7. . . の蝶道

7.1　不思議な循環小数

$\frac{1}{7} = 0.142857\cdots$ は不思議な性質を持っている。正確には、循環節 142857 の性質が面白いのだ。それは

$$
\begin{aligned}
142857 \times 2 &= 285714 \\
142857 \times 3 &= 428571 \\
142857 \times 4 &= 571428 \\
142857 \times 5 &= 714285 \\
142857 \times 6 &= 857142
\end{aligned}
$$

となるからだ。なんと、循環節の数字が巡回しているではないか。

こうなってくると、他にも不思議な循環節を持つ分数を調べたくなるだろう。以前、循環節を調べたことを覚えているかい？　この道に入り始めた頃は、$\frac{1}{113}$ がどの程度の循環節を持つか調べるのに苦労したね。余りをザーッと出力して、最初の余りと同じものがあるかを調べたんだ。これは手間のかかる作業だし、第一、商がどうなっているのか分からずじまいだった。当時は、循環節を求める力量が不足していたので、仕方なく余りを求めることで目的を達していたのだ。ところが、いまや力量はかなり上昇している。循環小数の循環節を見せてくれるマクロを書いてみよう。

対象とする分数は $\frac{1}{n}$ の形で十分だが、n の値によっては困ることがある。循環しないで割り切れてしまう場合があるからだ。割り切れてしまう分数は、必ず $\frac{1}{2^r 5^s}$ の形をしている。したがって、n には 2 と 5 が含まれていないことが条件だ。ただ、この条件の分数だけを対象とすると、$\frac{1}{6}$ のような循環小数も除外されてしまう。しかし $\frac{1}{6}$ は、$\frac{1}{3}$ の循環小数を $\frac{1}{2}$ で分割していると見れば、本質は $\frac{1}{3}$ と同じだ。よって 2 や 5 がひとつでも含まれている分数は、対象から外して構わないだろう。そのほうがマクロも簡単に書ける。

```
\newcommand\cycdec[1]{%
  \newcount\n \newcount\r \newcount\c
  0.\r=1 \c=1
  \loop \ifnum\c<#1
    \multiply\r10
    \n=\r \divide\n#1 \number\n
    \multiply\n#1 \advance\r-\n
    \ifnum\r=1 \c=#1 \else \advance\c1 \fi
  \repeat
  ...
}
```

これで、『$\frac{1}{7}$ の循環節は`\cycdec{7}`である。』と書けば『$\frac{1}{7}$ の循環節は 0.142857... である。』が出力される。

マクロは、$\frac{1}{7}$ なら “0.142857...” と出力することにした。“...” 以下が循環するという意味である。分子は 1 であるから、間違いなく 1 以下の商になる。したがって、先頭には “0.” を出力するようにしてある。

以前のスクリプトでは、$\frac{1}{n}$ の余りを $(n-1)$ 回出力させた。これだと $\frac{1}{9}$ のような分数では、最初の余りから 8 回目の余りまで 1 が出力されてしまう。今回は商の出力を目的としているので、$\frac{1}{9}$ は何も “0.11111111...” でなく “0.1...” となれば十分だ。それに、余分な循環節を出力するようでは、$\frac{1}{113}$ が本当に 112 の循環節を持つ小数かどうかは、綿密に調べなくてはならず効率が悪い。

しかし、このマクロは不完全である。ひとつ目は大きな数で表面化する。たとえば循環節が 112 あれば、当然ながら 112 桁分の数字が出力される。ところがマクロは、数字を出力することに気が向いているため、改行されることがない。その結果、数字の列が文面をはみ出してしまう。

2 つ目は、分母に 2 または 5 を**因数**に持つ数を引数にしたときに問題が生じる。とくに極端な場合が、分母が 2 または 5 だけの数を引数にしたときだ。この場合は割り切れる。割り切れるということは、余り`\r` が 1 になることはない。よって、たとえば 16 を引数にすると小数点以下 15 までが自動的に出力される。しかも途中で割り切れているため、意味もなく 0 が出力され、さらに “...” まで末尾に付く。もっとも、数値としては何ら間違ってはいないのだけれど。

もう少し見栄えを良くするには、何行かコードを書き加えなくてはならない。せめて文面をはみ出さないようにしておこう。それは簡単なことである。`\number\n` を`\number\n\` にすればよいだけだ。要するに、数字を全部つなげるのではなく、間に空白を入れながら出力するのである。そうすれば適切な空白の位置で改行が起こる。

また大きな数の場合、数字が羅列されると何桁の循環かが分かりづらい。循環節を出力するついでに循環桁数も出力するとよいかもしれない。

```
\newcommand\cycdecB[1]{%
  \newcount\n \newcount\r \newcount\c
  0.\r=1 \c=1
  \loop \ifnum\c<#1
    \multiply\r10
    \n=\r \divide\n#1 \number\n\
    \multiply\n#1 \advance\r-\n
    \ifnum\r=1 ...（循環節\number\c 桁）\c=#1
    \else \advance\c1 \fi
  \repeat
}
```

マクロ\cycdec は、$\dfrac{1}{n}$ の循環節が n 以上になることがないことを利用して、余りが\r=1 になったとき\c に n にあたる引数#1 を代入していた。これにより、\ifnum\c<#1 が偽になるので\loop〜\repeat を抜けるのである。こういうときは\newif を使うのが TEX 流なんだろうけど。

すると循環節を出力するには、\c に#1 が代入される前に値を確保しておかなくてはならない。新しい変数を使ってもよいけれど、TEX 流は、その場で循環節の桁数を出力させてしまうのがスマートだろう。$\dfrac{1}{113}$ の循環節は『\cycdecB{113}』を実行して以下のようになる。

『0.0 0 8 8 4 9 5 5 7 5 2 2 1 2 3 8 9 3 8 0 5 3 0 9 7 3 4 5 1 3 2 7 4 3 3 6 2 8 3 1 8 5 8 4 0 7 0 7 9 6 4 6 0 1 7 6 9 9 1 1 5 0 4 4 2 4 7 7 8 7 6 1 0 6 1 9 4 6 9 0 2 6 5 4 8 6 7 2 5 6 6 3 7 1 6 8 1 4 1 5 9 2 9 2 0 3 5 3 9 8 2 3 ...（循環節 112 桁）』

ところで “この手の” 循環節には面白い性質がある。$\dfrac{1}{113}$ の循環節は 112 桁だから、56 桁ずつ真っ二つにできる。実際にやってみよう。56 桁目の次に改行を入れればすぐだ。こんなことは何度も試すまでもないので、$\dfrac{1}{113}$ 限定のマクロになっているが十分だろう。

```
\newcommand\justhalf{%
  \newcount\n \newcount\r \newcount\c
  \r=1 \c=1 \par
  \loop \ifnum\c<113
    \multiply\r10
    \n=\r \divide\n113 \number\n
    \ifnum\c=56\par\fi
    \multiply\n113 \advance\r-\n
    \ifnum\r=1 \c=113
    \else \advance\c1 \fi
  \repeat
  \par\noindent
}
```

これで、『$\dfrac{1}{113}$ を真っ二つにして並べると\justhalf となる。』と書けば『$\dfrac{1}{113}$ を真っ二つにして並べると

0088495575221238938053097345132743362831858407079646017 6

9911504424778761061946902654867256637168141592920353982 3

となる。』が出力される。それがどうしたかって？　上下に並んだ数を眺めてほしい。ジーっとね。ジャーン！　上下の数を足せばきっちり 999···9 になるんじゃないかな？

どうしてこんなことが起こるのだろうか。もちろんどんな分数にも起こるわけではない。さっき言った “この手の” 循環節をもつ分数に限ってのことだ。じゃあ、それはどんな種類の分数なんだろう。謎は深まるが、それは真っ当な道を歩いて探してもらいたい。