4. . . の夢見鳥

4.1 ゴールドバッハの予想

4 はちょっとだけ面白い性質を持つ。それは

$$4 = 2 + 2 = 2 \times 2$$

のように同じ数どうしで、和と積に分解できる唯一の自然数だからである。さらに使われている数 2 は、唯一の偶素数でもある。数学は唯一という言葉が好きである。

さて、ここで**素数**という言葉が登場した。そもそも私が夢見鳥となって空中を舞うはめになったのは、TEX も時代とともに進化するものだと悟ったからである。TEX ってマクロ以外にも組版中にプログラミングできるんだ。

一般に自然数は、素数の積に**一意的に**分解できる。一意的とは、掛け算の順番を無視すれば、ただ一通りの積にしかならないことを意味している。しかし自然数は、素数の和に一意的には分解できない。それは

$$10 = 5 + 5 = 3 + 7 = 2 + 3 + 5$$

を見れば分かるように、いくつかの分解の仕方がある。

素因数分解のように一意的に解が求められるものは、分かりやすいし研究の対象にもなりやすい。一方、素数の和にする場合はつかみどころがない。どちらかと言えば、面白味に欠ける話題だ。と思ってはいけない。実は大変面白いものがあるのだ。

ゴールドバッハ[1)]はオイラー[2)]との手紙のやり取りで、「2 以上の偶数は 2 個の素数の和になる」ことを話題にした。当時の二人は、1 を素数の仲間に入れていた様子なので、$2 = 1 + 1$ も話題に含まれていたようだ。しかし、われわれは 1 を素数の仲間に入れていないので

4 以上の偶数は 2 個の素数の和になる

1) クリスティアン・ゴールドバッハ（1690–1764）：ロシアのドイツ人数学者。
2) レオンハルト・オイラー（1707–1783）：スイスの数学者・物理学者。

と言うべきだろう。現在、この問題に対する反例も証明も見つかっていない。

惜しいことに、この中には唯一という単語は含まれない。つまり、4 以上の偶数はただ一組の 2 個の素数の和になる、わけではない。実際、20 は $20 = 3 + 17 = 7 + 13$ のように二組の素数の和になっている。

ゴールドバッハの予想を検証するには、素数の判定ができなくては困る。そこで素数を判別するスクリプトが必要となる。素数の判定だけなら簡単だろう。こんな関数『

```
\begin{luacode*}
function pmsg(n)
    msg = '素数'
    for i = 2, (n-1) do
        if n % i == 0 then
            msg = '合成数'
        end
    end
    return msg
end
\end{luacode*}
```

』にして、『`\directlua{tex.print(pmsg(41) .. ', ' .. pmsg(42) .. ', ' .. pmsg(43))}`』と書けば『素数, 合成数, 素数』が出力される。

`pmsg` 関数は、引数に与えた数が素数であるかそうでないかを判定するものである。判定というのは、入力してある数に対して、それが素数なら 素数 と表示し、素数でなければ 合成数 と表示するのだ。これだけのことなら、もう少し別のスクリプトが書けるのだが、後のためにもちょっと変わったことをしている。さらに付け加えるなら、無駄も多い。

無駄というのはスクリプトの行数ではなく、処理の仕方である。プログラミングというのは、処理の無駄をなくすために余分なコードを必要とするものだ。`pmsg` 関数はその逆である。余分なコードがないために、無駄な処理をしてしまう。まあ、順を追って見ていこう。

始めに変数 `msg` —message の略—に `'素数'` を代入した。この際に注意することは、代入する文字列を `''` または `""` で囲まなくてはならないことだ。どちらを使ってもよいが、「It's a prime.」と表示したければ `""` を用いて `"It's a prime."` と書かなくてはならないし、「Say "a prime!"」と表示したければ `''` を用いて `'Say "a prime!"'` と書かなくてはならない。要するに `'` と `"` の組がはっきりするように書けばよいのだ。

いま、われわれがしたいことは、変数 msg に素数の判定をした文章を入れておいて、判定結果によって msg の内容を表示したいのである。

候補となる数が素数かどうかの判定は for 構文が受け持つ。for 文が 2 から n-1 までの繰り返しであることに注意してほしい。素数とは、1 と自分自身でしか割れない数である。どんな数でも 1 と自分自身で割り切れるのだから、1 や n 自身で割ってはいけないのだ。もし n が i で割り切れると、msg に 合成数 が代入される仕組みだ。

では、n が i で割り切れないとどうなるのだろう。これまでは else が続いたはずだが、ここには見あたらない。その理由は、割り切れないときは何もする事がないからだ。なぜって、候補である数は最初に素数であると仮定されているからである。したがって割り切れないときは、スクリプトは涼しい顔で次の i を試すことになる。if 構文は for ブロックの中にあるからだ。for 構文をすべて回して、滞りなく割り算を試したら結果を表示すればよい。

ところでこのスクリプトはあまり褒められたものではない。特に割り算テストをする際、2, 3, 4, ..., $n-1$ で割っているが、これらすべてで割るのは時間の無駄なのである。次に素数を判定するスクリプトを書く際は、今度はもう少し効率を考えて書いてみよう。