6ème3 DM n°1 : Autres Numérations

CORRECTION

1) La numération babylonienne

a) Le système de numération de la civilisation babylonienne[1] n'utilise que deux symboles : ∨ (valeur : 1) et < (valeur : 10).
On peut juxtaposer jusqu'à 9 fois le symbole ∨ et 5 fois le symbole <.
Ainsi 12 s'écrit <∨∨ (10+2) et 36 s'écrit <<<∨∨∨∨∨∨ (30+6).

☞ Comment écrit-on les nombres 2, 12, 24 et 59 en babylonien ?

2 s'écrit ∨∨ (2)
12 s'écrit <∨∨ (10+2)
24 s'écrit <<∨∨∨∨ (20+4)
59 s'écrit <<<<<∨∨∨∨∨∨∨∨∨ (50+9)
Remarque : ces nombres (jusqu'à 59 compris sont les *chiffres* du système de numération babylonien.

b) Ce système de numération est un *système de positions* : la valeur des symboles est multipliée par 60 en 2ème position, par 60×60 (=3600) en 3ème position, etc.
Ainsi 75 s'écrit sur deux positions ∨ ; <∨∨∨∨∨ (60+15) et 200 s'écrit ∨∨∨ ; << (3×60+20).

☞ Comment écrit-on les nombres 61, 3601, 100 et 2018 en babylonien ?

En babylonien, il y a un problème pour écrire certains nombres comme 61 ou 3601 car le zéro n'existe pas (il y a une forme de zéro en babylonien tardif, mais elle n'est pas systématiquement employée)
61 s'écrit 1;1, soit ∨;∨ car 61=1×60+1×1.
3601 s'écrit 1;0;1, soit ∨; ;∨ car 3601=1×3600+(0×60)+1×1 (l'absence de soixantaine nécessiterait un 0)
100 s'écrit 1;40, soit ∨;<<<< car 100=1×60+40×1.
2018 s'écrit 33;38, soit <<<∨∨∨;<<<∨∨∨∨∨∨∨∨ car 2018=1980+38=33×60+38×1.
Je n'ai pas actualisé ma question, mais 2022 s'écrirait 33;42, soit <<<∨∨∨;<<<<∨∨

c) Ce système a été conservé dans notre division du temps en heures, minutes et secondes.

☞ Expliquer cela en donnant un ou deux exemples.

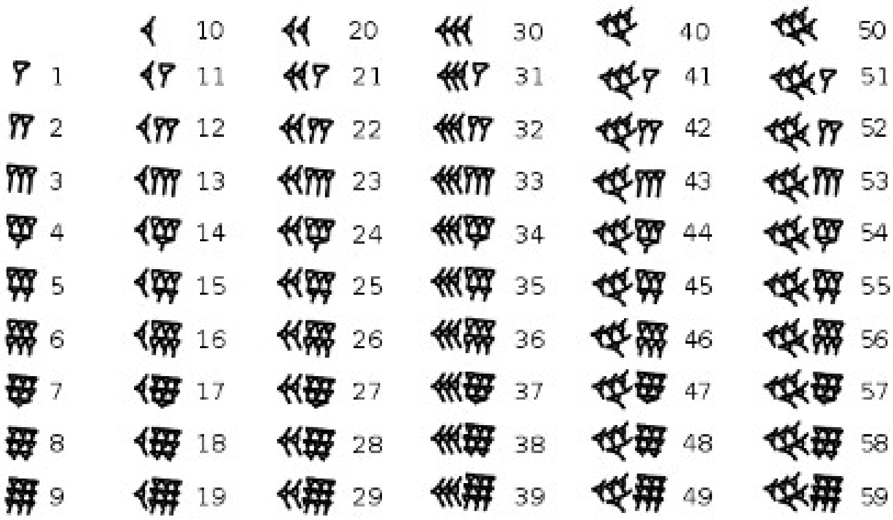
Dans notre notation du temps 1 heure vaut 60 minutes et 1 minute vaut 60 secondes.
Les durées en heures/minutes/secondes sont exprimées avec des nombres babyloniens car ils sont écrits comme eux en base 60. En babylonien, le nombre 1;32;45 correspond à 5565 unités tandis que, pour nous, une durée de 1h 32min 45s indique qu'il y a 1×3600+32×60+45×1=5565 secondes. Il s'agit de la même façon de décomposer les grands nombres. Si nous avons conservé l'antique notation babylonienne, c'est qu'elle a de nombreux avantages.

d) Conclusion

☞ Donner au moins un avantage et un inconvénient de ce système de numération ?

Inconvénients : un nombre babylonien est long à écrire, il faut compter les chevrons (on peut se tromper) car il n'y a que deux symboles. Comme il manque le zéro qui permet de distinguer 60 de 1 ou 61 de 3601. L'utilisation de deux symboles introduit des risques de confusion.
Avantages : Ce système de numération est très puissant pour un système antique car il permet (théoriquement) d'écrire des nombres très grands et très petits (décimaux proches de zéro). Le savant grec Archimède qui a calculé les premières décimales de π utilisait ce système. Soixante est le plus petit entier divisible à la fois par 2, par 3, par 4, par 5 et par 6 ce qui le rend plus intéressant que tous les nombres qui lui sont inférieurs.

Remarque : J'ai écrit les symboles en ligne pour nous simplifier la tâche, mais les babyloniens dessinaient leurs chiffres autrement. Voici, pour information, des écritures plus élégantes et plus authentiques des chiffres babyloniens.



1 La civilisation babylonienne de l'actuel Irak s'étend entre 3200 avant J.-C. et 500 avant J.-C.

1) La numération binaire

a) Les ordinateurs utilisent le *binaire* : deux chiffres (0 et 1) suffisent pour écrire tous les nombres.
On change de position à 2 (car il n'y a pas de chiffre 2), 4=2×2, 8=2×2×2, etc.
Ainsi les premiers nombres sont 1, 10 (2+0), 11 (2+1), 100 (4+0), 101 (4+1), 110 (4+2), 111 (4+2+1), ..., etc.
☞ Écrire les nombres de 1 à 20 en binaire.

décimal	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
binaire	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111	10000	10001	10010	10011	10100

b) Les nombres 1, 2, 2×2, 2×2×2, etc. sont appelés « puissances de 2 » ; on les note 2^0, 2^1, 2^2, 2^3, etc.
Les puissances de 2 donnent la valeur numérique des chiffres **1** de la notation binaire, selon leur position.
Ainsi 50 s'écrit **11001**0 car 50=**1**×2^5+**1**×2^4+(0×2^3+0×2^2)+**1**×2^1+(0×2^0)
☞ Écrire les dix premières puissances de 2 en notation décimale et en binaire (faire un tableau).

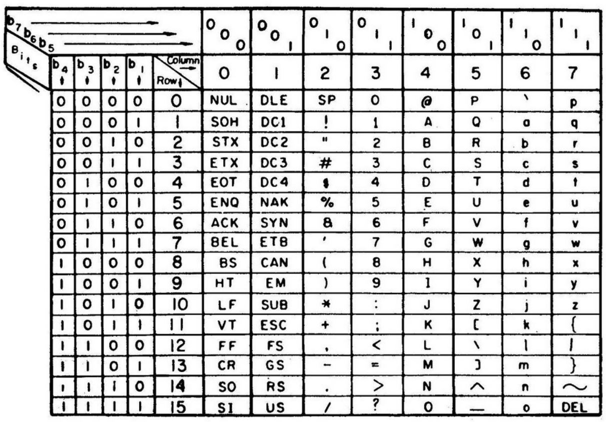
Nombre de facteurs 2	0	1	2	3	4	5	6	7	8	9	10
Notation	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Puissance de 2 (en décimal)	1	2	4	8	16	32	64	128	256	512	1024
Puissance de 2 (en binaire)	1	10	100	1000	10000	100000	1000000	10000000	100000000	100000000	1000000000

c) La notation binaire indique donc les puissances de 2 employées dans la décomposition d'un nombre.
Ainsi, comme 70=64+4+2=2^6+2^2+ 2^1 et que 64, on note ce nombre 1000110 en binaire.
☞ Comment écrit-on 61, 3601, 100 et 2022 en binaire ?
61=32+16+8+4+1 donc 61 s'écrit 111 101 en binaire.
3601=2048+1024+512+16+1 donc 3601 s'écrit 111 000 010 001 en binaire.
100=64+32+4 donc 100 s'écrit 1 100 100 en binaire.
2022=1024+512+256+128+64+32+4+2 donc 2022 s'écrit 111 111 001 10 en binaire.

d) En informatique, on appelle « octet », une séquence de 8 chiffres binaires (on dit 8 bits).
Ainsi, le nombre 91 qui s'écrit 01011011 en binaire tient sur un octet.
☞ Combien de nombres différents peut-on coder avec un octet ?
Avec 9 chiffres binaires, on écrit 100 000 000 (256) donc avec 8 chiffres binaires, on peut aller jusqu'à 11 111 111 (256–1=255). On peut donc coder 255 nombres non nuls différents. Avec le zéro, cela en fait 256. Avec un octet, on peut coder 255 nombres différents.
☞ Combien de nombres peut-on coder sur deux octets ?
Avec le même raisonnement, 16 chiffres binaires (2 octets=2×8chiffres binaires) codent un nombre de nombres égal à la seizième puissance de 2, notée 2^{16} et qui vaut 65 536. Avec deux octets, on peut coder 65 536 nombres différents.

e) Table ASCII
☞ Quelle fonction peut avoir le tableau ci-contre qui date de 1972 ? (Chercher éventuellement sur internet)
L'*American Standard Code for Information Interchange* (Code américain normalisé pour l'échange d'information), plus connu sous l'acronyme ASCII (ou USASCII comme il est marqué sur le tableau avec une pointe de chauvinisme typiquement américain (des USA)) est une norme de codage des caractères en informatique. Cette table associe à chacune des 128 combinaisons de 7 caractères binaires (**bi**nary dig**its**) une signification.

USASCII code chart



$b_7b_6b_5$ →					000	001	010	011	100	101	110	111
b_4	b_3	b_2	b_1	Row↓ Column→	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Exemple : au carrefour de la colonne 5 (**101** en binaire) et de la ligne 6 (0110 en binaire), on trouve le caractère « V » ce qui signifie que ce caractère est codé par le byte **101**0110 ; ce que l'on a traduit par l'octet 0**101**0110, en ajoutant un 0 à gauche. J'ai ajouté, pour faciliter la lecture, une indication des chiffres de la colonne **en gras**.

Pour information :
les bits d'un octet sont numérotés, de droite à gauche : b_1 (unités) à b_7 (×64) ;
j'ai ajouté un bit nul en position b_8 pour faire un octet.

Je m'en suis servi pour écrire le message ci-dessous.

☞ Décoder ce message et répondez-moi en binaire.

01010001-01110101-01100101-01101100-00100000-01100101-01110011-01110100-00100000-01110100-01101111-01101110-00100000-01101110-01101111-01101101-00111111

0**101**0001-0**111**0101-0**110**0101-0**110**1100-0**010**0000-0**110**0101-0**111**0011-0**111**0100-
Q - u - e - l - SP - e - s - t -
0**010**0000-0**111**0100-0**110**1111-0**110**1110-0**010**0000-0**110**1110-0**110**1111-0**110**1101-0**011**1111
SP - t - o - n - SP - n - o - m - ?

La phrase décodée dit : Quel est ton nom ?

Si je devais répondre à cette question j'écrirais : « Mon nom est Philippe Moutou. »

Cette phrase se code ainsi :

0**100**1101-0**110**1111-0**110**1110-0**010**0000-0**110**1110-0**110**1111-0**110**1101-0**010**0000-0**110**0101-0**111**0011-
M - o - n - SP - n - o - m - SP - e - s -
0**111**0100-0**010**0000-0**101**0000-0**110**1000-0**110**1001-0**110**1100- 0**110**1001-0**111**0000-0**111**0000-0**110**0101-
t - SP - P - h - i - l - i - p - p - e -
0**100**1101-0**110**1111-0**111**0101-0**111**0100-0**110**1111-0**111**0101-0**010**1110
M - o - u - t - o - u - .

Cela s'écrirait donc simplement, en enlevant le décodage et la graisse des caractères :

01001101-01101111-01101110-00100000-01101110-01101111-01101101-00100000-01100101-01110011-01110100-00100000-01010000-01101000-01101001-01101100-01101001-01110000-01110000-01100101-01001101-01101111-01110101-01110100-01101111-01110101-00101110

L'ordinateur ne sépare pas les octets d'un texte. C'est pour nous les humains que j'ai ajouté les traits d'union. L'ordinateur recevrait, lui, le flux binaire suivant, nettement moins lisible pour nous :

0100110101101111011011100010000001101110011011110110110100100000011001010111001101110100
0010000001010000011010000110100101101100011010010111000001110000011001010100110101101111
0111010101110100011011110111010100101110