6 . . . の豆旅

6.1 完全数

6 は興味深い数の一つだ。それはどういうことかというと、6 の約数は 1, 2, 3, 6 だが、

$$1+2+3=6$$

となっている。この性質を満たす数は 6 の近辺にはない。たとえば 8 は、約数として 1, 2, 4, 8 を持つが、明らかに $1+2+4 \neq 8$ である。約数をもれなく取り出すことができれば、6 の次にこの性質を満たす数が 28 であることを知るのは容易（たやす）い。実際、28 の約数は 1, 2, 4, 7, 14, 28 であるから

$$1+2+4+7+14=28$$

になっている。この性質を持つ数を**完全数**と呼ぶ。

性質を誤解のないように言えば、完全数とは

自分自身を除く約数の総和が自分自身に等しい数

であると言える。なんと、日本語にする方が式で説明するより難しくなってないかい？　数学とはそんなものだ。

それなら、28 の次の完全数はいくつだろうか。ちょっと計算すればすぐに見つかると思うだろう。悪いね。そんな簡単には見つからないのだ。よほど根気よくなければ、次の完全数を見つけるのは難しい。根気が勝負になるときこそコンピュータの出番というわけだ。

いきなり完全数を求めるプログラムを組むのも大変だろうから、まずは約数を見つけるプログラムから見ていこう。

programming list [DispOfCM.java]

```
1: import java.util.Scanner;
2:
3: public class DispOfCM {
4:
5:     public static void main(String[] args) {
6:         Scanner s = new Scanner(System.in);
```

```
 7:
 8:         System.out.print("input your candidate: ");
 9:         int n = s.nextInt();
10:
11:         System.out.print(n + " := ( 1,");
12:         for(int i = 2; i <= n / 2; i++) {
13:             if(n % i == 0) {
14:                 System.out.print(" " + i + ",");
15:             }
16:         }
17:         System.out.println(" " + n + " )");
18:     }
19: }
```

ここでは新たな命令は一切登場しない。そこで、ちょっと出力に凝ってみた。凝ったとはいえ、たとえば 6 の約数なら "6 := (1, 2, 3, 6)" と表示するようにしたに過ぎない。:= は特別の記号ではなく、ただ、$6 = (1,\ 2,\ 3,\ 6)$ のような等号の使い方に違和感があったので := としている。

プログラムはある数を入力すると、約数をすべて表示することになってる。ところである数が n のとき、約数に 1 と n があるのは当然なので、これらは最初と最後に表示することにした。

9:行目までは見なれた風景だ。まず、9:行目で約数を調査される数が取り込まれる。

さて、11:行目は明らかな最初の約数 1 の表示である。目には (が不揃いに映るだろうが、注意して見てほしい。この時点では、画面には入力した数 n を用いて "** := (1," まで表示されるはずだ。

12:行目の `for` 構文は、`n` を 2 から順々に割っていくものだ。約数を求める行為は素数を求める行為と違うので、2 で割れても 4 で割れるか試す必要があるし、1 ずつ大きな数で割る必要もある。ただし、`n / 2` までの数で試せばよい。それより大きい数が約数になることはないのだから。

13:行目は `n` が `i` で割れるかどうかの確認だ。割れれば約数であるから 14:行目で表示し、割れなければ何もせず次の `i` を試すのだ。

そして 17:行目で最後の約数、つまり自分自身を表示すれば終了だ。ここでも) が一見不揃いに見える点に注意だ。**Jaba** は ".`print(`" と対の ")" をちゃんと見分けることができる。大したものである。

また、プログラムは約数を列挙するものだが、素数を見つける役にも立つ。なぜなら、素数は 1 と自分自身しか約数を持たない数だから、たとえば 17 の入力には "17 := (1, 17)" が返ってくるのだ。

TRY! 大きな数の約数を調べてみよ。たとえば自分の誕生日を西暦年からつなげると大きな数になる。2005 年 4 月 19 日生まれなら 20050419 だ。さて君たちの誕生日はどれぐらい約数を持つだろうか。それとも素数だろうか。

ここに提示した約数を求めるプログラムは、単純きわまりないアルゴリズムであるだけに無駄が多い。当然のことながら、たとえばある数 n が i で割れたなら、その商である n/i も約数である。つまり約数は基本的に二つ同時に見つけることができる。基本的と表現したのは、49 が 7 割れたからと言って、49/7 が別の約数とは言えないからだ。しかし、除数 i と商 n/i を一緒に表示してしまえば、割り算は $\sqrt{n}$ まで試せばよいことになる。これは以前、素数を求めた際に使った手法だし、計算量を減らす効果も十分だ。

TRY! この考え方でプログラムを書き換えてみよ。

だが、完全数を求めるためには、これではまったく不十分なのだ。あとで分かるように、完全数は素数とは較べものにならないくらい稀にしか現れない。つまり、計算量が減ること自体は喜ばしいけれど、結局ほとんどの数で調査が不発に終わる。効果的に調べるには、計算量を減らすのではなく、調査しなくてもよい数を飛ばすことである。