4 . . . の豆旅

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。

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 以上の偶数は二つの素数の和になる』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 以上の偶数は二つの素数の和になる

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。

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プログラムが必要となる。

---

1) C. 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。

programming list [PrimeChk.java]

```
 1: import java.util.Scanner;
 2:
 3: public class PrimeChk {
 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:         String msg = "a PrimeNumber.";
12:
13:         for(int i = 2; i < n; i++) {
14:             if(n % i == 0) {
15:                 msg = "NOT a PrimeNumber.";
16:             }
17:         }
18:         System.out.println(n + " is " + msg);
19:     }
20: }
```

このプログラムは、入力し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、入力した数**に対して、それが素数なら "** is a PrimeNumber." と表示し、素数でなければ "** is NOT a PrimeNumber." と表示するのだ。これだけのことなら、もう少し別のプログラムが組めるのだが、のち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、不器用なプログラムでもある。とにかく、順を追って見ていこう。

9:行目まではお決まりのコードだ。いままで同様、メッセージを表示して入力を促している。

11:行目は初めて見る型だろう。`String` 型は文字または文字列を扱うためのキーワードになっている。この場合の変数名は `msg` で、扱い方は一般の変数と同じだ。この際に注意することは、代入する文字列を `""` で囲まなくてはならないことだ。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。

はじめ、変数 `msg` には`"a PrimeNumber"`が代入されている。ここで`"a PrimeNumber"`を代入したということは、候補となる数が素数であると仮定しているのだ 。そして、素数でないことが判明したときに、メッセージがすり替わる。

9:行目で変数 `n` に候補となる数が代入されていた。この数が何であれ、11:行目の仮定によって現

時点では n は素数である。

そして 13:行目から候補となる数が素数かどうかの判定が始まるのだ。for 構文が 2 から n の手前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。

実際の判定は 14:行目でなされる。n が i で割り切れると、15:行目で msg に"NOT a PrimeNumber" 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。本来なら else が続くはずだが、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、11:行目で候補である数は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、プログラムは涼しい顔で次の i を試すことになる。

滞りなく割り算を試したら 18:行目で結果が表示される。

ところで [PrimeChk.java] は、あまり褒められたものではない。とくに 13:行目から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。今度はもう少し効率を考えてプログラムをしてみよう。

TRY! このプログラムは一回判定をしたら終了してしまう。候補となる素数を繰り返し調べられるようにプログラムを書き換えよ。