Systèmes de numération

On appelle système de numération tout procédé permettant d'associer à chaque nombre une suite de caractères. Le système de numération en usage aujourd'hui est la numération dite de position en base 10, mais d'autres systèmes ont été utilisés et le sont encore : les chiffres romains par exemple. Il se trouve que dans l'état actuel des technologies, le système le plus commode pour les ordinateurs est la représentation de position en base 2, encore appelée représentation binaire.

Développement d'un entier en base B

Soit $B \geq 2$ un entier. On appelle *chiffres en base B* les entiers compris entre 0 et $B-1$. Donc, une fois la base fixée, il y a un nombre fini de chiffres, et en théorie, il est toujours possible de représenter chaque chiffre par un symbole.

Théorème 1 *Soit $B \geq 2$ un entier. Soit $n > 0$ un entier. Alors il existe une unique suite d'entiers naturels $(a_k, a_{k-1}, \ldots, a_1, a_0)$ telle que :*

- $a_k \neq 0$
- *Pour tout i, a_i est un chiffre en base B.*
- $n = a_0 B^0 + a_1 B^1 + a_2 B^2 + \cdots + a_k B^k$.

Cette suite s'appelle le *développement en base B* de n. Les a_i s'appellent les *chiffres* de n. On écrit $n = a_k a_{k-1} \ldots a_1 a_0$. Dans l'usage courant en informatique, les bases les plus utilisées sont la base 10 et la base 2. Un entier écrit en base deux est dit sous forme *binaire*, en base 10 sous forme *décimale*. On appelle *bits* les chiffres d'un entier en base 2 (de l'anglais 'binary digit'). Les bases 8 et 16 sont aussi utilisées par les informaticiens : on parle de nombres donnés en *octal* et en *hexadécimal*. En hexadécimal, le nombre de chiffres depasse 10. On utilise les lettres A, B, C, D, E, F pour noter les chiffres après 9.

Trouver les chiffres d'un nombre

Pour trouver les chiffres d'un nombre n dans une base B, il suffit d'être capable de calculer quelques divisions Euclidiennes :

$q_0 = n$
Pour tout $i \geq 0$ on définit a_i et q_{i+1} par :
$q_i = q_{i+1} B + a_i$, avec $0 \leq a_i < B$

Théorème 2 *À partir d'un rang $k+1$, la suite (a_i) devient nulle. Et on a, en base B : $n = a_k a_{k-1} \ldots a_0$.*

PREUVE — On montre d'abord que pour tout $i \in \mathbb{N}$ on a $n = q_i B^i + a_0 B^0 + \cdots + a_{i-1} B^{i-1}$. Pour $i = 0$, c'est vrai car $q_0 = n$. Si la propriété est vraie pour $i \geq 0$, alors $n = q_i B^i + a_0 B^0 + \cdots + a_{i-1} B^{i-1} = (q_{i+1} B + a_i) B^i + a_0 B^0 + \cdots + a_{i-1} B^{i-1} = q_{i+1} B^{i+1} + a_0 B^0 + \cdots + a_i B^i$. La propriété est bien vraie pour $i+1$, et de là pour tout $i \in \mathbb{N}$.

Comme $(B^i)_{i \in \mathbb{N}}$ tend vers l'infini, les q_i doivent tous être nuls à partir d'un certain rang, et les a_i le sont donc aussi. La relation ci-dessus montre bien $n = a_k a_{k-1} \ldots a_0$. □

Représentation des nombres réels

La représentation des nombres réels pose des problèmes théoriques et pratiques. En effet, le mathématicien Cantor a montré à la fin du XIX^{e} siècle qu'étant donné un ensemble fini de symboles A, il est impossible d'associer sans ambiguïté à chaque nombre réel une suite finie d'éléments de A. Toutefois, il existe de nombreux systèmes permettant d'associer à chaque réel une suite éventuellement infinie de symboles.

Afin de ne pas alourdir le cours, on se restreint au cas des réels de $[0,1[$. Cela a pour avantage que tous les nombres s'écriront $0,\ldots$. Et cela est suffisant, car tout nombre réel est la somme d'un entier relatif et d'un réel de $[0,1[$. Soit $(a_n)_{n\in\mathbb{N}^*}$ une suite de chiffres en base B, et soit $x \in [0,1[$ un nombre réel. On pose :

$$x_n = a_1 B^{-1} + a_2 B^{-2} + \cdots + a_n B^{-n}$$

On dit que $(a_n)_{n\in\mathbb{N}}$ est une représentation de x si $x = \lim_{n\to\infty} x_n$. On écrit alors $x = 0, a_1 a_2 a_3 \ldots$. Il faut noter que pour n'importe quel choix de la suite $(a_n)_{n\in\mathbb{N}^*}$, la suite $(x_n)_{n\in\mathbb{N}^*}$ converge.

Dans le cas où la suite $(a_n)_n \in \mathbb{N}^*$ est périodique, de période p à partir du terme a_k, on s'autorise la notation suivante :

$$x = 0, a_1 a_2 a_3 \ldots a_{k-1}\overline{a_k a_{k+1} \ldots a_{k+p-1}}$$

Attention, certains nombres réels peuvent avoir deux représentations différentes en base B. Par exemple, en base 10, on a $1/2 = 0{,}5 = 0,499999\cdots = 0,4\overline{9}$.

Algorithme pour trouver le développement des rationnels

L'algorithme suivant, qui n'est autre que celui enseigné aux enfants à l'école primaire, permet de trouver un développement en base B pour tout nombre rationnel de $[0,1[$. On se donne en entrée deux entiers naturels u et v, avec $u < v$. On définit les suites $(r_k)_{k\in\mathbb{N}}$ et $(q_k)_{k\in\mathbb{N}^*}$:

$r_0 = u$
Pour tout $k > 0$ on définit r_k et q_k par :
$Br_{k-1} = vq_k + r_k$, avec $0 \leq r_k < v$

On remarque que pour tout k, $r_k < v$. Donc, $0 \leq q_k < B$, autrement dit les entiers q_k peuvent être vus comme des chiffres en base B. Le théorème suivant montre qu'en fait, les suites définies ci-dessus permettent d'obtenir le développement en base b de u/v :

Théorème 3 $u/v = 0, q_1 q_2 q_3 \ldots$

PREUVE — On montre d'abord pour tout $k \geq 0$:

$$\frac{r_k}{B^k v} = \frac{u}{v} - \left(\frac{q_1}{B^1} + \cdots + \frac{q_k}{B^k}\right)$$

Pour $k = 0$, c'est évident car $r_0 = u$. Si $k \geq 1$, supposons la formule vraie pour $k-1$. Puisque $r_k = Br_{k-1} - vq_k$, on a alors :

$$\frac{r_k}{B^k v} = \frac{r_{k-1}}{B^{k-1} v} - \frac{q_k}{B^k} = \frac{u}{v} - \left(\frac{q_1}{B^1} + \cdots + \frac{q_k}{B^k}\right)$$

par hypothèse de recurrence.

Comme $r_k < v$, on a donc :

$$\frac{u}{v} - \left(\frac{q_1}{B^1} + \cdots + \frac{q_k}{B^k}\right) < \frac{1}{B^k}$$

ce qui montre bien que :

$$\lim_{k \to +\infty} \frac{q_1}{B^1} + \cdots + \frac{q_k}{B^k} = \frac{u}{v}$$

□

Encore une fois, l'algorithme nous permet de démontrer un résultat théorique, qui a pour conséquence, entre autres, l'existence de nombres irrationnels :

Théorème 4 *Un nombre réel est rationnel si et seulement si il admet un développement périodique en base B.*

Exercice 1

1. Écrire une fonction de prototype `void conv(int)` convertit un entier en base 2. Cette fonction affiche les bits de l'entier n passé en argument.

2. Écrire une fonction de prototype `void conv(int n, int B)` convertit un entier n en base B. Cette fonction affiche les chiffres en base B de l'entier n passé en argument. La fonction doit fonctionner pour les bases 2 à 16 au moins.

Corrigé

L'application directe de l'algorithme vu en cours a un petit défaut : les chiffres sont affichés à l'envers. Ça n'est pas grave : l'énoncé ne demande pas de les afficher l'endroit. Résoudre ce problème en toute généralité nécessite d'utiliser un algorithme inefficace ou des tableaux. Ci-dessous, une combine pour afficher à l'endroit en base 2.

```
#include<stdio.h>

void conv2(int n){

  while (n!=0){
    printf("%i", n%2);
    n=n/2;
  }
  printf("\n");
}

void convB(int n, int B){

  int c; char affc;

  while (n!=0){
    c=n%B;
```

```
    if (c<=9) affc='0'+c; else affc='A'+c-10;
    printf("%c", affc);
    n=n/B;
  }
  printf("\n");
}


void convAlendroit(int n){
  int res, d;

  res=0;
  d=1;

  while (n!=0){
    res = res + n%2*d;
    d=10*d;
    n=n/2;
  }
  printf("%i", res);
  printf("\n");
}



main(){
  convB(255, 16);
}
```

Exercice 2

1. Écrire une fonction de prototype `void inv(int)` qui calcule et affiche les chiffres en base 10 de l'inverse d'un entier n passé en argument.

2. Vérifier la variante suivante du petit théorème de Fermat : le développement en base 10 de l'inverse d'un entier premier p différent de 2 et 5 est de la forme $0,\overline{a_1 a_2 \dots a_k}$ où k est un diviseur de $p-1$.

Indication : on pourra utiliser le fait qu'en base 10, les inverses des entiers qui ne sont ni divisibles par 2 ni par 5 sont de la forme $0,\overline{a_1 a_2 \dots a_k}$. La période peut être détectée lors de l'exécution de l'algorithme par le retour de 1 dans la suite des restes.

3. Le petit théorème de Fermat tel qu'énoncé ci-dessus permet-il de tester si un nombre est premier ?

Corrigé

```
#include<stdio.h>
```

```
#define B 10

// Affiche les chiffres de l'inverse
void affinv(int n){
  int r; int q;

  r=1;

  do{
    q=(B*r) / n;
    r=(B*r) % n;
    printf("%i", q);
  }
  while (r!=1);

  printf("\n");
}


//Calcule la longueur de la période
int periode(int n){
  int r; int q; int i;

  r=1; i=0;

  do{
    q=(B*r) / n;
    r=(B*r) % n;
    //printf("%i", q);
    i++;
  }
  while (r!=1);
  //printf("\n");

  return i;
}



main(){
  int n, d;
  int prem, ferm;

  for(n=2; n<10000; n++){
    if (n%2!=0 && n%5!=0){
      //On teste si n est premier
      d=2;
      while(d<=n/2 && n%d != 0) d++;
      prem = (n%d != 0);
```

```
      ferm = (n-1) % periode(n)  == 0;

      if (ferm && prem) printf("%i confirme le petit theoreme de Fermat\n", n);
      if (!ferm && prem) printf("%i infirme le petit theoreme de Fermat\n", n);
      if (ferm && !prem) printf("%i infirme la reciproque du petit theoreme de Fermat\n", n);

    }

  }

}
```

1. Voir ci-dessus.

2. Voir ci-dessus.

3. L'exécution du programme montre que la réciproque du théorème de Fermat admet des contre-exemples, comme 9, 561 ou 1729. Mais il y a assez peu de contre-exemples. Mais on ne peut pas utiliser le théorème de Fermat pour tester si un nombre est premier. Changer de base permet d'éliminer certain de ces contre-exemples, mais pas tous. Les nombres de Carmichaël sont par définition les nombres contre-exemple en toute base. Les plus petits nombre de Carmichaël sont 561, 1105 et 1729. Noter que 1729 est bizarre à plus d'un titre : c'est le plus petit entier s'exprimant de deux manière différentes comme somme de deux cubes : $1729 = 10^3 + 9^3 = 12^3 + 1^3$.