0.4 循環小数の秘密

もうしばらく小数の話題を続けてみよう。

小数には 0.333... のような**無限小数**と、0.125 のような**有限小数**がある。また、ひと口に無限小数といっても、0.333... は**循環小数**と呼ばれる数で、円周率 3.141592... のように**循環しない小数**とは区別している。実は、すべての有理数は有限小数か循環小数になる。言い換えれば、分数は必ず有限小数か循環小数にできるということで、決して循環しない無限小数にはならない。その逆に、循環しない無限小数は決して分数にすることはできない。どういうことだろうか。

たとえば $\frac{1}{6}$ は循環する無限小数である。実際に割り算を行ってみれば一目瞭然だろう。

$$
\begin{array}{rr}
 & 0.\ 1\ 6\ 6 \\
\cline{2-2}
6\ ) & 1.\ 0\ 0\ 0 \\
 & \ \ 6 \\
\cline{2-2}
 & \ \ 4\ 0 \\
 & \ \ 3\ 6 \\
\cline{2-2}
 & \ \ \ \ 4\ 0
\end{array}
$$

割り算は途中で計算を止めているが、このあとは 6 が続くだけである。その理由は簡単だ。割り算の最後の行に余りである 40 がある。そしてこれと同じ余りが一つ前にも出ているね。そう、小数が循環する理由は、以前の余りと同じものが出るからなのだ。

分数を小数に直すときは、分子を分母で割るはずである。そのときの余りは、割る数である分母より小さい数しかありえない。具体的には、6 で割り算をすれば余りは 0, 1, 2, 3, 4, 5 の 6 種類に限られる。つまり余りの種類は、0 を含めて**高々**分母に使われた数だけしかないのだ。そのせいで余りにあたる数は、いつか必ず同じものになってしまう。一度同じ余りになれば、あとは循環するしかないし、余りが 0 になれば割り切れるということなのだから。

それでは、循環する無限小数は、はじめどんな分数だったか気にならないだろうか？　しかし、循環する無限小数をもとの分数に復元するのは簡単である。$0.1666\cdots$ であれば $x = 0.1666\ldots$ とおいて

$$\begin{array}{rrcl} & 100x & = & 16.666\ldots \\ -) & 10x & = & 1.666\ldots \\ \hline & 90x & = & 15 \end{array}$$

のようにすれば、$x = \dfrac{15}{90}$ であることが分かるのだ。この方法はどんな循環小数にも使える。コツは循環する部分がそろうように、**適当な** 10 の倍数を掛けてやればよい。

ところで循環する無限小数のうち、ひときわ目を引くものがあるだろう。$0.999\ldots$ のことだ。これも同様に $x = 0.999\ldots$ とおいて

$$\begin{array}{rrcl} & 10x & = & 9.999\ldots \\ -) & x & = & 0.999\ldots \\ \hline & 9x & = & 9 \end{array}$$

としてみよう。あれ？　$x = \dfrac{9}{9}$ になったぞ。ということは $0.999\ldots = 1$ なんだろうか。その説明の前にぜひ $0.4999\ldots$ や $0.6999\ldots$ を分数にしてほしい。

雰囲気がつかめただろうか。話をちょっと前に戻すけれど、小数の濃度を調べているときに、すべての小数を $0.\alpha_1\alpha_2\alpha_3\alpha_4\ldots$ の形に表したね。このとき、0.5 のような小数は $0.5000\ldots$ とでもするのかなと考えなかっただろうか？　実を言うと、そこには 0.5 や $0.5000\ldots$ のような、いわゆる有限小数は含まれていなかったのである。そこでは 0.5 は $0.4999\ldots$ の形で登場していたのだ。有限小数はすべて $999\ldots$ を含む、循環する無限小数になっていたのである。有限小数は無限小数で表記しておくと都合がよいのだ。そうしておけば、同じ数を 2 回数えることはなくなるから。

さて、話題は $0.999\ldots$ へ戻る。$0.999\ldots = 1$ である。なぜなら $\dfrac{9}{9}$ を実際に割り算してみると、$0.999\ldots$ であることが確認できるのだから。

$$\begin{array}{rr} & 0.\ 9\ 9\ 9 \\ \hline 9\ ) & 9.\ 0\ 0\ 0 \\ & 8\ 1 \\ \hline & \quad 9\ 0 \\ & \quad 8\ 1 \\ \hline & \quad\ \ 9\ 0 \end{array}$$

これは何だか不思議な計算だ。でも、合っている。このようなことが起こるのは、割り切れる割り算に 2 通りの表記法があるからだ。一つは素直に割り切ってしまう計算である。そうすれば $\frac{9}{9} = 1$ となる。そしてもう一つは、いまの例のように $999\ldots$ と商を立て続けてしまう計算である。こちらが濃度の話に登場した表記なのだ。

それにしても $999\ldots$ には悩まされそうだ。その底流をなすのは無限であることは間違いない。路に迷う前に、循環小数の話へ戻ろう。

循環小数を計算してみると、$\frac{1}{6} = 0.1666666\ldots$ のように**循環節**が短いものと $\frac{1}{7} = 0.1428571\ldots$ のように循環節が長いものがある。$\frac{1}{6}$ は余りが最大で 5 種類出る可能性がある。$\frac{1}{6}$ は割り切れないので、0 は余りの種類に含めていない。このことは循環節が最大で 5 になる可能性があるわけだが、実際の循環節は 1 だ。ところが $\frac{1}{7}$ は循環節が最大で 6 になる可能性を持ち、その通り 6 の循環節を持っている。どんな有理数が、可能な限度を目一杯使うのだろう。そんなときはどうしよう？

いちばんの問題は、循環節の長さの調べ方だ。コンピュータが無限に小数を出力してくれれば楽だが、そうはいかない。そこで発想を変えよう。小数は以前と同じ余りが出たときに循環を繰り返す。すると商を調べるのではなく、余りを調べればよいことに気付くだろう。また、分数は分子が分母より小さいと決めつけてよい。なぜなら $\frac{22}{7}$ のような分数は、必ず $3 + \frac{1}{7}$ のような形にできる。この場合、循環する鍵を握っているのは $\frac{1}{7}$ のような、分子が分母より小さい分数である。したがって、分子が分母より大きい分数を考える必要はない。たとえば $\frac{1}{7}$ の筆算余りは『1 3 2 6 4 5』である。これは『

```
\begin{luacode*}
    r = 1  b = 7
    for i = 1, (b-1) do
        tex.print(r)
        r = (r * 10) % b
    end
\end{luacode*}
```

』によって出力された。コードは `luacode*`環境に書いたが、この場合は `\directlua{}` を使って書くとまずいことがある。理由はスクリプトを説明している途中で言おう。

まず分数は $\frac{a}{b}$ という形だが、いま考えている分数は割られる数が割る数より小さいので、割られる数はすでに余りになっていると考えている。そこで $\frac{r}{b} = \frac{1}{7}$ と見て代入をしている。スクリプトで使う変数は、「$a \div b = q$, 余り r」の表現に合わせた。変数 `r` と `b` への代入は並べて書いたが、

もちろんそれぞれ 1 行に書いてかまわない。Lua では、とくに , などで区切ることなく変数を並べて代入できる。また、繰り返しの回数は最大でも分母の値を下回ることに注意しよう。そのため、割り算を繰り返す回数は b より小さくなるので繰り返しは `(b-1)` までよい。

`for` 文はお馴染だが、余りを求める処理は注目に値する。次の余りは `r = (r * 10) % b` で求めている。`(r * 10)` の計算は余りを 10 倍しているところだ。これはわれわれが割り算をする際、上から 0 を下ろす操作である。続く `% b` は余りを求める演算である。Lua にはいろいろな演算子があるものだが、`%` はその一つだ。`A % B` と書いた場合この演算は、`A` を `B` で割ったときの余りを求める計算なのである。よって一連の `(r * 10) % b` は、余りの 10 倍をもう一度割って次の余りを求める操作になっているわけだ。そして次の余りを `r` に代入してやれば、再び同じ式でさらに次の余りを求められることになる。

さて、ここで `\directlua` 命令を使ってコードを書くとまずい理由を言おう。スクリプトは当然命令どおりに処理されるが、コードは TEX の文書の中に書かれていることを忘れないでほしい。つまり、TEX の作法にも影響されるのだ。`\directlua` 命令を使った場合、TEX は演算子 `%` を見てそれをコメントと解釈する。そのため新たな `r` には `(r * 10)` の `%` 以降が無視されて代入される。それでは正しい結果が得られない。実際、`\directlua{〜}` の中にコードを書いて実行してみればよい。期待外れの出力を目にするだろう。`luacode*`環境は `%` に TEX の規則を適用しない。したがって、このあとの路で割り算の余りに関わる演算に出会ったら、`luacode*`環境にコードを書くことは必須となる。

というわけで、与えられた分数の余りが次々と表示され、分数の循環節が長いか短いかは目にできるようになった。しかし、余りを眺めていてもさほど楽しいわけではない。やはり商を眺めて、その分数の特徴を知りたいだろう。だが、いまは翔び回り始めたばかりだ。われわれには、まだ知るべきことが山ほどある。ここの景色をじっくりと眺めるのは、帰り道で再びこの地に来たときにしよう。