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。\cycdec 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、\cycdec 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、ひとつの変数にまとめてやればよい。

```
\newcommand\cycval[1]{%
  \newcount\n \newcount\r \newcount\c \newcount\val
  \r=1 \c=1
  \loop \ifnum\c<#1
    \multiply\r10
    \n=\r \divide\n#1 \multiply\val10 \advance\val\n
    \multiply\n#1 \advance\r-\n
    \ifnum\r=1 \c=#1 \else \advance\c1 \fi
  \repeat
  %
  \n=\val \val=0 \c=1
  \loop \ifnum\c<#1
    \advance\val\n \number\val\par
    \advance\c1
  \repeat
}
```

これで、\cycval{7}を処理すると、

142857

285714

428571

571428

714285

857142

が出力される。循環節が循環している様子がよく分かるだろう。

マクロ\cycval は以前の\cycdec とほとんど変わらない。違いは、数字をひと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。それで\val には整数値が入る。

さあ、循環節に数を掛ける番だ。$\times 2$, $\times 3$, $\times 4$, ... としたいところだが、掛け算は繰り返しの足し算であるから、\val の値を繰り返し加えればよい。これで循環節の掛け算がどうなるか一覧できるようになる。

しかし、これで喜んでいられない。なぜなら、循環節を計算できるようになったのは前進でも、循環節が長ければマクロは簡単にオーバーフローする。それを避けるには、さらなる工夫が必要だ。

TEX に限らず多くのプログラミング言語でも、桁数が多い数を扱うのは難しい。その場合は**配列**の仕組みを用いて対処している。もちろん、それなりのプログラムを書かなくてはならないけれど。TEX に配列はない。だったら、どうしよう。う〜む、作ってしまおう。

```
\newcount\ARY \newcount\upf
\newcount\I   \newcount\II  \newcount\III  \newcount\IV \newcount\V
\newcount\VI  \newcount\VII \newcount\VIII \newcount\IX \newcount\X
\newcount\TOP
\newcount\i   \newcount\ii  \newcount\iii  \newcount\iv \newcount\v
\newcount\vi  \newcount\vii \newcount\viii \newcount\ix \newcount\x
%
\newcommand\arrayadvance{\ARY=1
  \loop \ifnum\ARY<11
  \advance\ifcase\ARY%case0 is null
    \or\I\i   \or\II\ii   \or\III\iii   \or\IV\iv \or\V\v
    \or\VI\vi \or\VII\vii \or\VIII\viii \or\IX\ix \or\X\x\fi
  \upf=\ifcase\ARY%case0 is null
    \or\I  \or\II  \or\III  \or\IV \or\V
    \or\VI \or\VII \or\VIII \or\IX \or\X\fi
  \divide\upf10000
  \advance\ifcase\ARY%case0 is null
    \or\II  \or\III  \or\IV \or\V \or\VI
    \or\VII \or\VIII \or\IX \or\X \or\TOP\fi\upf
  \multiply\upf10000
  \advance\ifcase\ARY%case0 is null
    \or\I  \or\II  \or\III  \or\IV \or\V
    \or\VI \or\VII \or\VIII \or\IX \or\X\fi-\upf
  \advance\ARY1 \repeat
}
\newcommand\fchk[1]{%
  \ifnum#1<10 000\else\ifnum#1<100 00\else\ifnum#1<1000 0\fi\fi\fi}
\newcommand\arraydisplay{%
  \ifnum\TOP>0 \number\TOP\fi
  \ARY=10
  \loop \ifnum\ARY>0
  \fchk{\ifcase\ARY%case0 is null
    \or\I  \or\II  \or\III  \or\IV \or\V
    \or\VI \or\VII \or\VIII \or\IX \or\X\fi}%
  \number\ifcase\ARY%case0 is null
    \or\I  \or\II  \or\III  \or\IV \or\V
    \or\VI \or\VII \or\VIII \or\IX \or\X\fi\
  \advance\ARY-1 \repeat
}
```

まず、以上が配列を用いて 40 桁の数を足し算するルーティンである。配列は`\I`〜`\X`、および`\i`〜`\x` の 10 個を用意して、余分に`\TOP` を作ってある。これは、“`\X\IX...\I`” と並べて作る 40 桁

の数 A と、“\x\ix...\i” と並べて作る 40 桁の数 B の足し算を想定している。\TOP は繰り上がりが生じた場合の受け皿になり、そのときは 41 桁の数になっている。\arrayadvance が実際に 40 桁分の計算をする部分で、\arraydisplay は 40 桁分の出力をする部分である。TEX は有効桁数を 9 桁強持っているのだが、配列に与える数は 4 桁までを想定している。それに合わせて出力の際にも 4 桁ずつ区切ることになった。

4 桁というのは少し中途半端に思うだろう。TEX ではひとつの変数に 9 桁強の整数が格納できるので、配列には最大 9 桁与えて 90 桁の計算をすることもできるし、さらに\XI 以降の配列を増やせば 100 桁以上の計算も可能である。それでも各配列に 4 桁しか与えなかったのには理由がある。足し算では気にする必要はないのだが、割り算ではマクロの都合上、上の桁に想定した配列からの“桁下がり” があるからだ。当然、最大で 4 桁が下がってくるので、次の桁は最大 8 桁になるからこれが限界なのだ。もちろん、配列に 5 桁を与えて桁下がりで 10 桁になっても対処する方法はあるだろう。でも、さすがに面倒である。

さて、$\frac{1}{17}$ は 16 桁の循環節を持つ。そこで、40 桁にはだいぶ余裕ではあるが、$\frac{1}{17}$ の循環節がどのように循環するかを見よう。

```
\newcommand\cyclic[1]{%
  \newcount\n \newcount\r \newcount\c \newcount\val \newcount\j
  \r=1 \c=1
  \loop \ifnum\c<#1
    \multiply\r10
    \n=\r \divide\n#1 \multiply\val10 \advance\val\n
    \advance\j1
    \ifnum\j=4 {\divide\c4
      \global\ifcase\c%case0 is null
        \or\x \or\ix \or\viii \or\vii \or\vi
        \or\v \or\iv \or\iii  \or\ii  \or\i\fi=\val}%
    \j=0 \val=0 \fi
    \multiply\n#1 \advance\r-\n
    \ifnum\r=1 \c=#1 \else \advance\c1 \fi
  \repeat
}
```

これで、『\cyclic{17}\arrayadvance\arraydisplay』と書けば『0588 2352 9411 7647 0000 0000 0000 0000 0000 0000 』が出力されるので、$\frac{1}{17}$ の循環節が分かる。後ろの 0 が目障りだが、16 桁目までが $\frac{1}{17}$ の循環節になっている。さあ、これを加算してみよう。

おっと、あわてない。いま\cyclic{17}を処理したので\I〜\X には $\frac{1}{17}$ の値が格納されているね。このまま加算すると 1 回余分に加算されてしまう。初期化のルーティンを書いておこう。\cntr を減らすようにして配列の初期化をすれば\cntr の値が 0 になって初期化が終わる。そこで 17 回

目まで加算するなら\cntr<17 まで数えればいいね。さあ、加算の様子を見よう。

```
\makeatletter
\newcount\cntr
\cntr=10
\@whilenum\cntr>0 \do{%
  \ifcase\cntr%case0 is null
    \or\I  \or\II  \or\III  \or\IV \or\V
    \or\VI \or\VII \or\VIII \or\IX \or\X\fi=0%
\advance\cntr-1}\par
%
\cyclic{17}
\@whilenum\cntr<18 \do{%
  \arrayadvance\arraydisplay\par
\advance\cntr1}
\makeatother
```

0588 2352 9411 7647 0000 0000 0000 0000 0000 0000

1176 4705 8823 5294 0000 0000 0000 0000 0000 0000

1764 7058 8235 2941 0000 0000 0000 0000 0000 0000

2352 9411 7647 0588 0000 0000 0000 0000 0000 0000

2941 1764 7058 8235 0000 0000 0000 0000 0000 0000

3529 4117 6470 5882 0000 0000 0000 0000 0000 0000

4117 6470 5882 3529 0000 0000 0000 0000 0000 0000

4705 8823 5294 1176 0000 0000 0000 0000 0000 0000

5294 1176 4705 8823 0000 0000 0000 0000 0000 0000

5882 3529 4117 6470 0000 0000 0000 0000 0000 0000

6470 5882 3529 4117 0000 0000 0000 0000 0000 0000

7058 8235 2941 1764 0000 0000 0000 0000 0000 0000

7647 0588 2352 9411 0000 0000 0000 0000 0000 0000

8235 2941 1764 7058 0000 0000 0000 0000 0000 0000

8823 5294 1176 4705 0000 0000 0000 0000 0000 0000

9411 7647 0588 2352 0000 0000 0000 0000 0000 0000

9999 9999 9999 9999 0000 0000 0000 0000 0000 0000

おお、なんと循環節が循環している上に、最後は $999\cdots9$ になっているじゃないか。このことは、$\dfrac{1}{17}\times 17=(0.0588235294117647\cdots)\times 17=0.999\cdots$、すなわち 1 であることを意味しているのである。