6. . . の旅

6.1 完全数

6 は興味深い数のひとつだ。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。例えば 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、日本語にするほう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.cpp]

```
#include <iostream>

int main() {
    int n, i;
    while(1) {
        std::cout << "input your candidate : "; std::cin >> n;
        std::cout << n << " := ( 1,";
```

```
 8:         for(i = 2; i <= n/2; i++) {
 9:             if(n % i == 0) {
10:                 std::cout << " " << i << ",";
11:             }
12:         }
13:         std::cout << " " << n << " )" << std::endl;
14:     }
15:
16:     return 0;
17: }
```

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、例えば 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。

4:行目で候補の数を代入する変数 `n` とカウンタ変数の `i` を用意した。

5:行目は繰り返しの入力ができるように、お馴染みの `while(1)` を用いている。

6:行目は入力要請だ。

さて、7:行目は明らかな最初の約数 1 の表示である。ちょっと変な感じの出力形式だが、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。

8:行目の `for` 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。

9: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 10: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 13:行目で最後の約数、つまり自分自身を表示すれば終了だ。

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、例えば 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。例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。2007 年 1 月 24 日生まれなら 20070124 だ。さて君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。