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 数の大きいほう（被除数）を小さいほう（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、今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、私たちの旅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プログラムを組んでみよう。

programming list [findgcm.cpp]

```
 1: #include <iostream>
 2:
 3: int gcm(int aa, int bb) {
 4:     int t;
 5:     while(aa % bb) {
 6:         t = bb;
 7:         bb = aa % bb;
 8:         aa = t;
 9:     }
10:     return bb;
11: }
12:
13: int main() {
14:     int a, b;
15:     while(1) {
16:         std::cout << "input 2 numbers a,b : "; scanf("%d,%d", &a, &b);
17:         std::cout << "GCM(" << a << ", " << b << ") = " << gcm(a, b)
                                                                << std::endl;
18:     }
19:
20:     return 0;
21: }
```

プログラム自体は単純だ。先に main 関数から見ておこう。

2 数の最大公約数を求めるので、14:行目で 2 変数 a, b を用意している。

実質的に 16:行目が入力要求だが、ここでは書式を"%d,%d"に決めている。つまり “60,36” や “11,26” のように入力せよということだ。“,” で区切らない入力では正しい結果が出力されない。ちなみに “,” の後に空白があっても大丈夫である。

17:行目で最大公約数の表示をしている。表示には関数 gcm() を利用しているので、入力された 2 数が関数 gcm() へ引き渡されるのだ。

では、関数 gcm() へ目をやろう。3:行目から分かるように、この関数は整数値 2 つを受け取り、ひとつの整数値を返す。

4:行目にある変数 t は、一時代入用だ。効用はすぐ後で分かるはずだ。

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。そこで 5:行目にあるように aa % bb が 0 にならない限り、while 文が繰り返されるようにしてある。

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。それが 6:–8:行目の一連の作業なのである。この手法は前章の [contfrac.cpp] と全く同じことをしていること

を確認してほしい。[contfrac.cpp] では分子・分母の入れ換えだったのに、ここでは除数と余りの順送りなのだ。同じ記述でも違う作業をしてしまうから、プログラミングは厄介なのだ。自分が書いたプログラムが、半年後には理解不能になる理由は、こんなところにもある。重要なプログラムを組むなら、細かすぎるほどのコメントを記述しておくべきだ。

さて、余りが 0 になると while 文を抜ける。このとき最後の除数は bb に代入されているので、10:行目で bb の値を返せば、それが求める最大公約数である。

EX. 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。プログラムは $a < b$ である 2 数が入力されても正常に動く。その理由を考えよ。

EX. ユークリッドの互除法の手続きは再帰的である。関数 int gcm(int aa, int bb) を以下のものと差し替えて実行してみよ。再帰の考え方が理解できただろうか。

```
int gcm(int aa, int bb) {
    if(aa % bb > 0)
        return gcm(bb, aa % bb);
    else
        return bb;
}
```