7.4 そろそろ家に帰ろう

ああ、数に関する景勝地をだいぶ回ったようだ。世の中には数だけでなく、図形や方程式や統計など様々な景勝地があるよね。いやいや、数の景勝地だって、今回の旅で全部回ったわけじゃない。でも、そろそろ家に帰る頃かもしれない。**VBA** についても何らかの知識を得られたし。それに、このまま旅を続けてしまうと、**VBA** の一般的でない癖が身についてしまう可能性がある。数学の話題も然り。多少、厳密性を犠牲にしているしね。一度、家へ帰ってから旅の道を振り返ろう。その際、手元にはプログラミングの書籍と数学の書物を置いておこう。正しいガイドをしてくれるはずだ。

そうして、今回の旅の色々なことが身にしみる頃、また旅に出ようじゃないか。次の旅は数に関する別の景勝地を訪ねてもいいし、違う分野の景勝地を訪ねるのもいい。コンピュータプログラミングは、数だけしか扱えないってことじゃない。図形を深く知るのにも役立つし、統計のシミュレーションも得意だ。そのときは新たな **VBA** の知識が要求されると思う。楽しみはいくらでもあるのだ。

実は、楽しみは今でも各自が持てる。旅の途中に π の値に出会ったことを覚えているね。あのときは、配列をプロシージャに渡す方法を知らなかったために、苦労をしたはずだ。しかし、今ではその方法を知っているのだ。だから π を小数点以下 1000 桁まで計算させることなんてチョロイものさ—おっと、口がすべってしまった。本当はチョロイものではない。しかし、決してできないことではないのだよ。君たちには十分な知識がある。

でも、その前にちょっと練習をしておきたい。練習とは、ある項の値をプロシージャに引渡して配列で計算し、プロシージャが返す加工した値を加算していくことだ。難しい表現をしているけれど、

$$1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\frac{1}{5!}+\frac{1}{6!}+\cdots \tag{7.1}$$

の値を求める計算がひとつの例だ。もちろんコンピュータには無限の計算はできないので、練習では 210 項までの和を求めてみたい。ただ、もし無限に計算をすることができれば、(7.1) の値は

$$1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\frac{1}{5!}+\frac{1}{6!}+\cdots=\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n \tag{7.2}$$

である。右辺は旅の途中で出会っているね。等式 (7.2) は右辺を展開すると左辺になることを示しているのではない。右辺と左辺の式が等価であることを示しているのだ。何とも不思議な関係だ。右辺の大体の値は旅の途中で知ることができたはずだ。ならば、左辺の値を計算してみよう。

大体の値で良ければ配列を用いることはない。**Excel** の表計算機能でも十分である。実際にやってみてほしい。ほらね。(7.2) の等式が正しいと思えてきたでしょう。だけど、これで納得してしまっては家でくつろいでいる意味はない。配列を用いて (7.1) の正確な値を計算させよう。配列を

100 個用意すれば 400 桁の計算になる。この精度で (7.1) の値を求めてみよう。

ところで、ここではマクロの説明は省略させてもらいたい。プロシージャをいくつも呼んでいるが、今までに見たことがあるプロシージャが並んでいるだろう。そう、以前から見慣れているものをちょっと拝借しただけだ。ただ、Sub プロシージャでは配列を 0 で初期化しているね。これをしないとマクロを立て続けに走せた場合、配列に直前の値が残ってしまうからだ。あれ？　これまで配列を 0 で初期化なんてしてたっけ？　そう。だけどこれまでのマクロでも初期化はするに越したことはなかったのだけれど、うまい具合に直前の値が残らなかっただけである。本当は、初期化はしておくべきなのだ。

programming list [Evalue.vba]

```
 1 : Const eSUP As Integer = 210
 2 : Dim gd(103), ge(103) As Long
 3 :
 4 : Sub Evalue()
 5 : Dim t, n As Integer
 6 :
 7 :     gd(1) = 1: ge(1) = 1
 8 :     For t = 2 To 101
 9 :         gd(t) = 0: ge(t) = 0
10 :     Next t
11 :
12 :     For n = 1 To eSUP
13 :         Call factdiv(n)
14 :         Call eadd
15 :     Next n
16 :
17 :     Call edisp
18 : End Sub
19 :
20 : Sub factdiv(nn)
21 : Dim i As Integer
22 :
23 :     For i = 1 To 101
24 :         gd(i + 1) = gd(i + 1) + (gd(i) Mod nn) * 10000
25 :         gd(i) = gd(i) \ nn
26 :     Next i
27 : End Sub
28 :
29 : Sub eadd()
30 : Dim i As Integer
31 :
32 :     For i = 101 To 1 Step -1
33 :         ge(i - 1) = ge(i - 1) + (ge(i) + gd(i)) \ 10000
34 :         ge(i) = (ge(i) + gd(i)) Mod 10000
35 :     Next i
36 : End Sub
```

```
37 :
38 : Sub edisp()
39 : Dim i, j As Integer
40 : Range("A2:J12").NumberFormat = "0000"
41 :
42 :     Sheet1.Cells(2, 1) = ge(1)
43 :     For i = 0 To 9
44 :         For j = 1 To 10
45 :             Sheet1.Cells(i + 3, j) = ge(10 * i + j + 1)
46 :         Next j
47 :     Next i
48 : End Sub
```

今、求めた値は**自然対数の底**と呼ばれ、微積分等で重要な役割を与えられる数だ。円周率の真値を π で表すように、自然対数の底の真値は e で表している。近似値は 2.718 である。

ここで言うのも何だが、[Evalue.vba] で用いたプロシージャ `factdiv(n)`, `eadd()` はあまり褒められたものではない。それは、これらのプロシージャが (7.1) の値を求めるための、専属のプロシージャになっている点だ。プロシージャは汎用であるほうが望ましい。e や π の計算には、桁数が長大な数の加減乗除が必要となる。だから、今回の e の計算のような専属プロシージャにしないで、長大な数用の加算プロシージャ、減算プロシージャ、乗算プロシージャ、除算プロシージャを作っておいて、それを流用する形で利用するほうが便利だろう。

まあ、今の段階では e の値を精確に求められたことを良しとしよう—と言っても、最後のほうの 1, 2 桁は誤差を含んでいるんだけどね。確かな 400 桁の精度が欲しいとき、普通は 400 桁分の配列では足りない。401 桁分の配列が必要だ。400 桁目が正しい値になるよう、計算し直してみたらよいだろう。ちなみに、最後の 5 桁の正しい値は 01416 である。

さあ、ここまで無事に済んだら π の値に挑戦できるってもんだ。π の値を、小数点以下 400 桁の精度で計算してみようじゃないか。その結果が正しいかどうかは、次の 400 桁分の π の値を参照してほしい。

3. 14159 26535 89793 23846 26433 83279 50288 41971 69399 37510
   58209 74944 59230 78164 06286 20899 86280 34825 34211 70679
   82148 08651 32823 06647 09384 46095 50582 23172 53594 08128
   48111 74502 84102 70193 85211 05559 64462 29489 54930 38196
   44288 10975 66593 34461 28475 64823 37867 83165 27120 19091
   45648 56692 34603 48610 45432 66482 13393 60726 02491 41273
   72458 70066 06315 58817 48815 20920 96282 92540 91715 36436
   78925 90360 01133 05305 48820 46652 13841 46951 94151 16094

うわー、家に帰ってからのプログラムの方が、旅で見てきたプログラムより何倍も大変だあ。でも、家にはいつまでもいられるじゃないか。一人旅でものんびりと巡ったように、家でものんびりと構えてみよう。きっと解決できるから。

ただし、安易な解決法方に頼り過ぎないことが大事だ。グローバル変数を使ってマクロを組んだ場合は、プロシージャ内部で変数を宣言して関数へ値を渡す仕様に変更できる。それは、**VBA**の値の渡し方が規定値では**参照渡し**になっているので、実質的にグローバル変数と同等だからだ。しかし、π の値を計算させようとすると、**値渡し**にしたほうがよい所があるはずだ。この違いは、是非自分で調べてもらいたい。今では、自分で調べて解決する十分な力があると思うからだ。後は手元の参考書を調べてくれたまえ。今回の旅は、まだ終わったわけではないのだから。