7.4 円周率 10,000 桁

ああ、長く空中を漂うと景勝地をたくさん見ることができるけれど少し疲れただろう。でも、そろそろ家に帰る頃だ。あとは観光地に自分の足跡を残すだけだ。この観光地に円周率の 10,000 桁でも刻んでおこう（よい子は観光地でそんな真似はしないように)。

円周率を長桁数で行うために必要な加・減・除の計算ルーティン『

```
\begin{luacode*}
function addL(aL, bL)
    for i = 1, #aL do
        aL[i-1] = aL[i-1] + (aL[i] + bL[i]) // 100000000
        aL[i] = (aL[i] + bL[i]) % 100000000
    end
end

function subL(aL, bL)
    for i = #aL, 1, -1 do
        aL[i-1] = (aL[i-1] - 1) + (100000000 + aL[i] - bL[i]) // 100000000
        aL[i] = (100000000 + aL[i] - bL[i]) % 100000000
    end
end

function divL(aL, b)
    l = #aL  aL[l+1] = 0
    for i = 1, l do
        aL[i+1] = aL[i+1] + (aL[i] % b) * 100000000
        aL[i] = aL[i] // b
    end
end

function tmpD(aL, b)
    tL = aL  l = #tL  tL[l+1] = 0
    for i = 1, l do
        tL[i+1] = tL[i+1] + (tL[i] % b) * 100000000
        tL[i] = tL[i] // b
    end
    return tL
end
\end{luacode*}
```

』を載せておこう。前節の text.print 文を削っただけである。ただし、除算用に一つ tmpD 関数を追加した。見て分かるように、return 文を除いて divL 関数とまったく同じである。同じ機能で

別関数を必要としたのは、`return` 文の有無ではない。切実な理由からである。

Lua の場合、関数の引数は**参照渡し**で渡される。参照渡しとは、関数に渡した変数を受け取った関数で加工するとき、渡す側と受ける側の変数が連動していることを意味する。それが何に関係するかは、再びマチンの式

$$\frac{\pi}{4} = 4\left(\frac{1}{1\cdot 5} - \frac{1}{3\cdot 5^3} + \frac{1}{5\cdot 5^5} - \cdots\right) - \left(\frac{1}{1\cdot 239} - \frac{1}{3\cdot 239^3} + \frac{1}{5\cdot 239^5} - \cdots\right)$$

を見ながら説明しよう。

以前は第 n 項の $\frac{1}{2n-1}\cdot\frac{1}{5^{2n-1}}$ と $\frac{1}{2n-1}\cdot\frac{1}{239^{2n-1}}$ を一緒にして計算したものだが、長桁数の計算でそれをやるとひどく効率が悪い。なぜなら、$\frac{1}{5^{2n-1}}$ よりはるかに早く $\frac{1}{239^{2n-1}}$ が小さくなるからだ。つまり、同時に計算を進めると途中から $+\frac{1}{239^{2n-1}}$ は 0 を加え続けるだけになる。

そこで $\frac{1}{5^{2n-1}}$ の項と $\frac{1}{239^{2n-1}}$ の項は別々に計算したいのだが、各項に $\frac{1}{2n-1}$ が掛けられているのが面倒なのだ。これが付いてなければ、第 n 項を計算したらそれを一旦保持し、次に保持したものを $\frac{1}{5\times 5}$ なり $\frac{1}{239\times 239}$ なりで割れば効率的である。Lua は引数を参照渡しで受け取るので、自然とそのようになる。

しかし、$\frac{1}{2n-1}$ の方は前の項とは関係ない値だ。この値を続けて $\frac{1}{5^{2n-1}}$ の項や $\frac{1}{239^{2n-1}}$ の項に掛けてしまっては、余計なものを掛けてから次の計算に回されてしまう。つまり、こちらは連動してもらっては困る。そこで、連動してほしい割り算と連動してほしくない割り算の 2 種類を用意した次第である。

では、$\frac{1}{5^{2n-1}}$ の項と $\frac{1}{239^{2n-1}}$ の項はそれぞれ何項まで計算すればよいだろう。たとえば 1,000 桁ならば、大雑把に $\frac{1}{5^{2n-1}}$ や $\frac{1}{239^{2n-1}}$ が $\frac{1}{10^{1000}}$ を下回るまでである。逆に言えば、$\frac{1}{5^{2n-1}}$ においては $5^{2n-1} > 10^{1000}$ を満たす n を知りたい。一般に f 桁として解いてみよう。対数をとれば簡単に求められる。

$$\begin{aligned}
\log_{10} 5^{2n-1} &> \log_{10} 10^f \\
(2n-1)\log_{10} 5 &> f \\
n &> \left(\frac{f}{\log_{10} 5} + 1\right)/2
\end{aligned}$$

これは $\frac{1}{5^{2n-1}}$ の項なので、$\frac{1}{239^{2n-1}}$ の項なら $n > (f/\log_{10} 239 + 1)/2$ である。これで、求めたい桁数から計算すべき項数が分かった。このことから円周率の長桁数計算をする主関数は『

```
\begin{luacode*}
function mpiL(f)
    TBL = f // 8
    pL, uL, vL, tL = {}, {}, {}, {}
    for i = 0, TBL do
        pL[i], uL[i], vL[i] = 0, 0, 0
    end
    uSUP = (f / math.log(  5, 10) + 1) / 2 + 1
    vSUP = (f / math.log(239, 10) + 1) / 2 + 1

    uL[1] = 16*5 * 100000000
    for n = 1, uSUP do
        divL(uL, 5*5)
        for i = 0, TBL do tL[i] = uL[i] end
        if n % 2 == 0 then
            subL(pL, tmpD(tL, 2*n-1))
        else
            addL(pL, tmpD(tL, 2*n-1))
       end
    end

    vL[1] = 4*239 * 100000000
    for n = 1, vSUP do
        divL(vL, 239*239)
        for i = 0, TBL do tL[i] = vL[i] end
        if n % 2 == 0 then
            addL(pL, tmpD(tL, 2*n-1))
        else
            subL(pL, tmpD(tL, 2*n-1))
        end
    end

    for i = 0, TBL do
        tex.print(string.format('%08d', pL[i]))
    end
end
\end{luacode*}
```

』のようにしてみた。

ちょっとかっこ悪い仕様だが、なんとか正しい値はでる。普通、連動してほしくない引数は**値渡**しをするとよいのだが、変なことをしてしまった。

さて、これで円周率を 10,000 桁計算させることはできるのだが、LuaTEX での組版は相応の時間がかかる。とりあえず 1,000 桁の場合を処理してみるので、どこまで計算させられるかは各自で挑

戦してほしい。実際『`\directlua{mpiL(1000)}`』と書いて処理すると『

00000003 14159265 35897932 38462643 38327950 28841971 69399375 10582097 49445923 07816406
28620899 86280348 25342117 06798214 80865132 82306647 09384460 95505822 31725359 40812848
11174502 84102701 93852110 55596446 22948954 93038196 44288109 75665933 44612847 56482337
86783165 27120190 91456485 66923460 34861045 43266482 13393607 26024914 12737245 87006606
31558817 48815209 20962829 25409171 53643678 92590360 01133053 05488204 66521384 14695194
15116094 33057270 36575959 19530921 86117381 93261179 31051185 48074462 37996274 95673518
85752724 89122793 81830119 49129833 67336244 06566430 86021394 94639522 47371907 02179860
94370277 05392171 76293176 75238467 48184676 69405132 00056812 71452635 60827785 77134275
77896091 73637178 72146844 09012249 53430146 54958537 10507922 79689258 92354201 99561121
29021960 86403441 81598136 29774771 30996051 87072113 49999998 37297804 99510597 31732816
09631859 50244594 55346908 30264252 23082533 44685035 26193118 81710100 03137838 75288658
75332083 81420617 17766914 73035982 53490428 75546873 11595628 63882353 78759375 19577818
57780532 17122680 66130019 27876611 19590921 64201996』が出力される。

ちなみに、出力の最後 1, 2 桁は誤差がある。また、先頭の 00000003 は 3. の意味であるが、見栄えよくする出力は各自で工夫してもらいたい。