4.3 双子素数

素数の話題は尽きることがない。話題のひとつに**双子素数**というものがある。2 を除けば素数は奇数であるが、ときどき 17, 19 や 41, 43 のように連続して素数が現れることがある。これらを双子素数と呼ぶのだ。

素数が無数にあることは証明されている。それはユークリッド[1]による証明が有名である。証明の大筋は次のようなものだ。

素数が有限個しかないと**仮定**し、素数を $2, 3, 5, \ldots, P$ とする。このとき $2\cdot3\cdot5\cdots P+1$ なる数を作れば、これは P より大きい新たな素数となるから、素数が P までしかないことに矛盾する。この矛盾は素数が有限個であると仮定したことに起因している。すなわち素数は有限個ではない。

このように、ある仮定から始めて矛盾を引き出し、結果、仮定が間違っていると結論する証明方法は**背理法**と呼ばれる。

ところで、素数がどれぐらいの頻度で現れるかとか、双子素数も無数にあるのかなど、素数には未解決の問題が多い。素数の頻度というのも変な感じもするが、ガウスは素数が現れる頻度について、次のような見解を持っていた。

$$1 \text{ から } N \text{ までには、素数はおよそ } \frac{N}{\log_e N} \text{ 個含まれる} \tag{4.1}$$

例えば $N = 100$ とすると、(4.1) より

$$\frac{100}{\log_e 100} \approx 21.7$$

だが、実際は 25 個の素数がある。ぴったりではないが、まあ近いところをついている。式は N が大きいほど、より正しい頻度を教えてくれる。もし N を与えたとき、本当にぴたり正確な素数の数を求める式が発見されれば、画期的なことであるが、今のところそのような式は発見されていない。またガウスの式とは別の、素数の頻度を計算する式はあるが、(4.1) の美しさにはかなわないのだ。何とも不思議なことである。

双子があれば三つ子も考えたくなるのが人情で、3, 5, 7 のような三つ子素数がどれくらいあるかと考えてしまう。だが結論を言おう。これ以外の三つ子素数はない。連続する 3 つの奇数は三つ子素数にはならないのだ。少し考えれば理由は分かるだろう。

三つ子素数がないと分かったところで、双子素数に集中しよう。と言っても場当たり的に双子素数を探すのでは効率が悪い。徹底した調査で、どのような条件のとき双子素数が現れるのかが分かれば、プログラムも組みやすいものだ。そこで、ここでは次の単純な事実をもとに、双子素数を見つけていこう。

$$2, 3 \text{ を除くすべての素数は } 6m \pm 1 \text{ の形をしている}$$

[1]ユークリッド (330?B.C.–275?B.C.) : ギリシアの数学者。

そう、素数は決して $6m \pm 2$ や $6m \pm 3$ のような形をしていないのだ。誤解しないでもらいたいことがある。私は「素数は $6m \pm 1$ の形をしている」と主張しているのであって、決して「$6m \pm 1$ の形をしている数は素数である」と言っているのではない。**逆**は必ずしも正しくないとは、正にこのことを言う。そして、ここでも少し考えれば、すべての素数が $6m \pm 1$ の形をしている理由も分かると思う。

さて、そういうことなら話は簡単だ。$6m - 1$ が素数であるかどうか調べ、素数であるときに限り $6m + 1$ が素数であるかどうか調べればよい。運良く 2 つとも素数なら、それが双子素数ということだ。

programming list [TwinPnum.vba]

```
 1 : Sub TwinPnum()
 2 : Const SUP As Integer = 100
 3 : Dim m, row As Integer
 4 :
 5 :     m = 1: row = 1
 6 :     While 6 * m < SUP
 7 :         If (pchk(6 * m - 1) And pchk(6 * m + 1)) = True Then
 8 :             row = row + 1
 9 :             Sheet1.Cells(row, 1) = 6 * m - 1
10 :             Sheet1.Cells(row, 2) = 6 * m + 1
11 :         End If
12 :         m = m + 1
13 :     Wend
14 : End Sub
15 :
16 : Function pchk(n)
17 : Dim i As Integer
18 : Dim flag As Boolean
19 :
20 :     i = 3: flag = True
21 :     While i <= Sqr(n)
22 :         If n Mod i = 0 Then
23 :             flag = False
24 :             GoTo OUT
25 :         End If
26 :         i = i + 2
27 :     Wend
28 : OUT:
29 :     pchk = flag
30 : End Function
```

マクロが [PrimeNum.vba] に較べ、とても長くなったように感じるかもしれない。だが、そんなことはない。関数 pchk(n) は [PrimeNum.vba] の 8: 〜 16: 行目の焼き直しに過ぎない。大きな違いは pchk(n) が値を返す関数だから、flag の値を返すことだ。しかし [PrimeNum.vba] とちょっとだけ値の扱いが違っている。これは後で述べよう。

結局、双子素数を見つけるマクロは、Sub プロシージャの中のわずか数行分である。ではその部分を見ておこう。

5: 行目の m は $6m \pm 1$ の m に対応した整数変数であり、初期値は 1 である。また、row は双児素数を順次表示させるためにワークシートの行位置であり、初期値は 1 である。しかし、1 行目に表示されることはない。なぜなら、双児素数は 8: 行目で row の値を増やした後に表示されるので、初期値はそれに合わせたからだ。

マクロは 6: 行目の While 文で、上限に設定した SUP までの整数を調べることにしている。今は 100 までの双子素数を調べたいのだ。そのための条件を 6 * m < SUP にしていることに注意を払ってほしい。m < SUP としてしまうと、600 前後までの双子素数を出力してしまう。だからといって何も困ることはないけどね。

さて、このマクロの実質的な内容は 7: 行目に凝縮されている。まず pchk(6 * m - 1) だが、6 * m - 1 の値—最初は 5 ということだ—を関数 pchk(n) に渡して、素数かどうかの判定を仰いでいる。そして最初の 5 は素数なので、pchk(n) は "真" を返してくる。

また、pchk(6 * m + 1) も同様だ。これは最初 7 を判定するので、やはり "真" を返してくるのである。すると問題は And が何をするのかということだ。A And B は**論理演算子**のひとつで、A と B が共に真のときに限り、A And B は真になる。それ以外は偽と判定される。さらに A And B の特徴は、A が偽になれば B の真偽に関わらず、A And B は偽であるから、そのときは B をいちいち計算しないのだ。このことは計算量を押さえる役に立つ。

いずれにせよ 7: 行目では、$6m \pm 1$ が共に素数になったときだけ、真と判定されるというのが大事な点だ。それによって、9:, 10: 行目で双子素数だけがワークシートに表示されるのである。A 列, B 列にペアで表示する仕様にしてある。また、判定が偽の場合は何もする必要がないので、Else 文は省略してある。

では、ここで関数 pchk(n) の値の扱いに触れておこう。内容的には [PrimeNum.vba] と同じなのだが、Dim flag As Boolean が新鮮だ。flag は $6m \pm 1$ が共に素数になったかどうかを記録するための変数だが、この型 Boolean は少しばかり特殊だ。ふつう変数には任意の数値や文字列が代入されるのだが、Boolean 型の変数は "真 (True)" と "偽 (False)" の 2 値だけを取る。これまでのマクロでは、正の値か 0 かで素数を判定したと思うが、このような二者択一の判定には Boolean のほうが適しているので使ったのである。