100}^2$. Pour décomposer n en base 16 à partir de la base 2, il est possible de repasser par la base 10, plus familière, puis chercher la décomposition en base 16. Cependant, on peut faire la traduction directement : on découpe l'écriture en base 2 par paquets de 4, puis l'on remplace chaque paquet par le chiffre en base 16 correspondant :

$$\begin{aligned} n &= \overline{1101} \ \overline{0010} \ \overline{1111} \ \overline{0100} \\ &= \quad d \quad \quad 2 \quad \quad f \quad \quad 4 \end{aligned}$$

Donc, $n = \overline{d2f8}^{16}$. On notera aussi **0xd2f8**, en préfixant le nombre par **0x**, pour signifier que l'on parle en hexadécimal. Cette notation peut être utilisée en C :

```
1 int x = 0xd2f8;
```

Être à l'aise avec l'hexadécimal est très utile en informatique, car les adresses mémoires, les couleurs, et d'autres valeurs sont souvent affichées en base 16.

Q3. Combien de chiffres hexadécimaux faut-il pour représenter un octet ? Et pour un entier 32 bits ?

Q4. Remplir le tableau suivant :

Dec	15	735	493		
Bin	1111	0010 1101 1111		1010 0110 0001	
Hex	f	2df			d4c

Exercice 3.*Lecture en base B*

On considère l'algorithme suivant de lecture en base B :

Algorithme 1 : Lecture depuis une base

Entrée(s) : B une base, $a_{l-1} \dots a_0$ un mot sur l'alphabet $\llbracket 0, B-1 \rrbracket$

Sortie(s) : $n \in \mathbb{N}^*$ tel que $n = \overline{a_{l-1} \dots a_0}^B$

```

1  $n \leftarrow 0$ ;
2  $i \leftarrow 0$ ;
  // Invariant: ...
3 tant que  $i < l$  faire
4    $n \leftarrow n + a_i B^i$ ;
5    $i \leftarrow i + 1$ ;
6 retourner  $n$ 
```

Q1. Montrez la correction de cet algorithme à l'aide d'un bon invariant de boucle.

Indication : Exprimez n en base B à chaque étape de l'algorithme.

Q2. En fonction du nombre de chiffres l lus, quel est le nombre total de multiplications effectuées par l'algorithme ? On considèrera que le calcul de puissance est fait naïvement, i.e. que calculer x^k demande $k-1$ multiplications.

Q3. En introduisant une variable p vérifiant l'invariant " $p = B^i$ ", réduire le nombre de multiplications que fait l'algorithme.

On s'intéresse maintenant à un deuxième algorithme, encore moins coûteux en nombre de multiplications, appelé la **méthode de Horner**. Cette méthode permet plus généralement d'évaluer la valeur d'un polynôme en un point, on l'applique ici au problème similaire de la lecture d'un nombre en base B :

Algorithme 2 : Horner

Entrée(s) : B une base, $a_{l-1} \dots a_0$ un mot sur l'alphabet $\llbracket 0, B-1 \rrbracket$

Sortie(s) : $n \in \mathbb{N}^*$ tel que $n = \overline{a_{l-1} \dots a_0}^B$

```

1  $n \leftarrow 0$ ;
2  $i \leftarrow l-1$ ;
  // Invariant: ...
3 tant que  $i \geq 0$  faire
4    $n \leftarrow Bn + a_i$ ;
5    $i \leftarrow i-1$ ;
6 retourner  $n$ 
```

Q4. Montrer la correction de cet algorithme. Quel est le nombre total de multiplications qu'il effectue en fonction de la longueur l du mot en entrée ?

Exercice 4.*Addition des entiers non-signés*

Dans cet exercice, on considère des entiers non-signés sur 8 bits.

Effectuer les additions suivantes **en binaire**, et dire lorsqu'elles causent un dépassement d'entier :

a) $95 + 132$

b) $12 + 167$

c) $7 + 64$

d) $184 + 72$

Exercice 5.*Opérations bit à bit*

Lorsque l'on considère des entiers écrits en base 2, on doit les voir à la fois comme des entiers sur lesquels on peut effectuer des opérations arithmétiques, et comme des vecteurs booléens sur lesquels on peut effectuer des opérations logiques bit-à-bit. Par exemple, l'octet 01001101 représente l'entier non-signé 77 ET est une suite de 8 valeurs booléennes.

On rappelle les trois opérateurs booléens bit-à-bit : $\&$, $|$ et $\sim$. Ces trois opérateurs font respectivement un ET, un OU et un NON bit à bit. Par exemple sur 4 bits :

$$1001\&1100 = 1000, 1001|1100 = 1101 \text{ et } \sim 1001 = 0110$$

On fixe $l \in \mathbb{N}$ un nombre de bits sur lequel les nombres sont écrits. On note $\mathbb{N}_l$ l'ensemble des entiers représentables sur l bits non-signés. Les opérateurs bit-à-bit agissent techniquement sur les mots binaires, i.e. sur les suites de 0 et de 1, mais on étend assez naturellement leur action au entiers de $\mathbb{N}_l$: pour n et m deux entiers ayant pour écritures binaires respectives $a_{l-1} \dots a_0$ et $b_{l-1} \dots b_0$, on aura :

$$n\&m = \overline{(a_{l-1} \dots a_0)\&(b_{l-1} \dots b_0)}^2$$

et idem pour $|$ et $\sim$. Autrement dit, pour calculer $n\&m$, on écrit n et m en binaire sur l bits, on calcule le ET bit à bit, et l'on interprète le résultat comme un entier en base 2.

Q1. En se plaçant sur $l = 8$ bits, donner les résultats des calculs suivants :

- | | | |
|--------------|------------|---------------------|
| 1. $100\&52$ | 3. $95 44$ | 5. ~ 216 |
| 2. $31\&64$ | 4. $31 64$ | 6. $(\sim 100) 100$ |

Q2. On note $\&\&$, $||$ et $!$ les opérateurs logiques habituels, définis sur $\{0,1\}$ uniquement. Pour $x, y \in \mathbb{N}_l$ admettant comme écritures binaires respectives $x_{l-1} \dots x_0$ et $y_{l-1} \dots y_0$, donner des formules pour exprimer $x\&y$, $x|y$ et $!x$.

Q3. Pour $x \in \mathbb{N}_l$, que vaut $x|\sim x$? Et $x\&\sim x$? (une preuve formelle est attendue).

Q4. Montrez que pour tout $x, y \in \mathbb{N}_l$, $x\&y \leq \min(x, y)$. Sur 8 bits, trouvez un cas où l'inégalité est stricte, et un où il y a égalité mais où $x \neq y$.

On rappelle également les opérateurs de **décalage** : $>>$ et $<<$:

$$\forall u = a_{l-1} \dots a_0 \in \{0,1\}^*, \forall k \in \mathbb{N}, u << k = a_{l-1-k} \dots a_0 00 \dots 0$$

avec k 0 ajoutés à droite de u et k bits perdus à gauche, et

$$\forall u = a_{l-1} \dots a_0 \in \{0,1\}^*, \forall k \in \mathbb{N}, u >> k = 00 \dots 0 a_{l-1} \dots a_k$$

avec k 0 ajoutés à gauche de u et k bits perdus à droite. On étend l'action de ces opérateurs aux entiers de $\mathbb{N}_l$.

Q5. Sur 8 bits, donner le résultat des calculs suivants :

- | | | | |
|-------------|--------------|--------------|--------------|
| 1. $7 << 3$ | 2. $73 << 5$ | 3. $32 >> 3$ | 4. $55 >> 2$ |
|-------------|--------------|--------------|--------------|

Q6. Pour $a \in \mathbb{N}_l$ un entier et $k \in \mathbb{N}$, exprimer $a >> k$ et $a << k$ en fonction de a , k et l .

Pour $x \in \mathbb{N}_l$, admettant $x_{l-1} \dots x_0$ comme écriture binaire, on appelle " i -ème bit de x " le bit $x_i \in \{0,1\}$.

On rappelle que les opérateurs bit-à-bit et de décalage existent en C. Les questions suivantes doivent être faites sans utiliser de boucles, ni de fonctions récursives.

Q7. Écrire une fonction `int bit(int n, int i)` renvoyant le i -ème bit de n .

Q8. Écrire une fonction `int set(int n, int i)` mettant le i -ème bit de n à 1 et renvoyant le résultat.

Q9. Écrire une fonction `int toggle(int n, int i)` inversant le i -ème bit de n et renvoyant le résultat.

Par exemple, `toggle(21, 3)` doit valoir 29 car $21 = \overline{10101}^2$ et $\overline{11101}^2 = 29$. Inversement, `toggle(29, 3)` vaut 21.

Exercice 6.

Ou exclusif

Le but de cet exercice est d'écrire un algorithme d'addition en binaire n'utilisant que des opérations booléennes. On introduit un nouvel opérateur booléen : le ***OU exclusif***, aussi appelé ***xor***, noté $\oplus$ (Cf Fig. 1).

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

FIGURE 1 – Table de vérité du ***xor*** logique

Q1. Rappelez les tables de valeurs des opérateurs booléens `&&` `||` et `!`.

Q2. Pour a et b deux booléens, exprimez $a \oplus b$ en n'utilisant que les opérateurs `&&` `||` et `!`.

Q3. Montrer que le xor est associatif, c'est à dire que pour $x, y, z \in \{0, 1\}$, on a $(x \oplus y) \oplus z = x \oplus (y \oplus z)$.

En C, il n'existe pas de symbole pour le xor logique. En revanche, on peut faire des xor bit à bit avec l'accent circonflexe `^`.

Q4. On se place sur 8 bits. Calculer :

a) $9 \wedge 20$

b) $91 \wedge 107$

c) $255 \wedge 183$

Q5. La question précédente change t-elle si l'on se place sur 16 ou 32 bits ?

Q6. Sur l bits, pour n un entier non-signé représentable, que vaut :

a) $n \wedge 0$?

b) $n \wedge (2^l - 1)$?

c) $n \wedge n$?

Q7. Pour n, m représentables sur l bits, à quelle condition a-t-on que $n \wedge m = n + m$?

Q8. Que fait le code suivant (on suppose que x et y ont été définies plus haut) :

```
1 x = x^y;
2 y = x^y;
3 x = x^y;
```

Exercice 7.*Flottants*

Pour $k, l \in \mathbb{N}^*$, on rappelle que l'encodage flottant normalisé à k bits d'exposants et l bits de mantisse consiste à interpréter la suite de bits $se_{k-1} \dots e_0 m_1 m_2 \dots m_l$ comme le réel

$$(-1)^s 2^{\overline{e_{k-1} \dots e_0} 2 - (2^{k-1} - 1)} \left(1 + \sum_{i=1}^l \frac{m_i}{2^i}\right)$$

Q1. Pour $k = 2$ et $l = 2$, énumérer tous les flottants positifs. On traitera tous les flottants comme normalisés.

On s'intéresse à une manière de déterminer l'encodage d'un réel donné, c'est à dire à partir de $x \in \mathbb{R}$ quelconque, de déterminer les bits $s, e_{k-1} \dots e_0$ et $m_1 \dots m_l$ tels que

$$x \approx (-1)^s 2^{\overline{e_{k-1} \dots e_0} 2 - (2^{k-1} - 1)} \left(1 + \sum_{i=1}^l \frac{m_i}{2^i}\right)$$

Notons qu'il n'y aura pas égalité a priori, car nous avons vu en cours que tous les réels ne sont pas représentables de manière exacte. La méthode présentée dans cet exercice donnera le flottant inférieur ou égal à x le plus grand possible.

Q2. Expliquer comment déterminer le bit de signe s en fonction de x .

Dans la suite, on suppose $x \geq 0$. L'étape suivante est de trouver l'exposant. Pour cela, étudions l'effet de la mantisse.

Q3. Montrer que

$$1 \leq 1 + \sum_{i=1}^l \frac{m_i}{2^i} < 2$$

Q4. En déduire une méthode permettant de trouver $e_{k-1} \dots e_0$. On pourra commencer par trouver $p \in \mathbb{Z}$ tel que $2^p \leq x < 2^{p+1}$.

Dans la suite, on suppose que $1 \leq x < 2$. On cherche $m_1 \dots m_l$ tels que :

- $1 + \sum_{i=1}^l \frac{m_i}{2^i} \leq x$;
- $1 + \sum_{i=1}^l \frac{m_i}{2^i}$ est le plus grand possible.

Q5. En multipliant par 2^l , réduire le problème à un calcul de partie entière, puis en déduire la méthode pour trouver $m_1 \dots m_l$.

Q6. Application (calculatrice autorisée) : pour $k = 4$ et $l = 4$, calculer l'encodage flottant des réels suivants, puis vérifier que les erreurs relatives sont inférieures à $\frac{1}{2^l}$:

- a) $\frac{-1}{3}$
- b) π
- c) 99

Exercice 8.*Encodage de jeux*

On veut encoder une grille de morpion par un entier non signé 32 bits. On veut que notre encodage nous permette de récupérer l'état de la grille facilement en manipulant les bits.

On suppose qu'on a deux joueurs : le joueur X et le joueur O. Une case peut contenir un X, un O ou rien. On assigne à chaque case un nombre entre 0 et 3 selon son contenu :

X	$\mapsto$	2
O	$\mapsto$	3
vide	$\mapsto$	0

On remarque que le nombre 1 ne correspond à rien. Ensuite, on numérote les 9 cases d'une grille comme suit :

0	1	2
3	4	5
6	7	8

Enfin, en notant a_i le nombre correspondant au contenu de la case i , on encode une grille G par :

$$\sum_{i=0}^8 a_i 4^i$$

Q1. Donnez les écritures en bases 2, 4, et 16 de l'encodage des grilles suivantes :

a)

b)

		O
X	X	O
		X

c)

O	X	O
X	X	O
	O	X

Q2. Si n est l'encodage d'une grille et $0 \leq i \leq 8$, exprimez à l'aide des opérateurs $\gg$ et $\&$ une formule calculant un booléen indiquant si la case i de la grille est remplie ou non.

Q3. Vérifier que l'encodage proposé tient sur 32 bits. Quel est le nombre minimal de bits nécessaire ?

Q4. Soit n l'encodage d'une grille, et $0 \leq i \leq 8$. On suppose que la case i de la grille encodée par n est remplie. Exprimez à l'aide des opérateurs $\gg$ et $\&$ une formule calculant un booléen indiquant si la case i de la grille contient un O ou un X.

Q5. Soit n l'encodage d'une grille, et $0 \leq i \leq 8$. On suppose que la case i de la grille encodée par n est vide. Exprimez à l'aide des opérateurs $\ll$ et $|$ une formule calculant l'encodage de la grille obtenue en rajoutant le symbole X dans la case i .

Q6. Idem pour le symbole O.

Q7. Expliquez, selon le même principe, comment on pourrait encoder une grille de Sudoku. De combien de bits aurait-on besoin ?