１０進BASICのプログラミングで数学の勉強を

登場人物
T　パソコンのプログラミングを教えると見せて、数学を教えようとしている
　ちょっとおたっきーな先生。
S　BASICのプログラミングを学ぼうとしている初心者の高校生

【PART１　整数】

1　割り算のプログラムを作る

T：割り算をして商と余りを求めるプログラムを考えてみよう。例えば、23÷7というのはどうすればいい？

S：単純に、print 23/7 とすると、3.2857142…となってしまってまずいですね。これだと商はわかるんですが…

T：割り算の原理を考えてごらん。23÷7とは23が7で何回取れるかということ。

S：1回取ると、23－7＝16
2回取ると、16－7＝9
3回取ると、9－7＝2
もう取れないから余りは2ということになります。

T：ではこれをプログラミングしてみよう。

```
prg1
input a,b          ……> 割る数と割られる数の入力
10 let a=a-b       ……> a－bをあらためてaとおく
   if a<0 then
     let r=a+b     ……> a＜0のときの処理。a＜0のときはその1つ前の
   goto 20             aの値が余りである
   else
   let k=k+1       ……> a≧0のときはさらにa－bを計算。何回a－b
   goto 10             を計算したかを数える（これが商）。
   end if          ……> if構文終了
20 print k,r       ……> 商と余りの出力
end
```

2　整数はユークリッド整域である

T：このプログラムの原理は整数aはbで何回引かれるか数えるということ。そしてこれ以上引かれなくなったら（つまり割る数より小さくなったら）残った数が余りであるということです。

S：つまり、$a \div b = q \cdots r$ のとき、
$a - bq = r$ $(b > r)$
（a から b を q 回引く。答えが b より小さくなったら終わり）
という式で割り算を考えているんですね。

T：そう。そして、a, b に対して、q, r はただ一通りに決まることに注意しよう。まとめると、

> a を b で割ったとき、q, r を用いて、
> $a = bq + r$ $(r < b)$　と表せる。

1つ問題をやってみよう。

＜練習問題１＞
11で割ると7余り、5で割ると3余る自然数がある。これを11×5で割ったときの余りを求めよ。

S：この自然数を x とすると、$x = 11q + 7$ と置けるし、また、$x = 5s + 3$ とも置けますね。この後はどう考えるのだろう。

T：最初の式を変形して2番目の式にするのがポイント。
$x = 11q + 7 = 5q + 6q + 3 + 4 = (5q + 3) + 6q + 4$
この式が、$5s + 3$ の形になるのだから？

S：$6q + 4$ が5の倍数になればいいんですね。ということは、例えば $q = 1$ で10になりますから、$x = 11 \times 1 + 7 = 18$ などが答えですね。すると余りは18でいいんですか。

T：答えだけだったらそれでいいけれど、一般にはどういえるだろう。
$6q + 4$ が5の倍数なので、$6q + 4 = 5r$ とおくと、さっきと同じような考えから、
$6q + 4 = 5q + (q + 4)$ より、$q + 4$ は5の倍数。
そこで、$q = 5n - 4$ として x の式に代入すると？

S：$x = 11(5n - 4) + 7 = 55n - 37$　これは55で割ると余りが－37である数？

T：さっきのプログラムを思い出してごらん。余りがマイナスになるということは引きすぎなわけですよ。

S：ああそうか。つまり、$x = 55n' - 37 + 55 = 55m + 18$ つまり x は55で割って18余る数全体ということですね。

T：そうですね。ところで、この問題は次のようにして解くこともできます。
$x = 11q + 7$　…①　　$x = 5s + 3$　…②
とおいて、①×10－②×11を計算すると、
$x = 55(s - 2q) - 37$　これをまとめなおして、$x = 55n + 18$ と表せます。

S：どうやって10と11をかけて引けばうまくいくとわかるのでしょう。

T：①と②に何かをかけてひくことによって、x＝(55の倍数)＋(数) という格好にしたいので、①に$5m$ をかけて、②に$11n$ をかけて、それらをたして1になるような数を見つけるんだ。つまり、$5m + 11n = 1$ となる整数 m, n を見つければいいんだ。

S：$m = -2, n = 1$ が1つの解ですね。でもこれはどんなときでもすぐ見つけることができるんですか。

T：これは整数の重要な性質なので、次の回でもう少し詳しく話そう。

3　$mx + ny = d$ は整数の大切な性質

T：では、$mx + ny$ (m, n, x, y は整数) について考えてみよう。例えば、$m = 2, n = 5$ として、$2x + 5y = 1$ となる整数 x, y はあますか？

S：…$x = 3, y = -1$ などがあります。

T：そうだね。じゃあ、$m = 2, n = 4$ のときはどうなるかな？

S：$2x + 4y = 2(x + 2y)$ となり、これは絶対2の倍数なので、1にはなりません。

T：うん。じゃあどんなときに、$mx + ny = 1$ になると思う？

S：m, n の最大公約数が1のときですか？

T：うん。そんな気がするね。m, n の最大公約数が1のとき、m と n は互いに素という。

S：でも、例えば、$23x + 27y = 1$ を見つけろなんていうと大変です。

T：じゃあその辺のところをパソコンを使って考えてみよう。今君が言った、23と27でやってみる。まず23を何倍かした数を27で割った余りを求めるプログラムを作ってみよう。前のプログラムを利用すればできるが、ここで少し工夫してみる。
a を b で割った商は、わざわざｉｆ文を使わなくても切捨て関数「int()」を使えばうまくいく。（例　int(3.6)＝3など）

S：じゃあ、23を7で割った商は、int(23/7)でいいのですね。すると余りは…

T：$a = bq + r$ を使う。

S：そうか。$r = a - bq$ だから、r＝a－b*int(a/b)でいいんだ。するとプログラムは、

```
prg2
inpur a,b          ……> 割る数と割られる数の入力
let q=int(a/b)     ……> 商の計算
let r=a-b*q        ……> 余りの計算
print q,r          ……> 出力
end
```

T：さっきのプログラムよりだいぶすっきりしたね。ではこれを使って考えよう。a を23、b を27として、23の何倍かを27で割った余りを求めるプログラムにしてごらん。

S：できました。23の1倍からｋ倍までの数を27で割った余りを求めるようにしました。

T：ではこれを使ってやってみよう。

```
prg3
let a=23           ……> 割られる数
let b=27           ……> 割る数
input k            ……> k倍までする
for i=1 to k       ……> 1からkまで1つずつ繰り返す（for ～next文）
let q=int(a*i/b)   ……> 商
let r=a*i-b*q      ……> 余り
print i;q;r        ……> 出力
next i
end
```

ｋに27を入れます。すると、$23 \times 1, 23 \times 2, 23 \times 3, \cdots 23 \times 27$ の27個の数について、27で割った商と余りを求めることになります。出力結果を見てみよう。どんなことがいえるだろう。

i	商	余り
1	0	23
2	1	19
3	2	15
4	3	11
5	4	7
6	5	3
7	5	26
8	6	22
9	7	18

i	商	余り
10	8	14
11	9	10
12	10	6
13	11	2
14	11	25
15	12	21
16	13	17
17	14	13
18	15	9

i	商	余り
19	16	5
20	17	1
21	17	24
22	18	20
23	19	16
24	20	12
25	21	8
26	22	4
27	23	0

T：この表の余りに注目してごらん。同じものがあるかい？

S：あっ、同じものはありません。０～26まで27個の余りが1回ずつでています。

T：そうすると、もし、このあと、i を28として、23×28 の余りを求めるとどうなると思う？

S：また違う余りがでるんでしょうか。パソコンでやってみましょう。

T：いや、ちょっと待て。まず考えてからパソコンを実行しよう。前に出てきた重要な式 $a=bq+r$ を思い出そう。このとき、b , r にはどんな関係があった？

S：余りは割る数よりも小さい。つまり、$r<b$ です。

T：ということは27で割った余りは、27よりも小さい。
ところで、今の表ではもう27種類の余りがでているよ。

S：ということは、次に出てくる余りは当然すでに出てきた27種類の余りのどれかになるのですね。

T：そういうこと。こういう考え方を「部屋割り論」というんだ。では、一般の場合について証明してみよう。

【定理１】

m と n が互いに素のとき、m を k 倍$(1\leqq k\leqq n)$したものを n で割った余りはすべて異なる。

証明

k が i , j $(1\leqq i<j\leqq n)$ のとき、mi 及び mj を n で割った余りが等しくなったとすると、　$mi=nq+r$

$mj=np+r$

このとき、$m(j-i)=n(p-q)$

ここで、 m , n は互いに素なので、$j-i$ は n の倍数でなければならない。

ところが、i , j の条件より、これは n の倍数にならない。

よって、余りはすべて異なる。

余りの個数は n 個なので、0 ～ $n-1$ までの数になっている　㊮

このことによって、m と n が互いに素のとき、 $mx\,(1\leqq x\leqq n)$ を n で割って余りが１になるものがあるので、その商を y とすると、

$mx=ny+1$　となり、$mx-ny=1$ が成り立つ。ここで、$-y$ をあらためて y とおけば、$mx+ny=1$ となる x , y があることがわかる。

以上をまとめると、

【定理２】

m と n が互いに素であるとき、
$mx+ny=1$ を満たす整数 x , y が存在する。

これも整数の大切な性質なんだ。

4　任意のイデアルが主イデアル

S：m と n が互いに素のとき、$mx+ny=1$ とできるということはわかったのですが、これからどんなことがいえるのでしょうか。

T：では、まず、いくつかの m , n の組について、$mx+ny$ がどんな数になるのかパソコンを使って調べてみよう。 x , y は整数なので、どちらも-5 から５まで変化させてそれに対応する、$mx+ny$ の値を表示するプログラムを作ってごらん。

S：for ～next 文を使えばいいですね。

prg4

```
input  m , n
for x=−5  to  5
 for  y=−5  to 5
  print  m*x+n*y;
 next y
next x
end
```

T：出力結果を見てどんなことがわかる？

S：m と n が互いに素のときは、すべての整数が現れるような気がします。互いに素でないとき、例えば $m=27$, $n=21$ としてみたのですが、出てくる数は当然全部３の倍数です。ただ、３の倍数すべてが現れるのかどうかは…

T：なかなかいいね。ではこのことを証明してみよう。
整数全体の集合を Z、$mx+ny$ で表される数の集合を $A=\{mx+ny \;\; x,y\in Z\}$
とする。このとき、A の要素は必ず整数なので、

$A\subset Z\cdots①$

また、m , n が互いに素のとき、定理１から$mx+ny=1$ を満たす整数 x , y が存在したから、$1\in A$ 。

ということは、$mx+ny=1$ とするx , y をk 倍したものを考えると、

$mkx+nky=k(mx+ny)=k\;\in Z$　k は任意の整数なので、

$A\supset Z\cdots②$

①②から $A=Z$ がいえた。

m , n が互いに素でないときは、m , n の最大公約数を d とすると、

$mx+ny=d(m'x+n'y)$ となり、m' とn' は互いに素なので、$\{m'x+n'y\}$ で表される数全体は Z になる。ということは、$\{mx+ny\}$で表される集合は d の倍数全体の集合に一致することになる。

S：難しいけれど、パソコンを使って眺めてみると理解できるような気がします。

T：難しいついでにもう一ついうと、
$A=\{mx+ny\}$ で表される集合を m と n の2項で生成されるイデアルというんだ。
例えば、$2x+3y$ は２と３から生成されるイデアルだ。

S：そうすると、$\{2x+3y+5z\}$ なんていうのは、２，３，５の3項で生成されるイデアルというのですか。

T：そういうこと。一般には、
$m_1x_1+m_2x_2+\cdots m_nx_n$ （x_1 , x_2 , x_3, $\cdots$,x_n は整数）というように表された整数の集合を、整数環 Z のイデアルというんだ。
特に、項が１つのイデアルつまり $\{mx\}$ を単項イデアルまたは主イデアルというんだ。

S：単項イデアルというのは、ある数 m の倍数全体の集合なんですね。

T：そう。さて、ここで整数の性質として重要なことをまとめておきます。
さっき調べたように、２と３など互いに素な数で生成されるイデアルは整数全体の集合に一致した。で、この整数全体の集合は、１という単項で生成されるイデアルといってもいいね。

S：$\{1\cdot m\}$ ということですね。

T：同様に、例えば27と21で生成されるイデアルはその最大公約数３で生成される単項のイデアルになっている。
一般に整数の世界では、どんなイデアルも必ず主イデアルになることが言えるんだ。
このような世界を「主イデアル整域」と呼びます。

【補足】

イデアルの定義がおおざっぱだったので、補足する。
環 R の部分集合 I が R のイデアルであるとは、
① I は加群
② $a\in I$, $x\in R$ のとき、$ax\in I$
つまり、I 内の元による1次結合もまた I の元である

5　2数の最大公約数を求めるプログラム

T：では、今度は４で話題になった最大公約数を求めるプログラムを作ってみよう。
まず、最大公約数を考える前に、ある数の約数を求めるプログラムを作ってみよう。

S：ある数を a とすると、a より小さい数で１つ１つ割っていって、余りが０ならば約数であると考えればいいんですね。
a を b で割った商が、$q=int(a/b)$ 、余りが $r=a-bq$ だったので、プログラムは次のようになりました。

prg5'

```
input a
 for  i=1 to  a
  let  r=a−i*int(a/i)  ·····➢ 余り
  if r=0  then print i;  ·····➢ 余り０なら印字
 next i
end
```

T：これでいいんだけれど、ちょっと時間がかかりそうだ。2行目の for ～next は１からａまでやる必要はないぞ。
例えば24の約数を求めるとき、１と24、2と12、3と8、4と6というように、ペアで求めていくだろう。

S：そうか。　「q が a の約数　$\Longrightarrow$　a/q もa の約数」
なんですね。ということはどこまでやればいいんだろう。$q=a/q$ と考えて、
$q^2=a$ つまり、$\sqrt{a}$ まで考えればいいんですね！

prg5

```
input a
let  n=int(sqr(a))
for  i=1 to  n
  let  r=a−i*int(a/i)
  if r=0  then print i;a/i
 next i
end
```

T：最大公約数の話に行く前に、このプログラムを応用して３つのことをやってみよう。

① 素因数分解

T：prg5を少し変えれば素因数分解のプログラムができる。やってみてごらん。

prg6

```
input a
let n=int(sqr(a))
for i=2 to n
10  let r=a-i*int(a/i)
 if r=0 then
   print i;"*"          ……> 素因数の表示
   let a=a/i            ……> a/i について素因数分解を繰り返す
   goto 10
 else
 end if
next i
print a                 ……> 残りの因数の表示
end
```

出力例　？1998　2*3*3*3*37*1

② 素数の判定

T：ある整数が素数であるかどうかを判定するプログラムを作ってごらん。

S：これは簡単そうです。prg5で、全部のiで割りきれなければ素数とすればいいんですね。

prg7

```
input a
let n=int(sqr(a))
for i=2 to n
  let r=a-i*int(a/i)
 if r=0 then
  print "素数でない"
  goto 10
 end if
next i
print "素数である"
10 end
```

T：これを応用すれば素数表なども作成することができます。

③ 完全数

T：次にprg5の応用として、約数の和（自分自身を除く約数の和）を求めるプログラムを作ってみよう。
これは、prg5に和を求める部分を1行追加すれば良い。

S：できました。

prg8

```
10 input a
let n=int(sqr(a))
for i=1 to n
  let r=a-i*int(a/i)
 if r=0 then
  print i;a/i
  let s=s+i+a/i          ……> これがprg5から新たに加わった部分
 end if                       約数の和を求めている
next i
print "次分自身を除く約数の和";s-a
let s=0
goto 10                 ……> 何回もいろいろな数で試せる
end
```

S：最後にgoto文をつけて何度でもできるようにしました。

T：ではこれを活用してみよう。$a=6$としてみてごらん。

S：自分自身を除く約数の和が6と表示されました。

T：このように、自分自身を除く約数の和が自分自身になるような数を「完全数」というんだ。他の完全数をこのプログラムを使って見つけてごらん。

S：28がうまくいきました。後は、なかなか見つかりません。

T：このプログラムで1つ1つの数を入力していったのでは大変だね。そこで、prg8を少し改良してみよう。

prg9

```
10 input n
for a=1 to n            ……> 1からnまでの整数について1つ1つ
 let n=int(sqr(a))           調べる。
for i=1 to n
  let r=a-i*int(a/i)
 if r=0 then
   let s=s+i+a/i
 end if
next i
 if a=s-a then print a;s-a  ……> 完全数なら表示する
 let s=0
next a
end
```

T：これで実行してみよう。

S：n=100とすると、1，6，28が表示されました。1から100まででではこの3つが完全数ですね。

T：普通は1を除くけれど、まあ今は入れておくことにしよう。
1000まででではどうなる？

S：1，6，28，496の4つです。
10000までだと、1,6,28,496,8128の5つです。

T：さて、ここからパソコンを離れてみよう。ここに現れた5つの数には何か規則があるだろうか。

S：うーん。1を除けば各桁に1個ずつなのかな。

T：実は、この完全数の研究をしたのは紀元前のギリシャ時代の数学者ユークリッドなんだ。ユークリッドは、君がパソコンで求めた4つの完全数、6,28,496,8128を求めていたそしてその規則性も見つけていたが、8128の次の完全数を求めることはできなかった。この5番目の完全数が求められるのは、ユークリッドの後1500年後くらいなんだ。

S：今ならパソコンを使ってすぐ求められるんですよね。

T：そう。コンピュータは明らかに歴史の時間を縮めているといえるだろう。しかし、逆にいうと、紀元前の数学者達が何もない状態でこんなに凄いことをやっていたことに感動を覚えるね。
さあ、それでは8128の次の完全数の発見に挑戦してみよう。

S：まず、1を除いて、4つの完全数をそれぞれ素因数分解してみます。
127はさっき作ったprg6を利用したら素数でした。ということはこれを見ると、4つの完全数はどれも、$2^n\times$(素数)　という形になっています。

T：その調子。後ろの素数の部分ももう少し詳しく調べるんだ。

S：3,7,31,127　あっ、これに1を足すと皆2のn乗の形になる！
つまり、
$6=2\times(2^2-1)$
$28=2^2\times(2^3-1)$
$496=2^3\times(2^5-1)$
$8128=2^5\times(2^7-1)$
わかりました。完全数は一般に、
$2^{p-1}(2^p-1)$　（pは素数）…①
という形で表されるんだ！

T：えらい！といいたいところだが、2つ間違えていることがある。一つは、完全数ならば必ず①の形だ、という推論はいけない。他の形で表される完全数もあるかもしれない。だから、いうならば、①の形で表される数は完全数である、としなければいけない。そして、もう一つの間違えは……、実際、①の形から推測して第5番目の完全数を求めてごらん。

S：ええと、2,3,5,7 の次の素数だから、$p=11$ということで、完全数の「候補」は、$2^{10}(2^{11}-1)=2096128$ です。

T：じゃあ、これをprg8で完全数かどうか試してごらん。

S：自分自身を除く約数の和　2325392と表示されました。これは完全数ではないですね。

T：そうなんだ。ここで、この問題は1500年以上のブランクが置かれるんだ。
問題は、さっきS君が推測した、$2^n\times$(素数)という式。この式と、①の$2^{p-1}(2^p-1)$（pは素数）の式が矛盾しているんだ。確かに、pが2,3,5,7と素数のとき、2^p-1も素数になっているが、$p=11$のときはどうなるだろう。

S: $2^{11}-1=2047$ これをさっき作った、prg6の素因数分解のプログラムを使って調べてみると、なるほど、$2047=23\times89$となって、これは素数ではありません。
「pが素数　⟹　2^p-1は素数」と簡単にいってはいけないんですね。

T：そうなんだ。そこで、さっき君が出した式、

$N=2^{p-1}(2^p-1)$ …①　において、2^p-1 が素数のとき N が完全数になるという予想が立つ。これを証明してみよう。

S：こんなことどうやって証明するんですか？

T：これは大したことはない。数Ⅰの個数の処理のところで出てきた考えを使う。ちょっと練習問題をまずやってみようか。

＜練習問題＞

72の約数の個数と約数の総和を求めよ。

S：まず、$72=2^3\cdot3^2$ と素因数分解して、72の約数は、集合

$A=\{1,2,2^2,2^3\}$

$B=\{1,3,3^2\}$

から、1個ずつ取り出してかければいいのですね。そうすると積の法則で

$4\times3=12$ 個

T：そうだね。これは集合 A, B の直積を求めていることなんだね。

S：約数の総和は、

$(1+2+2^2+2^3)(1+3+3^2)$ を展開すれば各項の和が約数の総和でした。

よって、$15\times13=195$ です。

T：そう。その通り。じゃあさっきの命題もできるはずだ。

S：やってみます。

【定理３】

$N=2^{p-1}(2^p-1)$　(2^p-1 は素数) は完全数である。

証明

Nの、自分自身を除く約数の和は、

$$(1+2+2^2+2^3+\cdots2^{p-1})(1+2^p-1)-N$$
$$=(2^p-1)2^p-2^{p-1}(2^p-1)$$
$$=(2^p-1)2^{p-1}(2-1)$$
$$=2^{p-1}(2^p-1)=N$$

（示した）

S：ということは、この形の完全数は、2^p-1 が素数であるかどうかが問題なんですね。

T：そう。これで、17世紀のメルセンヌの領域までやってきた。

S：メルセンヌって？

T：17世紀の思想家で、2^p-1 型の素数について予想を立てたんだ。そこで、2^p-1 型の素数をメルセンヌ数ということがあるんだ。当時はもちろんコンピュータなんかないから大変だったと思う。

さあ、ではパソコンを使って、第5の完全数を求めてみよう。

S：prg7 の素数の判定プログラムを使います。2^p-1 を計算するのが面倒なので、prg7 を次のように変えます。

prg7'

```
input a
 let a=2^a-1
 let n=int(sqr(a))
 for i =2 to n
   let r=a-i*int(a/i)
   if r=0 then
   print "素数でない"
   goto 10
   end if
 next i
print "素数である"
10 end
```

T：ここでちょっと気をつけたいのは、十進ＢＡＳＩＣは通常は15桁の制度なので、オプションの設定を変えておきます（249桁）。

S：では実行してみます。

$p=11$ はだめでしたので、$p=12$ からやってみます。

$p=12$ のとき　素数でない

$p=13$ のとき　**素数である**

$p=14$ のとき　素数でない

$p=15$ のとき　素数でない

$p=16$ のとき　素数でない

$p=17$ のとき　**素数である**

$p=18$ のとき　素数でない

$p=19$ のとき　**素数である**

$p=20$ のとき　素数でない

$p=13,17,19$ のときがとりあえず素数になりました。

ところで、pが素数でないときには、2^p-1 は素数にならないと思うんですが本当でしょうか。

T：面白いところに気づいたね。パソコンを使いながらこんなふうに新しい興味が起こってくるのはとてもいいことだ。今、君がいったことは、

pが合成数　⟹　2^p-1 も合成数

ということです。じゃあ、ちょっと横道にそれるがこれを証明してみよう。

X^p-1 の因数分解を使ってみるんだ。

証明

pが合成数のとき、$p=kq$（q は素数）とおくと、

$$2^p-1=2^{kq}-1=(2^k)^q-1$$
$$=(2^k-1)\{(2^k)^{q-1}+(2^k)^{q-2}+\cdots+1\}\quad\cdots①$$

pは合成数なので、$k\neq1$, $q\neq1$

よって、①の２つの因数は１になることはない。

このことより①は合成数であることがわかる。

T：ということで5～７番目の完全数はわかったね。わかったところまで並べてみよう。

S：
$$6=2\times(2^2-1)$$
$$28=2^2\times(2^3-1)$$
$$496=2^4\times(2^5-1)$$
$$8123=2^6\times(2^7-1)$$
$$33550336=2^{12}\times(2^{13}-1)$$
$$8589869056=2^{16}\times(2^{17}-1)$$
$$137438691328=2^{18}\times(2^{19}-1)$$

T：あっという間に求めてしまったね。パソコンというタイムマシンで、歴史をひとっとびという感じだ。

S：完全数は無限にあるんですか。

T：結局、2^p-1 型の素数が無限にあればいいんだけれど…これは証明されていないんじゃないかな。

18世紀の大数学者オイラーは $p=31$ のときに2^p-1 が素数になることを発見したらしい。その後19世紀にはルカスというフランスの数学者が$p=127$ の時に素数になることを発見。

20世紀になってからは、コンピュータの世界に入っていて、現在発見されている最も大きい完全数は50万桁を超えているらしい。

S：πの計算と同じようにコンピュータの性能合戦ということなんでしょうか。

T：それもあるだろうけれど、どうすれば効率良く早く計算できるかという部分に数学が活躍しているんじゃないかな。

T：最大公約数の話にいくはずが、だいぶ道草をしてしまったね。では、いよいよ最大公約数を求めるプログラムを作ってみよう。まず、具体的な例を手計算で考えて、その原理を見つければいい。

S：例えば、24と60で考えます。まず、それぞれを素因数分解すれば良かった。

$24=2^3\cdot3$　　$60=2^2\cdot3\cdot5$

よって、$2^2\cdot3=12$ が最大公約数です。

これをプログラムにするには…素因数分解をして、共通の因数を求める…

T：ちょっとまった。今、24と60で考えたんだけど、例えば371と901なんていうときはどうする？

S：これは簡単には素因数分解が見つかりません。ではパソコンで…

T：いや、紀元前のユークリッドに倣って、まず頭と手計算でやってみよう。ヒントは $\frac{901}{371}$ を約分すると考えるんだ。

S：確かに、$\frac{901}{371}$ は2数の最大公約数で約分できますね。

T：整数の性質を思い出すんだ。

S：そうか。$901\div371=2$ 余り 159　ということは、$901=371\times2+159$　となる。

T：901 と371 の最大公約数を d とすると、901 は d の倍数で、301 も d の倍数だから、$901-371\times2$ も d の倍数となる。

S：そうか。ということは 159 も d の倍数なんだ。だから、371 と159 の最大公約数を考えればいいんだ。

T：そう。このことを分数でかくと、

$\frac{901}{371}=2+\frac{159}{371}$　なので、$\frac{901}{371}$ の約分のかわりに、$\frac{159}{371}$ の約分を考えればいいということだよね。

S：そうすると、371 と 159 で同じことをやって、　$371=159\times2+53$
今度は 159 と 53 で考える。　$159=53\times3$　あっ割りきれた。ということは159 と 53 の最大公約数は53、つまり、371 と 159 の最大公約数も 53、そして、901 と 371 の最大公約数も 53 なのですね。

T：そう。これをユークリッドの互除法というんだ。文字を使って説明しよう。
a と b が整数のとき、$a=bq+r$ $(r<b)$ の関係が成り立っていたね。このとき、
$r=a-bq$ とおくと、a , b は共に最大公約数 d の倍数になってるから、$a-bq$ つまり r も d の倍数ということ。
そこで、a , b の代りに b , r で考えていくわけだ。
確認のためにもう一つやってみよう。
714825 と 428571 の最大公約数をユークリッドの互除法で求めてごらん。

S：$714285=428571\times1+285714$
$428571=285714\times1+142857$
$285714=142857\times2$
よって、714285 と 428571 の最大公約数は 142857 です。

T：そうですね。ここにでてきた4つの数、
{142857,285714,428571,714285} は皆 142857 の倍数であり、この集合の最小元が、142857 というわけだ。このような集合を 142857 で生成されるイデアルといったのを覚えているかい。

S：つまり、2数の最大公約数を求める作業というのは、2数が含まれるイデアルの最小元を求めるということなのですね。

T：そう。で、整数は主イデアル整域なので、どんなイデアルも主イデアル。つまり、「イデアル＝ある数の倍数の集合」と考えれば簡単だ。
ちょっと荒っぽい言い方になるけれど、整数でなくても、$a=bq+r$ が成り立つ世界では、ユークリッドの互除法が使えるんだ。少し難しいけれど大切なことなので補足しておきます。

【補足：ユークリッド整域は主イデアル整域である】

環 R がユークリッド整域であり、I を R の任意のイデアルとする。
a を I の任意の元、 d を I の最小元とする。
$a\in R$ でもあるので、a は I の最小元 d を用いて、
$a=dq+r$ と表すことができる。$(r<d)$
$r=0$ ならば、$a=dq$ となり、a は d の倍数。つまり I は d で生成される主イデアルである。
$r\neq0$ ならば、$a-dq=r$ から、a , $dq\in I$ より、$r\in I$
これは d の最小性に反する。よって $r=0$ であり、I は主イデアル整域である。

ここであげたことは、まさにユークリッドの互除法そのものといってもいい事柄です。

S：整数以外にユークリッド整域になるものはどんなものですか。

T：例えば整式全体の集合がそうだね。ただ整式では大小関係をいうことができないから
$A=BQ+R$ (R の次数 $<$ Bの次数) と次数によって大小を表現する。
また、$\{m+ni \quad m,n$ は整数$\}$ という複素数の集合もそうだ。これはガウスの整数環という。この場合も大小関係はないので絶対値の大小で比較することになる。
つまり、整式にしてもガウスの整数にしても、ある元に対して、実数に対応するものを考えておかなければならないんだ。

S：ガウスの整数で一例を挙げてください。

T：例えば、5 と $1+3i$ に属するイデアルの最小元、簡単に言うと、2数の最大公約数を互除法で求めると、$5=(1+3i)(-1+2i)+2+i$ このとき、
$|1+3i|>|2+i|$ に注意する。
さらに、$1+3i=(2+i)(1+i)$ となり、最大公約数は $2+i$ となるわけだ。
つまりガウスの整数の世界では5 は素数ではなくて、$5=(2+i)(2-i)$ と因数分解されることがわかる。この辺を話すときりがないので、本題の2数の最大公約数を求めるプログラムを作ってみよう。

S：できました。
prg10

```
input a, b
10 let r=a-b*int(a/b)
 if r=0 then
  print  b
  else
  let  a=b
  let  b=r
  goto 10
 end if
end
```

余りの計算

値の入れ替え

T：ＯＫ。ユークリッドの互除法はセンター試験にもよく登場するね。
一応、この互除法でとりあえず§１の整数の勉強を一区切りとします。
いろいろなことをやったけれど、実は紀元前のユークリッドの範囲のことを出ていないといってもいい。それだけユークリッドは偉大だっていうことかな。
最後に宿題として、今の最大公約数を求めるプログラムを利用して、分数と分数の足し算の結果を分数で表示するものを作ってごらん。そうすればパソコンと整数はとりあえず卒業だ。

！分数の計算（一例）

```
print "a/b+c/d"
input prompt "a,b,c,d":a,b,c,d
let bb=b
let dd=d
call gcm(b,d,g)
let  e=a*dd/g+c*bb/g
let  f=bb*dd/g
let  ee=e
let  ff=f
call gcm(e,f,g)
print ee/g;"/";ff/g
end
  external sub gcm(x,y,g)
   10 let r=x-y*int(x/y)
      if r=0 then
      let  g=y
      else
      let  x=y
      let  y=r
      goto 10
      end if
   end sub
```

b , d の最大公約数を求めるサブルーチンを呼び出す

分子

分母

答えの印字（ｇで約分をした）

最大公約数を求めるサブルーチン（ｇｃｍとした。引数はx , y , g）

（1996　盛岡三高　下町壽男）