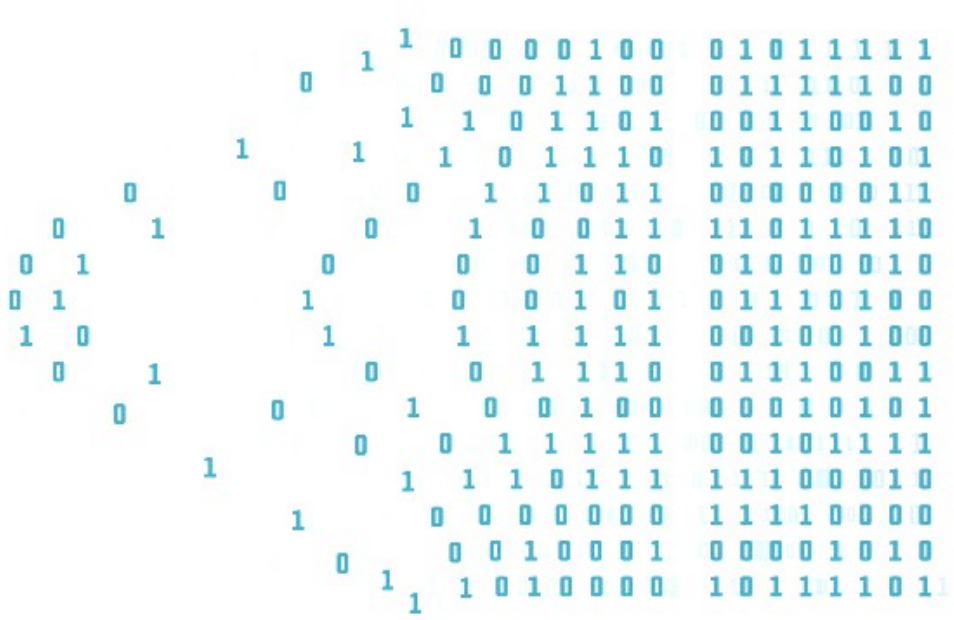
REPRÉSENTATION DES DONNÉES

NSI | 01



REPRÉSENTATION DES NOMBRES ENTIERS POSITIFS

[Frédéric PEURIERE - Marion SZPIEG]

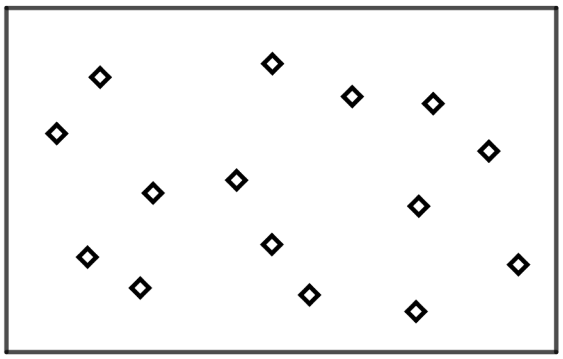
Savoir comment sont représentées les données dans un ordinateur.

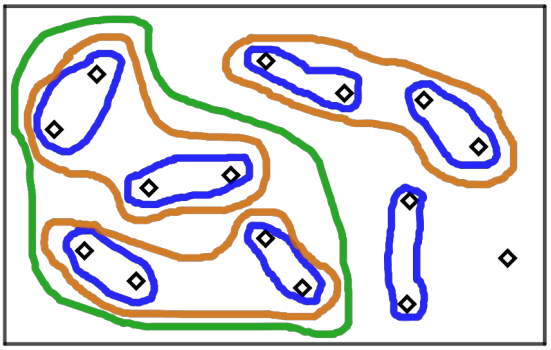
Connaître la base 2 et savoir la manipuler : évaluer le nombre de bits nécessaires pour l'écriture d'un entier en base 2, la somme de deux entiers ou le produit de deux entiers.

Connaître l'écriture d'un entier dans la base 2, 10 et 16 : savoir convertir une base dans une autre

1. Histoires de bases

1.1. Activité d'introduction





Voici deux rectangles identiques dans lesquels il y a des motifs.

1. Dans un rectangle, combien y a-t-il de motifs? 15
2. Dans le rectangle de gauche, entourer les motifs par paquet de 10. On retrouve ainsi la décomposition de 15 dans la base 10 :

$$15 = 1 \times 10 + 5 \times 1 = 1 \times 10^1 + 5 \times 10^0$$

3. Faire la même chose dans le rectangle de droite, mais cette fois-ci en faisant des paquets de 2 motifs:

On a :

- 1 gros paquet (de 8 motifs, soit 2^3 motifs)

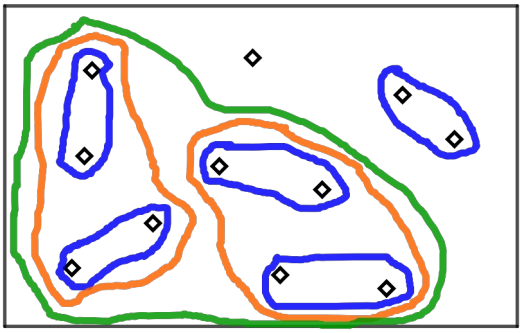
  PLUS
- 1 paquet moyen (de 4 motifs, soit 2^2 motifs)

  PLUS
- 1 petit paquet (de 2 motifs, soit 2^1 motifs)

  PLUS
- 1 motif tout seul (soit 2^0 motif)

Donc le nombre 15 s'écrit en base 2 : 1111, ce qu'on écrit : $(15)_{10} = (1111)_2$

Faire la même chose avec 11 motifs au départ, puis écrire le nombre 11 en base 2 :



On a :

- 1 gros paquet (de 8 motifs, soit 2^3 motifs)

  PLUS
- 0 paquet moyen (de 4 motifs, soit 2^2 motifs)

  PLUS
- 1 petit paquet (de 2 motifs, soit 2^1 motifs)

  PLUS
- 1 motif tout seul (soit 2^0 motif)

Donc : $(11)_{10} = (1011)_2$

Remarque : ne pas oublier le 0 du 2^2 !

1.2. Les différentes bases

1.2.1. La base 10

Vous savez tous compter en base 10 (décimal). Mais comment ça marche déjà ? Comment est construit notre système ?

- On compte avec **10 chiffres** : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- Avec ces derniers, on peut compter jusqu'à 9.
- Et si l'on veut aller au-delà de 9, il faut **changer de rang**. Cela signifie que si le rang des unités est plein, il faut passer à celui des dizaines, puis des centaines, milliers etc.

Par exemple : si on veut passer du nombre 19 au nombre suivant, le rang des unités est "saturé" (plein), car il contient le chiffre 9, et il n'y a pas (dans la base 10) de valeur plus élevée. Il faut donc incrémenter le rang de gauche (qui n'est pas saturé) puis réinitialiser l'état de celui des unités. Ce qui signifie : j'ai 19, je ne peux pas mettre plus de 9 à droite, donc j'ajoute 1 à celui de gauche et je remets à zéro celui de droite.

Un nombre entier va être composé de rangs (unités, dizaines, centaines, etc). Chaque rang vaut le rang précédent multiplié par l'indice de la base. Une centaine vaut dix dizaines, et une dizaine vaut 10 unités etc...

Par exemple, dans $(257)_{10}$, on peut voir trois rangs : celui des centaines, celui des dizaines et celui des unités.

Cela signifie que $(257)_{10}=2\times10^{2}+5\times10^{1}+7\times10^{0}$ ...et ce n'est pas la même chose que $(725)_{10}$ qui est composé des mêmes chiffres, mais dont la décomposition s'écrit : $(725)_{10}=7\times10^{2}+2\times10^{1}+5\times10^{0}$.

Exercice 1

1.2.2. La base 2 (ou le binaire)

Le binaire, c'est le système de comptage des ordinateurs. C'est comme la base 10, sauf que cette fois-ci il n'y a que 2 symboles qui sont 0 et 1 (chiffres binaires ou Binary Digits, plus simplement bits).

Ainsi, en base 10, chaque rang représente une puissance de 10, pour la base 2, chaque rang représente une puissance de 2.

Voici comment compter en binaire jusqu'à 10 :

Nombre en décimal	Nombre en binaire	Explication
0	0	Pour l'instant, ça va.
1	1	Là encore, c'est simple.
2	10	Le premier rang ayant été rempli, on passe au suivant !
3	11	On re-remplit le rang 1.
4	100	Le rang 2 est plein, le rang 1 aussi, on passe au suivant.
5	101	On continue en suivant la même méthode.
6	110	
7	111	
8	1000	On commence le rang 4.
9	1001	On continue comme tout à l'heure.
10	1010	
...		

De façon plus générale, chaque bit représente une puissance de 2. Ainsi :

$$(101101)_2 = 1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0 = 1\times32+0\times16+1\times8+1\times4+0\times2+1\times1$$
$$= 32+8+4+1 = (45)_{10}$$

Exercices 2 et 3 (questions 1 et 2)

1.2.3. La base 16 (ou l'hexadécimale)

L'hexadécimale est utilisée pour simplifier l'écriture binaire, qui peut très vite devenir longue. C'est comme la base 10, sauf que cette fois-ci il y a 16 symboles qui sont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F ; les lettres A, B, C, D, E, F correspondants respectivement aux nombres $(10)_{10}$, $(11)_{10}$, $(12)_{10}$, $(13)_{10}$, $(14)_{10}$ et $(15)_{10}$.

Voici comment compter en hexadécimale jusqu'à 15 :

Base 10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Base 16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Base 2	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

De façon plus générale, chaque chiffre de l'écriture en base 16 d'un nombre représente une puissance de 16. Ainsi :

$$(2A4D)_{16} = 2\times16^3+10\times16^2+4\times16^1+13\times16^0 = 2\times4096+10\times256+16\times4+13 = (10829)_{10}$$

Définition: quelque soit la base *b* dans laquelle on écrit un nombre, le premier chiffre s'appelle le ***poids fort*** de l'écriture en base *b* du nombre, et le dernier chiffre s'appelle le ***poids faible***.

Exercice 3 (question 3)

2. Conversion

2.1. Base 2 <==> base 10

Pour passer de la base 2 à la base 10, on se sert de la décomposition à l'aide des puissances de 2. Par exemple :

$$(1001101)_2 = 1\times2^6+0\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0 = 1\times64+0\times32+0\times16+1\times8+1\times4+0\times2+1\times1$$
$$= 64+8+4+1 = (77)_{10}$$

Pour passer de la base 10 à la base 2, il existe différentes méthodes. En voici une facile à utiliser, et assez efficace avec des grands nombres.

- On part du nombre en base 10.
- On le divise par 2 et on note le reste de la division (soit 1 soit 0)
- On refait la même chose avec le quotient précédent, et on met de nouveau le reste de côté.
- On réitère la division, jusqu'à ce que le quotient soit 0.
- Le nombre en binaire apparaît alors : il suffit de prendre tous les restes de bas en haut.

Par exemple, retrouvons l'écriture en base 2 de $(185)_{10}$:

$$185 = 2\times92+\mathbf{1}$$
$$92 = 2\times46+\mathbf{0}$$
$$46 = 2\times23+\mathbf{0}$$
$$23 = 2\times11+\mathbf{1}$$
$$11 = 2\times5+\mathbf{1}$$
$$5 = 2\times2+\mathbf{1}$$
$$2 = 2\times1+\mathbf{0}$$
$$1 = 2\times0+\mathbf{1}$$

L'écriture en base 2 de 185 se lit alors avec les restes DE BAS EN HAUT : $(185)_{10} = (10111001)_2$

Remarque : l'écriture en binaire de 185 est composée de 9 bits, il faudra donc AU MINIMUM 9 bits pour coder ce nombre. Si on a moins de bits disponibles, le message « *overflow* » sera renvoyé pour informer que la capacité d'écriture est insuffisante.

Exercice 4 + Sacado – Conversion 1)

2.2. Base 16 <==> base 10

Pour passer de la base 16 à la base 10, on se sert de la décomposition à l'aide des puissances de 16. Par exemple :

$$(12B7)_{16} = 1\times16^3 + 2\times16^2 + 11\times16^1 + 7\times16^0 = 1\times4096 + 2\times256 + 11\times16 + 7 = 4096 + 512 + 176 + 7 = (4791)_{10}$$

Pour passer de la base 10 à la base 16, on applique la même méthode que la base 2, mais en faisant des divisions euclidiennes par 16 :

- On part du nombre en base 10.
- On le divise par 16 et on note le reste de la division (un nombre compris entre 0 et 15)
- On refait la même chose avec le quotient précédent, et on met de nouveau le reste de côté.
- On réitère la division, jusqu'à ce que le quotient soit 0.
- Le nombre en binaire apparaît alors : il suffit de prendre tous les restes de bas en haut (en remplaçant 10 par A, 11 par B etc...)

Par exemple, retrouvons l'écriture en base 16 de $(2020)_{10}$:

$$2020 = 16\times126 + \mathbf{4}$$
$$126 = 16\times7 + \mathbf{14}\text{ , donc E}$$
$$7 = 16\times0 + \mathbf{7}$$

Donc : $(2020)_{10} = (7E4)_{16}$

Exercice 5

2.3. Base 16 <==> base 2

La première fausse bonne idée pour passer du binaire à l'hexadécimale (ou inversement) serait de repasser par la base 10 (de référence) … en utilisant les méthodes évoquées ci dessus. C'est LONG et source d'erreur !! Essayez vous verrez !

Pour passer de la base 2 à la base 16, il suffit de grouper les chiffres binaires par 4 (c'est pourquoi la base 16 est souvent utilisée pour simplifier l'écriture des nombres binaires), et de convertir chaque quadruplets de bits en base 16 (avec le tableau de la partie 1.2.3. du cours).

Par exemple : $(1001010111110011)_2 = (95F3)_{16}$ car : 1001 0101 1111 0011
... devient : 9 5 *F* 3

Pour passer de la base 16 à la base 2, on fait la même chose mais dans l'autre sens. Par exemple :
$(E5BB9)_{16} = (11100101101110111001)_2$ puisque : *E* 5 *B* *B* 9
... devient : 1110 0101 1011 1011 1001

Exercice 6 + Sacado – Conversion 2)

3. Encodage

3.1. Regroupements de bits

Un ordinateur stocke donc en binaire des informations sous forme de paquets de 8 informations appelés mots ou octets , byte en anglais (les processeurs récents disposent de 32 – soit 4 octets- ou 64 bits – soit 8 octets - pour stocker UN nombre…)

Groupements de bits	Calculs possibles	Peut coder les entiers …
UN QUADRET (4 bits)	2^4	… entre 0 et 15 (2^4-1)
UN OCTET (8 bits)	2^8	… entre 0 et 255 (2^8-1)
16 bits	2^{16}	… entre 0 et 65 535 (2^{16}-1)
32 bits	2^{32}	… entre 0 et 4 294 967 295 (2^{32}-1)
64 bits	2^{64}	… entre 0 et 2^{64}-1

Mais il faudra aussi stocker des (approximations de) nombres réels, des caractères alpha-numériques, des textes, du son ou des images : il faut donc inventer des ENCODAGES pour représenter ces informations.

Différents types existent donc : **char**, **int**, **unsigned int**, **long** ...(plus tard…)

3.2. Coder un nombre entier positif sur *n* bits

On utilise l'écriture binaire vue en début de chapitre : si le nombre de bits est inférieur à *n*, on rajoute des 0 pour parvenir à *n* bits. S'il n'y a pas assez de bits disponibles, on dit que la capacité d'écriture est insuffisante (« *overflow* »).

Exemple :

Entier	Sur un quadret	Sur un octet	Sur 16 bits	Sur 32 bits
$(135)_{10}$	« *overflow* »	10000111	0000000010000111	00000000000000000000000010000111

Exercice Sacado – Place en mémoire 1) et 2)

3.3. Un peu d'arithmétique ...

Voici la table d'addition en binaire :

+	0	1
0	0	1
1	1	10

Aussi pour sommer 2 nombres en binaire, il suffit d'additionner les 1 ou les 0 un à un, en démarrant par la droite, et sans oublier les retenues :

$(135)_{10}+(100)_{10}=(10000111)_2+(1100100)_2$

Donc : 10000111
+ 1100100
= 11101011 et $(11101011)_2 = (235)_{10}$: c'est le bon résultat !

Voici la table de multiplication en binaire :

×	0	1
0	0	0
1	0	1

Aussi pour multiplier 2 nombres en binaire, il suffit de procéder comme en base 10 et de décaler d'un rang quand on change le bit du multiplicateur :

$(6)_{10}\times(3)_{10}=(110)_2\times(11)_2$

Donc : 110
× 011
110
+ 110
= 10010 et $(10010)_2 = (18)_{10}$: c'est le bon résultat !

Exercice 7

CONVERSIONS DE BASE EN PYTHON

Les concepteurs de Python sont géniaux....Des fonctions existent en Python qui réalisent toutes les conversions : **la base par défaut est la base 10, les entiers illimités dans les bases 2, 10 ou 16 (pas d'overflow).**

Entrez dans la console de Python la série d´instructions suivantes en validant par ENTREE. Observez les résultats obtenus puis complétez le tableau « aide mémoire ».

Manipuler une séquence de nombres dans différentes bases :

>>>0b01	>>>0b1111	>>>0b11	>>>0b409	>>>0b0100101
>>>bin(2)	>>>bin(1)	>>>bin(101)	>>>bin(345)	>>>bin(78.9)
>>>0x679	>>>0xAC	>>>0xCY	>>>0x50BV	>>>0x50B
>>>hex(9)	>>>hex(AC7)	>>>hex(11)	>>>hex(0b11)	>>>hex(896781)
>>>int('101010',2)	>>>int('101010',16)	>>>int('11',2)	>>>int('11',10)	>>>int('11',16)
>>>int('AC',16)	>>>int('AC',2)	>>>int('1452',20) : il existe d'autres bases que les bases 2, 10 et 16 !		

Remarque : si vous obtenez un message d'erreur, expliquer pourquoi.

AIDE MÉMOIRE

Pour...	*Je tape dans la console...*
Pour convertir un nombre binaire en base 10	
Obtenir l'écriture décimale de $(01001101)_2$	>>>0b01001101 OU >>>int('01001101',2)
Pour convertir un nombre décimal en base 2	
Obtenir l'écriture binaire de $(43)_{10}$	>>>bin(43)
Pour convertir un nombre hexadécimal en base 10	
Obtenir l'écriture décimale de $(DF40E)_{16}$	>>>0xDF40E OU >>>int('DF40E',16)
Pour convertir un nombre décimal en base 16	
Obtenir l'écriture hexadécimale de $(1759)_{10}$	>>>hex(1759)
Pour convertir un nombre binaire en base 16	
Obtenir l'écriture hexadécimale de $(1001011)_2$	>>>hex(0b1001011)
Pour convertir un nombre hexadécimal en base 2	
Obtenir l'écriture binaire de $(A48E)_{16}$	>>>bin(0xA48E) OU >>>bin(int('A48E',16))