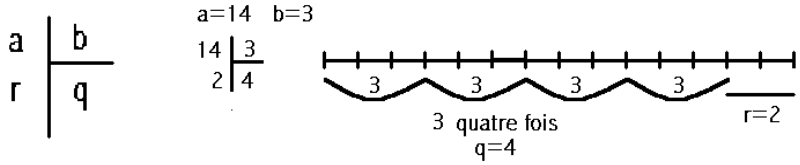
Chapitre 1
Division euclidienne et algorithme d'Euclide

L'arithmétique consiste à travailler exclusivement avec des nombres entiers, positifs, négatifs ou nul. Quand on additionne deux nombres entiers, on obtient un nombre entier, et de même en soustraction. Quand on multiplie deux nombres entiers, on trouve encore un nombre entier. Mais quand on divise ? En général, la division ne tombe pas juste, et le résultat n'est pas un nombre entier. On sait que diviser, c'est multiplier par l'inverse. Mais l'inverse d'un nombre entier n'est pas un entier, en général. Prenons 3, son inverse est 1/3, dans le cadre des nombres rationnels (fractions d'entiers), et ce n'est pas un entier. Cela signifie qu'on ne peut pas trouver un entier 3^{-1} tel que $3 \times 3^{-1} = 1$. En fait les seuls nombres entiers ayant un inverse entier sont 1 et –1, qui ont pour inverses eux-mêmes. On est dans un contexte très différent de celui des nombres réels ou des nombres rationnels, qui eux, à part 0, ont toujours un inverse. Toute l'arithmétique tourne autour de cet écueil propre aux nombres entiers, à savoir le problème de la division.

1.1. Division euclidienne

Il s'agit de la division la plus simple, celle où n'interviennent que des nombres entiers (pas de nombres à virgule). Donnons-nous deux nombres entiers positifs a et b. La division de a par b donne un quotient q et un reste r. On appelle a le dividende et b le diviseur. Mais qui sont q et r ? Par définition q est le plus grand nombre de fois que l'on peut mettre b dans a, et le résidu est le reste r. Par exemple quand on divise 14 par 3, on peut mettre au maximum 4 fois le 3 dans 14, d'où $q = 4$ et $r = 2$.



Ainsi définis, le quotient q est unique, ainsi que le reste r, et ce dernier est forcément inférieur à b : $0 \le r < b$.

On peut écrire $a = bq + r$ avec $0 \leq r < b$. Avec a et b donnés, cette équation ayant pour inconnues q et r (des entiers positifs ou nuls), avec en plus la contrainte pour r d'être inférieur à b, admet une solution unique.

Autrement dit, une division euclidienne est fausse si l'on prend un quotient trop grand, avec bq qui dépasse a (le reste serait alors négatif) ou s'il est trop petit, auquel cas c'est le reste qui est trop grand : $r \geq b$.

1.1.1. Programmation

Avec a et b déclarés comme entiers (*int*), le fait d'écrire *a/b* donne q, et le reste r s'obtient en faisant *a modulo b*, soit *a%b* en langage *C*. Par exemple :

```
int a,b,q,r ;
a=14; b=3;  /* un exemple */
q=a/b; r=a%b;
printf("la division de %d par %d donne q=%d, r=%d", a,b,q,r);
```

On peut aussi programmer soi-même l'opération de division :

```
a=14 ; b=3 ;
nbdefois=0 ;
while (nbdefois*b <= a) nbdefois++ ;
q=nbdefois-1 ; r=a-b*q ;
```

Remarque : La notion de division euclidienne peut s'étendre à des nombres entiers négatifs. Diviser par exemple –14 par –3 revient à diviser 14 par 3. Diviser 14 par –3 revient à diviser –14 par 3. Le seul cas nouveau est en fait a négatif et b positif. On s'arrange alors pour trouver b et q tels que $a = bq + r$ avec $0 \leq r < b$: le reste est toujours positif ou nul. Ainsi quand on divise –14 par 3, on trouve $q = -5$ et $r = 1$ (alors que $-14/3 \approx -4{,}66$).

1.1.2. Application : Comment tracer une droite sur un écran d'ordinateur ?

Prenons le cas d'un segment $[OA]$ avec A de coordonnées entières positives dx et dy dans le repère orthonormé d'origine O, et supposons que sa pente est inférieure ou égale à 1, soit $dy \leq dx$. Pour tracer cette droite sur la grille de l'écran de l'ordinateur, où tous les pixels ont des coordonnées entières, on doit allumer des pixels qui en général ne sont pas exactement sur la droite puisque celle-ci n'a que peu de points à coordonnées entières sur elle. On va en fait construire ce que l'on appelle un chemin rasant par en-dessous. Voici un exemple avec $dx = 7$ et $dy = 3$ (*figure 1*).

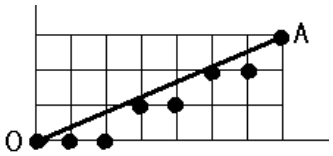


Figure 1 : Chemin rasant sous (*OA*)

Quand on prend les valeurs entières successives de x, de 0 à $dx = 7$, les points correspondants sur la droite ont pour ordonnée les nombres à virgule 0/7, 3/7, 6/7, 9/7, 12/7, 15/7, 18/7, 21/7 = 3, toutes ces ordonnées sont de la forme $k\,dy / dx$, avec les numérateurs augmentant de $dy = 3$ à chaque fois. Il s'agit d'approcher ces points par des points à ordonnées entières situés au plus près d'eux et au-dessous. On va remplacer les ordonnées exactes sous forme de fraction $(k\,dy) / dx$ par le quotient euclidien de $k\,dy$ par dx : par exemple l'ordonnée 9/7 est remplacée par l'ordonnée 1. A cause de la pente inférieure ou égale à 1, chaque fois que x augmente de 1, l'ordonnée entière (le quotient euclidien) y reste soit fixe, soit augmente de 1.

Fractions :	0/7	3/7	6/7	9/7	12/7	15/7	18/7	21/7
Quotient euclidien y :	0	0	0	1	1	2	2	3
Augmentation de y :	0	0	0	1	0	1	0	1
Reste euclidien :	0	3	6	2	5	1	4	0

Pour tracer la «droite», on va utiliser la suite des restes euclidiens. Quand le numérateur $k\,dy$ de la fraction augmente de $dy = 3$, le reste augmente de 3 aussi, le quotient restant fixe, mais il peut devenir trop grand en dépassant $dx = 7$, dans ce cas on doit rectifier l'erreur de la division en augmentant le quotient de 1 et en diminuant le reste de $dx = 7$. A chaque fois que l'on fait cette rectification, y augmente de 1. Ainsi la droite parfaite est remplacée par un cheminement à base de pas horizontaux → ou diagonaux ↗ .

D'où le programme :

```
On se donne dx et dy entiers positifs avec dy ≤ dx
x=0 ; y=0 ; reste=0 ;  dessiner ce point x,y
for(i=1 ; i <= dx ; i++)
  {x++ ; reste+=dy ; if (reste > =dx) {reste - = dx ; y ++ ;}  dessiner le point x,y}
```

1.1.3. Extension : La division avec virgule

On a deux nombres entiers positifs a et b et l'on veut diviser a par b. Au lieu de se contenter de la division euclidienne, on continue au-delà de la virgule, comme

on apprend à le faire à l'école. Après avoir fait la division euclidienne classique, on ajoute un 0 au reste et on refait une division euclidienne, ce qui donne le premier chiffre derrière la virgule, puis on rajoute un 0 au reste et on fait la division pour avoir un deuxième chiffre derrière la virgule. Et l'on continue autant qu'on le désire. On obtient de la sorte l'écriture du nombre rationnel a/b sous forme d'un nombre à virgule avec une infinité de chiffres derrière la virgule. Mais n'oublions pas que tous les restes successifs sont inférieurs à b, donc en nombre limité. On est sûr, au bout d'un nombre fini de divisions, de retrouver un reste que l'on avait déjà trouvé. Cela signifie que les mêmes chiffres vont revenir par blocs dans les quotients successifs. Autrement dit, le développement décimal du nombre rationnel (de la fraction d'entiers) finit toujours par devenir périodique, éternellement.

Exemple
Division de 759 par 28

```
759     |28
199     |-------
  30    |27, 10  714285
   200
     40
    120
     80
    240
     160
      20
```

Le bloc 714285 va se répéter indéfiniment

Voici le programme correspondant :

```
Les restes et quotients successifs sont placés dans des tableaux r[] et q[]
On se donne a et b entiers positifs, par exemple a=759  et  b=28.
r[0]=a%b;    q[0]=a/b;
printf("quotient: %d , ", q[0]);
/* c'est la partie entière du quotient, 27 dans l'exemple */
i=0;  flag=0;
for(;;)      /* boucle infinie qui sera arrêtée par un break  */
  {   i++;  r[i]=(10*r[i-1])%b;  q[i]=(10*r[i-1])/b;
              /* quotient et reste derrière la virgule */
     for(j=0; j<i; j++)    if (r[j]==r[i])  /* on teste si on a déjà trouvé ce reste */
        {  flag=1;
           for(k=1; k<=j; k++) printf("%d",q[k]);   printf("  ");
                       for(k= j+1; k<=i; k++)  printf("%d",q[k]);   break;
        }
     if (flag==1) break;
  }
T=i-j ; printf("\nLa longueur de la période est %d ",T);
```

1.2. Diviseurs d'un nombre

1.2.1. Définition

On dit qu'un nombre *d* est un diviseur d'un nombre *a*, ou encore que *d* divise *a*, lorsque la division de *a* par *d* tombe juste, ou encore si l'on peut trouver un nombre *k* entier tel que $a = k\,d$. Cela s'écrit $d \mid a$. Par exemple, 3|15, 3 est un diviseur de 15 puisque la division de 15 par 3 donne un reste nul, avec $15 = 5 \times 3$. Un nombre *a* (autre que 0) admet un nombre fini de diviseurs, qui sont tous inférieurs ou égal à lui. Parmi les diviseurs, le plus petit est 1 et le plus grand le nombre lui-même. Et l'on a une propriété évidente de transitivité : si un nombre *d* divise *a* et qu'à son tour *a* divise *b*, alors *d* divise *b*.

1.2.2. Comment obtenir tous les diviseurs d'un nombre ?

Partons d'un exemple, en cherchant les diviseurs du nombre $a = 30$. On écrit, en partant de 1 :

$30 = 1 \times 30$ 1 et 30 sont deux diviseurs
$30 = 2 \times 15$ 2 et 15 sont des diviseurs
$30 = 3 \times 10$ 3 et 10 sont des diviseurs
$30 = 5 \times 6$ 5 et 6 sont des diviseurs.

Il n'y a pas d'autres diviseurs, car au-delà du diviseur 5 écrit en premier, on retrouve les autres diviseurs. On vient de trouver les diviseurs de 30, qui sont au nombre de 8.

Autre exemple, avec $a = 25$ on a : $25 = 1 \times 25$, et $25 = 5 \times 5$, d'où 25 admet 3 diviseurs.

D'où la méthode : On essaye les nombres inférieurs ou égal à *a*, à partir de 1, par ordre croissant. On prend ceux qui divisent *a*. A chaque fois, on obtient deux diviseurs (sauf cas exceptionnel où les deux diviseurs sont les mêmes, comme pour $25 = 5 \times 5$). On continue tant que le premier diviseur est inférieur ou égal au second.

Et maintenant le programme :

```
On se donne le nombre a (>1)
d1=1 ; d2= a ; compteur= 2;  afficher d1 et d2 ;
for (d1=2 ; d1*d1<=a ; d1++)  if (a % d1= =0)
  { afficher d1; compteur++;
    d2= a/d1 ; if (d2 !=d1) { afficher d2 ; compteur ++ ;}
```

```
    }
    afficher compteur      /* c'est le nombre de diviseurs */
```

1.2.3. Nombre premier et diviseurs

1.2.3.1. Définition d'un nombre premier

Parmi tous les nombres, ceux que l'on appelle nombres premiers vont jouer un rôle essentiel. Par définition, un nombre (positif) est premier s'il admet exactement deux diviseurs, à savoir 1 et lui-même. Notamment 1 n'est pas premier, puisqu'il n'a qu'un diviseur. Les premiers nombres premiers sont 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, etc.

1.2.3.2. Propriété du plus petit diviseur

Le plus petit diviseur (autre que 1) d'un nombre a (supérieur à 1) est un nombre premier.

En effet, si ce plus petit diviseur d n'était pas premier, il admettrait un diviseur autre que 1 et lui même, soit d', avec $1 < d' < d$. A son tour d', déjà diviseur de d, diviserait le nombre a, et d ne serait plus le plus petit. Contradiction. Finalement ce nombre d est forcément premier.

On sait déjà qu'un nombre premier a comme plus petit diviseur autre que 1 lui-même, à savoir un nombre premier. Mais si un nombre (>1) n'est pas premier, il admet toujours au moins trois diviseurs, notamment un diviseur (autre que 1 et lui-même) qui est premier et c'est ce diviseur qui est le plus petit diviseur après 1.

1.2.3.3. Un théorème d'Euclide

Il existe une infinité de nombres premiers.

Pour le prouver, faisons un raisonnement par l'absurde. Supposons qu'il n'existe qu'un nombre fini de nombres premiers. Prenons le plus grand d'entre eux : P. Puis formons le nombre

$$Q = P\,! + 1 = 2 \times 3 \times 4 \times 5 \times \ldots \times P + 1$$

Ce nombre Q n'est pas divisible par 2 puisque la division euclidienne donne comme reste 1. Il n'est pas divisible par 3 non plus, à cause du reste 1. Et il en est de même jusqu'à P. Le nombre Q n'est divisible par aucun nombre premier, puisqu'ils sont tous supposés inférieurs ou égal à P. Mais on sait qu'un nombre non premier admet toujours un diviseur premier (c'est le plus petit diviseur autre que

1). Le nombre Q est donc premier, et supérieur à P qui était supposé être le plus grand. Contradiction. Notre supposition était fausse. Il existe un nombre infini de nombres premiers. Précisons que cette démonstration, attribuée à Euclide, date de 2300 ans.

Exercice 1
Le nombre mystérieux

D'un nombre à quatre chiffres on sait seulement que les deux premiers chiffres sont les mêmes, que les deux derniers chiffres le sont aussi, et que le nombre est un carré parfait. Trouver ce nombre en vérifiant qu'il est unique.

Ce nombre s'écrit *aabb*, avec a compris entre 1 et 9 et b entre 0 et 9. Sa valeur est $1000\,a + 100\,a + 10\,b + b = 1100\,a + 11\,b = 11\,(100\,a + b)$. Le nombre étant un carré parfait, il doit contenir 11^2, $100\,a + b$ doit contenir 11 en facteur, soit $100\,a + b = 11\,q$. Le nombre $100\,a + b$ est à trois chiffres et s'écrit $a0b$. Si ce nombre est divisible par 11, le nombre $100\,a + b - 99\,a = a + b$ l'est aussi. Cela donne comme couples (a, b) possibles : (2, 9), (3, 8), (4, 7), …, (9, 2). Mais un carré parfait ne peut que se terminer par 0, 1, 4, 5, 6, 9. Il reste quatre couples, donnant les nombres 2299, 5566, 6655, 7744. Le nombre 5566, divisible par 2 mais pas par 4 ne convient pas, tout comme 6655 divisible par 5 mais pas par 25. D'autre part $2299 = 11^2 \times 19$ ne peut pas convenir. Il reste 7744, qui est le carré de 88. On a bien une solution unique.

1.3. Pgcd et ppmc de deux nombres

1.3.1. Diviseurs communs et pgcd

Considérons deux nombres a et b (entiers positifs). Les diviseurs de a, comme ceux de b, sont en nombre fini (sauf si un nombre est nul). Prenons leurs diviseurs communs, eux aussi en nombre fini. Le plus grand de ces diviseurs communs est appelé le pgcd de a et b.

Le pgcd intervient implicitement lorsque l'on simplifie une fraction. Prenons par exemple la fraction $\frac{210}{392}$. Pour la simplifier, on divise en haut et en bas par le même nombre. Et tant que c'est possible. Dans le cas présent, on divise en haut et en bas par 2, puis par 3, puis par 7 :

$$\frac{210}{588} = \frac{105}{294} = \frac{35}{98} = \frac{5}{14}.$$

En résumé, on a divisé par le plus grand diviseur commun à 210 et 588, qui n'est autre que $2 \times 3 \times 7 = 42$.

1.3.2. Multiples communs et ppmc

Inversement, si un nombre *a* est un diviseur d'un nombre *m*, on dit que *m* est un multiple de *a*. Les multiples de *a* sont tous de la forme $m = k\,a$, avec *k* entier relatif. Les multiples d'un nombre (autre que 0) sont en nombre infini. Par exemple les multiples de 7 sont :

…, –28, –21, –14, –7, 0, 7, 14, 21, 28, …

Prenons maintenant les multiples positifs communs à deux nombres, et notamment le ppmc, le plus petit multiple commun positif. Les multiples du ppmc de *a* et *b* sont aussi des multiples communs à *a* et *b*. Notamment les multiples communs sont en nombre infini.

1.3.3. Programme pour avoir le ppmc

1.3.3.1. Ppmc de deux nombres

On se donne les deux nombres *a* et *b*. Puis on prend les multiples successifs de *a*, soit *a*, 2*a*, 3*a*,… jusqu'à ce que l'on tombe sur un multiple de *b*. On a alors le ppmc. D'où le programme :

```
multiple= a ;
while (multiple%b !=0)  multiple+= a;
afficher multiple  /* c'est le ppmc */
```

Exercice 2

Faire le programme donnant le ppmc de N nombres

On commence par chercher le ppmc des deux premiers nombres, puis on détermine le ppmc du ppmc précédent et du troisième nombre, et ainsi de suite. Le programme suivant permet d'avoir le ppmc de *N* nombres *a*[], ici pris au hasard.

```
#define N 6      /* declarations préliminaires */
int ppmc(int aa, int bb);
int ppmcN(int *a,int n);
int a[N];

int main()    /* programme principal */
{  int i, resultat;
```