Exemples d'utilisations du binaire et de l'hexadécimal

Exercice 1 - Cryptage d'un message (extrait de métropole mai 2019)

On dispose d'une clé de cryptage notée $(a_1, a_2, a_3, a_4, a_5) = (63, 62, 65, 68, 34)$.
Cette clé, publiée par la personne destinataire, permet à quiconque de lui envoyer un message crypté. Voici comment on crypte le message.
On associe d'abord à chaque lettre son rang dans l'alphabet, selon la correspondance suivante :

Lettre	A	B	C	D	E	F	G	H	I	J	K	L	M
Rang	1	2	3	4	5	6	7	8	9	10	11	12	13
Lettre	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Rang	14	15	16	17	18	19	20	21	22	23	24	25	26

Pour crypter une lettre :

- on détermine son rang à l'aide du tableau de correspondance précédent ;
- on écrit ce nombre en base 2 sur 5 bits ; on ainsi obtient 5 chiffres $(m_1, m_2, m_3, m_4, m_5)$, chaque chiffre étant égal à 0 ou à 1 ;
- on détermine alors la valeur cryptée, égale à la somme $\sigma = a_1m_1 + a_2m_2 + a_3m_3 + a_4m_4 + a_5m_5$.

On remarque qu'une lettre est ainsi cryptée par un nombre entier.
Exemple : on veut crypter la lettre « I ».

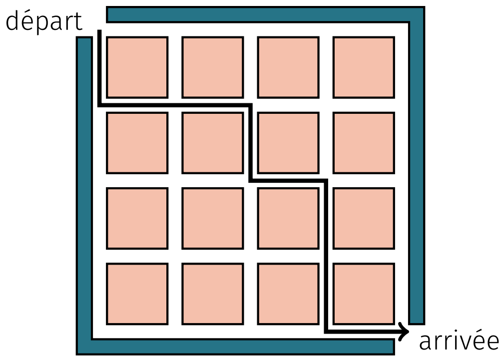
- Le rang de I est 9 ;
- on écrit ce nombre en base deux sur 5 bits : $9_{10} = 8 + 1 = (01001)_2$,
- on calcule la somme $\sigma = 0 \times 63 + 1 \times 62 + 0 \times 65 + 0 \times 68 + 1 \times 34 = 96$.

La lettre « I »est donc cryptée par l'entier 96.

Question : crypter la lettre « W » selon cette méthode.

Exercice 2 - Un petit jeu

On veut programmer un petit jeu : une bille commence dans la grille suivante, toujours à la case départ.
On doit l'amener à la case arrivée en appuyant seulement sur les touches → et ↓ du clavier. Voici un exemple de partie :



Pour représenter les différents parcours, on associe à chacun d'entre eux un entier de la manière suivante :

– on note la séquence de touches pressées dans l'ordre ;

– on remplace chaque ↓ par 0 et chaque → par 1 ;

– on obtient l'écriture binaire de l'entier final.

1. Justifier que l'entier obtenu à partir de l'exemple vaut 105.

2. Combien faut-il de bits pour représenter un chemin donné ?

3. Représenter le parcours associé au nombre 85.

4. On considère un entier N représentant un chemin.

   a. Est-il possible d'avoir $N = 31$?

   b. Rappeler ce que vaut $(\underbrace{1...1}_{k \text{ bits}})_2$.

   c. Quel est le plus petit entier N qui représente un chemin ? Dessiner le chemin associé.

   d. Quel est le plus grand ? Dessiner le chemin associé.

Exercice 3 - Additions en base 2

Les additions « à la main » en base 2 s'effectuent de la même manière qu'en base 10, la seule différence c'est que deux 1 donnent $(2)_{10}$ donc $(10)_2$, donc un zéro et une retenue de 1. Quand il y a deux 1 et une retenue de 1 en plus, cela donne $(3)_{10}$ donc $(11)_2$, donc un 1 et une retenue de 1.

Exemple

On veut calculer $(1101\ 0101)_2 + (1\ 1100)_2$, on pose l'opération en mettant les retenues en rouge.

				1	1			
	1	1	0	1	0	1	0	1
+				1	1	1	0	0
=	1	1	1	1	0	0	0	1

Le résultat est donc $(1111\ 0001)_2$.

1. Donner les écritures décimales de $(1101\ 0101)_2$, $(1\ 1100)_2$, $(1111\ 0001)_2$ et vérifier qu'il n'y a pas d'erreur d'addition.

2. Vérifier que l'on a bien l'égalité $138 + 111 = 249$ lorsque l'on effectue l'addition en binaire.

3. Calculer $(1011\ 1101)_2 + (11)_2$ et donner les écritures décimales des 3 nombres intervenant dans cette addition.

Exercice 4 - Entiers non signés sur un octet

Dans certains langages, on utilise un octet (c'est-à-dire 8 bits) pour stocker un entier positif. On stocke dans l'octet l'écriture binaire de l'entier, telle quelle.

- en C et C++, ce type de variable s'appelle `unsigned char` ;
- en C#, il s'appelle `byte`.

1. Quels sont les entiers représentables de cette manière ?
2. Pour ajouter deux valeurs de ce type, on effectue l'addition en binaire, mais s'il y a une retenue à reporter à la fin (qui irait théoriquement dans un 9ème bit) on ne la prend pas en compte : on ne garde que les 8 premiers bits du résultats en partant de la droite.

   a. Ajouter avec cette méthode 129 et 130 et donner le résultat sous forme décimale.

   b. Ci dessus on a écrit un petit programme en C++. Dire ce qu'il fait et interpréter le résultat affiché dans la console.

Code C++

```cpp
#include <iostream> // bibliothèque d'affichage
int main() // début de la fonction principale
{
    unsigned char c = 0; // on définit la variable c
    for (int i = 0; i < 300; i++) // on fait une boucle pour
    {
        std::cout << "valeur de i : " << i; // on affiche la valeur de i
        std::cout << "  et valeur de c : " << (int)c; // on affiche la valeur
         ↪  de c en base 10
        std::cout << endl; // on revient à la ligne (END Line)
        c++; // on augmente c de 1
    }
    return 0; // la fonction principale renvoie zéro par convention
}
```

Voilà ce que la console affiche :

```
valeur de i : 0 et valeur de c : 0
valeur de i : 1 et valeur de c : 1
```
et cætera
```
valeur de i : 254 et valeur de c : 254
valeur de i : 255 et valeur de c : 255
valeur de i : 256 et valeur de c : 0
valeur de i : 257 et valeur de c : 1
```
et cætera
```
valeur de i : 298 et valeur de c : 42
valeur de i : 299 et valeur de c : 43
```

Exercice 5 - Entiers signés en complément à 2 sur 8 bits.

Pour représenter un entier signé (positif ou négatif) sur un octet, on peut penser au système suivant : le bit de poids fort vaut 0 si l'entier est positif et 1 s'il est négatif. Les 7 autres bits servent à représenter la partie numérique du nombre.
Par exemple,

$$(1001\ 1010)_2$$

représente un nombre négatif car son bit de poids fort est à 1. Sa partie numérique est $(1\ 1010)_2 = 26$, donc il représente -26.

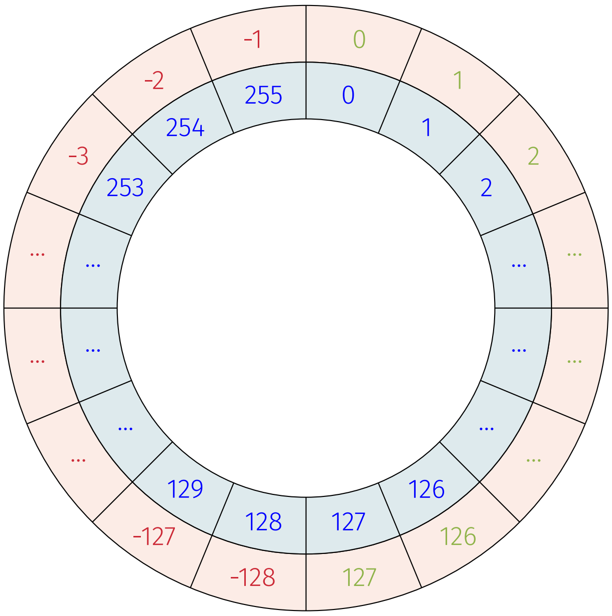
1. **a.** Que représente $(0000\ 0000)_2$? Et $(1000\ 0000)_2$?

   b. Donner la représentation de +26 et ajouter les représentations de +26 et -26 en binaire. Quel nombre représente le résultat obtenu ? Est-ce cohérent ?

2. Pour pallier ces problèmes, on représente les entiers *en complément à 2 sur 8 bits* :

   – si on veut représenter un entier compris entre 0 et $2^7 - 1 = 127$, alors on le représente tel quel sur un octet ;

   – si on veut représenter un entier n compris entre $-2^8 = -128$ et -1, alors on le représente par $n + 256$.



Ce schéma représente l'encodage des entiers relatifs en complément à 2 sur 8 bits (que l'on retrouve dans le type `char` en C et C++).

En bleu figurent les nombres tels qu'ils sont stockés dans la mémoire, sur un octet.
En orange figurent les nombres représentés par les nombres bleus :

– lorsqu'il est compris entre 0 et 127, un nombre bleu représente ce même nombre (affiché en vert) ;

– lorsqu'il est compris entre 128 et 255, un nombre bleu p représente le nombre $p - 256$ (affiché en rouge).

a. Donner la représentation en complément à 2 sur 8 bits du nombre +7.

b. Faire de même avec -3.

c. Ajouter en binaire ces deux représentations (une éventuelle retenue après le 8^e bit est ignorée), quel nombre cela représente-t-il ?

3. Recommencer la question précédente et vérifier que l'égalité $-12 + 8 = -4$ est vérifiée en complément à 2 sur 8 bits.

4. Le programme suivant affiche -116. Expliquer ce résultat.

Code C++

```cpp
#include <iostream> // nécessaire pour utiliser cout
int main() // début de la fonction main
{
    char c1 = 120; // on définit une première variable
    char c2 = 20; // puis une deuxième
    char c3 = c1+c2; // on les ajoute
    std::cout << (int) c3; // on affiche le résultat en base 10
    return 0; // la fonction main renvoie traditionnellement zéro
}
```

Exercice 6 - Masque jetable (métropole mai 2017)

Le but de cet exercice est d'étudier une méthode de cryptage inventée par Gilbert Vernam en 1917, et appelée « masque jetable ».
Dans tout l'exercice, on note respectivement M le mot initial, K la clé de cryptage et Y le mot crypté.
Les trois nombres M, K, Y sont des entiers naturels.
Les chiffres hexadécimaux sont notés 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

1. Questions préliminaires

   a. Donner la représentation en hexadécimal de l'entier binaire 1011101_2.

   b. Calculer en travaillant dans le système hexadécimal les sommes $(7)_{16} + (4)_{16}$ et $(A)_{16} + (C)_{16}$.

2. Soit M et K deux entiers naturels écrits en hexadécimal, tels que la longueur de l'écriture de K est supérieure ou égale à celle de M, et tels que l'écriture de K ne comporte aucun chiffre 0.

   Pour crypter le mot M avec la clé K, on procède comme suit : pour chaque chiffre m du mot initial M, on considère le chiffre k de la clé K qui a la même position que m dans l'écriture.

   On obtient alors le chiffre y du mot crypté Y qui a la même position que m dans l'écriture du mot initial M, de la façon suivante : y est le chiffre hexadécimal des unités de la somme $m + k$.

   Le mot crypté Y est déterminé en hexadécimal par la juxtaposition dans le même ordre des chiffres y calculés pour chaque chiffre m du mot M.

   > *Exemple* : avec $M = (49)_{16}$ et $K = (19)_{16}$
   >
   > - Avec le chiffre de rang 1 en partant de la droite : $m = 9$ et $k = 9$; donc $m + k = (12)_{16}$ et par suite $y = 2$;
   > - avec le chiffre de rang 2 : $m = 4$ et $k = 1$; donc $m + k = (5)_{16}$ et par suite $y = 5$.

   Donc le mot crypté est $Y = (52)_{16}$.

   Question : avec le mot initial $M = (7A)_{16}$ et la clé $K = (4C)_{16}$, déterminer le mot crypté Y.

3. Par cette méthode, on admet que le décryptage suit les mêmes étapes en remplaçant la clé K par une autre clé K'. Lorsque l'écriture de K comporte au maximum deux chiffres hexadécimaux, la clé K' est l'écriture en hexadécimal de la différence (écrite en décimal) $(272)_{10} - (K)_{10}$.

   Cette question est une question à choix multiple. Une seule réponse est exacte. Recopier sur la copie seulement la réponse exacte. On ne demande pas de justification.

   Avec la clé de cryptage $K = (19)_{16}$, la clé de décryptage K' est égale à :

Réponse A : $(253)_{16}$	Réponse B : $(247)_{16}$
Réponse C : $(FD)_{16}$	Réponse D : $(F7)_{16}$