6.3 ユークリッドの互除法

完全数の話題に約数が出たので、今度は最大公約数を考えてみよう。

一般に知られている**最大公約数**の求め方は次のようなものだろう。

$$
\begin{array}{rcrr}
2 &) & 60 & 36 \\ \hline
2 &) & 30 & 18 \\ \hline
3 &) & 15 & 9 \\ \hline
 & & 5 & 3
\end{array}
$$

この場合、60 と 36 の最大公約数は除数に現れた数の積 $2 \times 2 \times 3 = 12$ である。ちなみに**最小公倍数**であれば、商に現れた数までの積 $2 \times 2 \times 3 \times 5 \times 3 = 180$ となる。

ところで、最大公約数は次のようにしても求められる。同じく 60 と 36 での例である。

$$
\begin{aligned}
60 &= 36 \cdot 1 + 24 \\
36 &= 24 \cdot 1 + 12 \\
24 &= 12 \cdot 2 + 0
\end{aligned}
$$

何をしているかと言えば、まず 2 数の大きい方（被除数）を小さい方（除数）で割って余りを求める。普通なら $60 \div 36 = 1$, 余り 24 と書くところを、気取って $60 = 36 \cdot 1 + 24$ と書いている。次にすることは、除数を被除数に昇格（？）させ、余りを除数に昇格させて、同様に割り算をし余りを求める。この繰り返しだ。繰り返しは、余りが 0 になった時点で終了だ。このとき、最後の除数が最大公約数になるのである。

不思議な顔をしているね。それならもう一丁、11 と 26 の最大公約数を求めよう。おそらく、見ただけで共通の約数がないので—こういう 2 数を**互いに素**と呼ぶ—最大公約数は 1 だと気付くだろう。では、いまと同じ手順を踏んでみよう。

$$
\begin{aligned}
26 &= 11 \cdot 2 + 4 \\
11 &= 4 \cdot 2 + 3 \\
4 &= 3 \cdot 1 + 1 \\
3 &= 1 \cdot 3 + 0
\end{aligned}
$$

余りが 0 になったとき、最後の除数は 1 である。よって最大公約数は 1、すなわち 11 と 26 は互いに素であることが分かる。

最大公約数を求めるこのアルゴリズムは**ユークリッドの互除法**と呼ばれている。この方法で最大公約数が求められる理由は、さほど難しい理論ではないが、この蝶道では厳密なところで時間を使わない。詳しく知りたければ、整数について書かれた書物を読んでもらいたい。

ユークリッドの互除法を用いて、最大公約数を求めるマクロを書いてみよう。

```
\newcommand\findgcm[2]{%
  \newcount\a \newcount\b \newcount\r
  \a=#1 \b=#2
  \loop
    \r=\a \divide\a\b \multiply\a\b \advance\r-\a
    \ifnum\r=0 \number\b
    \else \a=\b \b=\r
  \repeat
}
```

これで、『60 と 36 の最大公約数は`\findgcm{60}{36}`である。』と書けば『60 と 36 の最大公約数は 12 である。』が出力され、『11 と 26 の最大公約数は`\findgcm{11}{26}`である。』と書けば『11 と 26 の最大公約数は 1 である。』が出力される。

マクロ自体は単純で新しいことは何もない。余り`\r` を求め、それが何かに等しいかどうか判断するルーティンはお馴染みになった。ユークリッドの互除法では、割り算をし余りを求めることを繰り返し、余りが 0 になれば最大公約数が求められるはずであった。そのため、余りが 0 にならない間は`\loop`〜`\repeat` 命令が繰り返されなくてはいけないのだ。

さて、余りがあるうちは常に除数を被除数へ、余りを除数へ引き継いでいけばよい。これは 2 つの変数の交換と違って、どこかへ退避させる必要がある値はない。よって、単純に代入を順送りしているだけである。さて、余りが 0 になると除数である`\b` が出力され、それが求める最大公約数である。

ところで、人がユークリッドの互除法を使うとき、自然と 2 数の大きい数を小さい数で割り始めるはずだ。しかしこのマクロは $a < b$ である 2 数を引数に与えても正常に動くので安心してよい。

最大公約数の次は最小公倍数の番だ。ところで、最大公約数が分かれば最小公倍数は直ちに計算できることを知っているかね？　それはこういうことだ。

2 つの数を M, N としておこう。この 2 数の最大公約数を g とすると、$M = mg, N = ng$ と書いてよいだろう。そして m, n は互いに素であることも重要だ。よって、M, N の最小公倍数は mng であることが分かる。

では、TEX が M, N の値を受け取ったとき、これだけの情報から最小公倍数を計算するにはどうすればよいだろう？　簡単なことだ。$\dfrac{MN}{g}$、すなわち $\dfrac{MN}{GCM(M,\ N)}$ でよい。ちなみに、$GCM(M,\ N)$ は M, N の最大公約数を意味する。そこで最小公倍数を求めるマクロは次のようになる。

```
\newcommand\lcm[2]{%
  \newcount\a \newcount\b \newcount\r
  \newcount\m \newcount\n
  \m=#1 \n=#2 \a=\m \b=\n
  \loop
    \r=\a \divide\a\b \multiply\a\b \advance\r-\a
    \ifnum\r=0 \multiply\m\n \divide\m\b \number\m
    \else \a=\b \b=\r
  \repeat
}
```

これで、『60 と 36 の最小公倍数は`\lcm{60}{36}`である。』と書けば『60 と 36 の最小公倍数は 180 である。』が出力され、『11 と 26 の最小公倍数は`\lcm{11}{26}`である。』と書けば『11 と 26 の最小公倍数は 286 である。』が出力される。

最小公倍数の求め方からすれば、マクロ`\lcm` は関数`\findgcm` を利用するようにするだけなのだが、`\findgcm` は最大公約数を出力するように作ってしまったので、多少手直ししたマクロ`\lcm` にせざるを得なかった。ただ、基本は`\findgcm` を使っているので、最大公約数を見つけたら、引数の積を最大公約数で割るように書き換えただけである。