3.3 大きい桁数の扱い

前節で定義した関数 mpi(n) に手を加えて何百桁かの π の値を計算することは、そんなに簡単な話ではない。大抵のプログラム言語では、大きな数を扱うには**配列**を使う。Lua には配列ではなく、**テーブル**という概念が用いられている。テーブルは

```
tbl = {}
```

と書くことで、tbl という名前のテーブルが作成される。テーブルは**キー**を文字列にして、たとえば

```
tbl['func'] = 'mpi'   tbl['value'] = 3.14
```

のような指定の仕方をするのが本来の使い方のようだ。この場合、『

```
\begin{luacode*}
    tbl = {}
    tbl['func'] = 'mpi'
    tbl['value'] = 3.14
    for key, var in pairs(tbl) do
        tex.print(var .. '  ')
    end
\end{luacode*}
```

』のように書いて格納されたすべての値を取り出せる。その結果『3.14 mpi』が出力されるが、出力される順は不定である。

しかし、われわれは数字をキーにして数値を管理したいし、出力される順も番号順に

```
tbl[1] = 3   tbl[2] = 14159   tbl[3] = 26535
```

のようになってほしい。そうすれば tbl をつなげて円周率の数値をどこまでも並べることができる。テーブルについては次章でもう少し詳しく述べよう。

問題はテーブルを大きな数に見立てて計算をする方法をどうするかだ。マチンの式には 239^{2n-1} が登場するので、これがすぐに大きな数になることは分かる。割り算の計算には掛け算がかかわってくるし、さらにその値を足したり引いたりする必要もある。要するに π の計算には、大きな数の四則演算がすべて必要なのである。

準備として Lua に $\dfrac{1}{239^m}$ の計算をさせてみよう。テーブルの各変数に 8 桁の数を格納すること

にして、それを 5 個用意すれば 40 桁の数ができあがる。以前確かめたところでは、Lua が精確に扱える整数は 19 桁であったはずなのに、8 桁で格納するの？と思うかもしれないね。しかし、8 桁どうしの掛け算は 16 桁になるので妥当だろう。そのセンでスクリプトを書いてみた。いきなり大仕事になったかもしれないが『

```
\begin{luacode*}
    p = {}  m = 3
    p[1] = 100000000  for i = 2, 6 do p[i] = 0 end
    for i = 1, m do
        for j = 1, 5 do
            p[j+1] = p[j+1] + (p[j] % 239) * 100000000
            p[j] = p[j] // 239
        end
    end

    tex.print('0.')
    for j = 1, 5 do
        tex.print(string.format('%08d', p[j]))
    end
\end{luacode*}
```

』というスクリプトを書いて、このまま処理すれば『0.　00000007 32497753 61251410 88223567 69037378』が出力される。本当に合っているか確かめるには『`\directlua{tex.print(1/239^3)}`』と比較すればよいが、『7.3249775361251e-08』と出力されてしまうので、たぶん合っているだろうと思えるだけである。

スクリプトを順に説明しておこう。まず、`p = {}` でテーブルが作成される。`m = 3` は $\dfrac{1}{239^3}$ の計算が前提なので、`m` の値を変えれば別のべき計算が示される。

テーブルの各変数にそれぞれ値を代入しているが、これは

$$\texttt{p[1]p[2]}\ldots\texttt{p[5]} = 1.00000000\ 00000000\ \ldots 00000000$$

を模していると考えてもらいたい。`p[1]` は 9 桁の数だが想定した 8 桁を超えることで `1.0` を表したと考えてもらいたい。`for` 文の初期化が `p[6]` まで行われたのに気づいただろうか。これは、このあと `j = 5` のときに `p[6]` への代入が発生するために必要なのである。いわゆるダミーだ。用意しておかないとエラーとなってしまう。

入れ子になった外側の `for` 文は `for i = 1, m` から分かるように、239 を `m` 回割るためのループ

である。そして、内側の `for` 文が 239 で 1 回だけ割るためのループである。1 回だけの割り算にループが必要なのは、テーブルの 5 個の変数すべてにわたって割り算の処理が必要だからである。

その方法は実際筆算で割り算をする様子を思い浮かべるとよいかもしれない。筆算でも先頭の `p[1]` から割り始めるだろう。普通の筆算なら適切な桁数を割って余りを 10 倍するところだが、ここでは 8 桁ずつまとめているので、8 桁の数をまとめて割った余りを 100000000 倍して次の桁と合わせるのだ。そして、改めて商を立てることになる。これが `for` 文に書かれた 2 行の処理である。

ひととおり割り算が済んだら、その結果を出力すればよい。1 を 239 で繰り返し割ることを模していたので、先頭に `0.` をつけて見栄えはよくしたい。

いずれにせよ大きな整数の四則計算は、少しだけ手間をかければ正確にできるのである。これで円周率の計算も好きな桁数表示できそうだ。でも、もうちょっと知識を増やしてからにしたいので、再びこの路に戻ったとき出力の仕方を工夫するのと合わせて、長大な桁数の円周率の出力をすることにしよう。