7.2 循環節の計算

循環小数の循環節を無駄なく見ることができるようになった。しかし、不十分だ。われわれが知りたいのは、$142857 \times 2 = 285714$ のように循環節が巡回をしているかどうか、である。cydec 関数は、循環節を眺めることはできても計算をすることはできない。なぜなら、cydec 関数が吐き出した数はまとまった数値でなく、桁ごとの数だからだ。

桁ごとの数をまとまった数値にするには、一つの変数にまとめてやればよい。そこで cydec 関数をちょっといじって『

```
\begin{luacode*}
function cyc(n)
    r = 10 % n  c = 1
    while r ~= 1 do
        r = (r * 10) % n
        c = c + 1
    end
    return c
end

function cyval(n)
    r = 1  val = ''
    for i = 1, cyc(n) do
        val = val .. (r * 10 // n)
        r = (r * 10) % n
    end
    return val
end
\end{luacode*}
```

』としてみた。これで『\directlua{tex.print(cyval(7))}』を処理すると『142857』が、『\directlua{tex.print(cyval(91))}』を処理すると『010989』が出力される。

まず、cyc 関数は前節のものである。もし、この節を前節と一緒に処理するなら今回は cyval 関数の入力だけでよい。

さて、cyval 関数が cydec 関数と違うのは、数字を一つずつ出力する代わりに、val 変数へまとめている点である。そのために val を ''（**ナルストリング**）で初期化している。0 でなく '' で初期化したのには訳がある。0 で初期化すると変数 val は数値扱いとなる。'' なら文字列扱いとな

る。Lua では変数の型は自在にできるので、どっちでも同じようなものだが、ここはぜひとも文字列で扱いたかったのだ。というのは、割り算の最初が 0 になっても 0 を表示したいからだ。数値だとそれは難しい。しかし、'' で初期化した場合、逆に計算には不向きとなる。どちらの手法をとるかは、スクリプト作成上のコストの問題なのである。ちなみに r と val の初期化を 1 行にまとめたのも、行数をケチった以上の意味はない。

val = val .. (r * 10 // n) の書き方に気づいただろうか。.. はこれまで tex.print 文の中で使ってきたが、.. 自体が二項演算子なのだから文字列の連結では常に使える。もし、先頭の 0 を表示しなくてよいなら、val = 0 で初期化した上で、val = val .. (r * 10 // n) と書いた部分を

```
val = 10 * val + (r * 10) // n
```

に直せばよいだろう。これで一気に val が数値になる。

違いはもう一つある。if 文が消えていることに気づいたかな。実は、cyval 関数は別の関数—循環節にいくつかの数を掛け、循環の様子を見る関数—の部品である。そこで、入力されて困る値をはねるための if 文は、本命の関数に移してある。

さあ、いよいよ循環節に数を掛けてみよう。処置は簡単だ。文字列を数値に変換するため tonumber 関数を使い、循環節の回数分の掛け算をすればよい。よって、次のようなスクリプト『

```
\begin{luacode*}
function cycalc(n)
    if (n % 2) and (n % 5) then
        tex.print(cyval(n) .. '\\' .. '\\')
        for i = 2, cyc(n) do
            tex.print(i * tonumber(cyval(n)) .. '\\' .. '\\')
        end
    end
end
\end{luacode*}
```

』を追加しよう。これで『\directlua{cycalc(7)}』と書けば『

142857

285714

428571

571428

714285

857142

』が出力される。

スクリプトはちょっと変わったことをしている。それは、最初に `cyval(n)` を出力して、2 から循環節分の数まで掛けていることだ。1 から循環節分の数まで掛けるコードではまずいのだろうか。

その理由は、ほんのわずかのこだわりからである。$\frac{1}{7}$ の循環節では問題ないけれど、分母が 2 桁になると循環節の先頭は 0 になってしまう。つまり、`n` が 2 桁の場合でも `0` を表示するようしたかったので、最初の値だけは文字列で出力したのである。しかし、これで十分でないことはすぐ分かるだろう。

分母が 3 桁になると、先頭から 2 桁が 0 になるからだ。一般に、分母が n 桁なら、先頭から $(n-1)$ 桁が 0 だ。これらの場合すべてに対応するには、状況によって書式を変えなくてはならない。その場合は、`n` の桁数 f はすぐ分かるので、最初の f 個の数値だけを文字列で表示すればよい。ただし、そのまま表示したのでは `0` がついていないので、あらかじめ $(f-1)$ 個の `0` を連結しておく必要がある。文字の連結にはもちろん `..` 演算子を使う。スクリプトはちょっと面倒になるけど、見栄えに凝るなら、ぜひスクリプトを書き換えよう。