0.2 無限と有理数

さあ整数が出揃って、いよいよ数学の世界の奥へ入っていく準備ができた。数は整数の他にも様々な種類のものがあることは知っているだろう。続けていろいろな種類の数を紹介したいけれど、まだ整数についてさえ十分に語っていないのだ。それは何かって？　それは ... の部分。特に $\ldots, -10,$ と $10, \ldots$ のところは単なる省略ではないことを言いたいのである。

ここに使われた ... はこの先の数を書く意味がないことを示している。なぜなら書き切ることができないから。当たり前だよね。ところが ... には、思いもかけない不思議が詰まっている。

君たちは**偶数**と**奇数**は知っているね。自然数に限れば 2, 4, 6, ... が偶数、1, 3, 5, ... が奇数ということになる。ものの集まりを**集合**と呼ぶことにして、偶数の集合を $\mathcal{E}$、奇数の集合を $\mathcal{O}$、そして自然数の集合を $\mathcal{N}$ で表せば

$$\mathcal{E} + \mathcal{O} = \mathcal{N}$$

であることが想像できる。でも残念ながら違う。想像が事実に合わないことはよくある話。単純な足し算は偶数や自然数のような**無限集合**にはそぐわないのである。そのことをカントール[1)]は次のように説明している。

たとえば「りんごが 5 個ある」とは、各りんごに自然数 1, 2, 3, 4, 5 が**一対一に対応**していると見る。このとき、りんごの数 （かず）は 5 に等しいと言える。同様のことを偶数と自然数で考えてみよう。すると『

$$\begin{array}{lccccccc} \text{偶数} & 2 & 4 & 6 & 8 & 10 & \ldots \\ & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \cdots \\ \text{自然数} & 1 & 2 & 3 & 4 & 5 & \ldots \end{array}$$

』のように、各偶数に自然数 1, 2, 3, 4, 5, ... が一対一に対応していることになる。

この用語はまったくもって適切で、われわれの生活感覚にも合っている。なぜなら、同じ濃度の食塩水を混ぜると同じ濃度を持つ食塩水ができるのだから。そして食塩水の量も増えているので、自然数も偶数だけの量に比べるときっと奇数の分だけ量が増えているんだろう。

ちょっと不思議な感覚だけど、日常からかけ離れたものにはありがちなことではある。たとえと

1) ゲオルク・カントール（1845–1918）：ドイツの数学者。

して、（想像を超える大金持ちの資産）+（想像を超える大金持ちの資産）はやはり（想像を超える大金持ちの資産）であるのと同様、... はわれわれの想像を超えた世界ということなのだ。

ところで Lua は、さっきの偶数と自然数の対応を『

```
\begin{tabular}{lcccccc}
\directlua{
    tex.print(' 偶数 &')
    for i = 1, 5 do tex.print('$' .. i*2 .. '$ &') end
    tex.print(\asluastring{$\dots$ \\})

    for i = 1, 5 do tex.print('&' .. \asluastring{$\updownarrow$}) end
    tex.print('&' .. \asluastring{$\dots$ \\})

    tex.print(' 自然数 &')
    for i = 1, 5 do tex.print('$' .. i .. '$ &') end
    tex.print(\asluastring{$\dots$})
}
end{tabular}
```

』のようにして出力したのだ。これは想像を超えてたかな？

ここでは luacode*環境ではなく、\directlua{} を tabular 環境内に用いた。どうも luacode* 環境は tabular 環境と合わないようだ。表組みには必須の & が邪魔をするらしい。

TEX の表組は、1 行中の各要素は & で区切り、\\ で改行を指示する。それに合わせて偶数や ↕ を配置するスクリプトを書けばよいが、これはパズルのようである。あほがやることだね。よい子はまねしちゃダメだよ。

スクリプトは見やすいように空行を入れているが、TEX にとっては連続する空白である。たぶん組版に影響はない。\directlua 命令が luacode*環境と違うのは tex.print 文の書き方である。TEX の命令はそのまま書くことはできず、\asluastring 命令の中で用いる。つまり

tex.print('$\\dots$') ⇒ tex.print(\asluastring{$\dots$})

と書く。少し手間だが TEX の命令をエスケープしなくてよい。そのため表組の改行命令も \\ と書けるので見やすい。\\ のエスケープってちょっと面倒なんだよね。

で、なんだっけ？ そうそう、偶数と自然数は一対一対応が成り立っているので、それらは同じ数（かず）だけあると考えようという話である。このときに数ということばは適切でないように思えるので、カントールは**濃度**という用語を使っている。すると奇数に対しても、奇数と自然数は同

じ濃度を持つと言えるのだ。

ところで Lua はどの程度の大きさの整数を扱えるのだろう。TEX は公称では**寸法**レジスタが扱える最大値は $2^{31}-1$ になっているが、Lua もそれに影響されるのだろうか。こんなスクリプト『

```
\directlua{
    n1 = 10^(14)-1
    n2 = 10^(14)
    n3 = 10^(308)
    n4 = 10^(309)
    tex.print( n1 .. ', ' ..  n2 .. ', ' ..  n3 .. ', ' ..  n4 .. '; ')
    tex.print(-n1 .. ', ' .. -n2 .. ', ' .. -n3 .. ', ' .. -n4)
}
```

』で確認したところ『99999999999999.0, 1e+14, 1e+308, inf; -99999999999999.0, -1e+14, -1e+308, -inf 』のようになった。本当は `'$' .. n1 .. '$'` のようにして数式モードで表示したいけど、煩わしいから手抜きして無様な出力になってごめん。もっとも、`\directlua{}` 全体を `$$` で囲めば数式モードになるんだけど。

とれる値の範囲は TEX と少し異なるようだ。Lua は数値に決められた**型**はなく、変数には整数、小数、文字列など自在に代入できる。ここでは**べき乗**の計算 `^` は結果が整数値でも小数点をつけて表示される。べき乗は $10^{0.30102} = 1.999953968796$ のような場合にも使われるからだろう。で、これを見るかぎりきっちり整数を表示できるのは ± 99999999999999.0 までのようで、これも小数点つきだ。それより大きいと指数表示となっている。そして指数表示でも $\pm 1\mathrm{e}+308$ を超えるとオーバーフローするようである。ちなみに微小な数は $\pm 10^{-324}$ あたりを下回ると 0.0 となってしまう。

ところが `10^(14)` は指数表示で出力されるが、`10*10*10*10*10*10*10*10*10*10*10*10*10*10*10` として出力すると 100000000000000 のようにきちんと表示される。どうなってんだ？　そこで『

```
$\directlua{tex.print(9223372036854775807..', '..9223372036854775807 + 1)}$
```

』としてみたら『$9223372036854775807, -9223372036854775808$』となった[2)]。直接数字を並べて整数を書けば 9223372036854775807 まではいけるんだ。これ $2^{63}-1$ だね。でもそれに 1 を足して -9223372036854775808 になるってどういうこと？　このことについては、しばらく空中を漂ってから説明しよう。

2）MacBook Pro (2019) を使用しての結果。

とりあえず電卓以上の計算能力はあるようだ。でも、とてつもない大きさの数は扱えないのである。そもそも、無限の大きさや無限に続く計算はコンピュータには向いていない。必ずどこかで計算を打ち切らなくてはならないからだ。では、もし計算を打ち切らなくて済むならどうなるんだろう。たとえば

$$1-1+1-1+1-1+1-1+1-1+1-1+1-1+\cdots$$

のように、無限に続く計算はできるのだろうか。先頭から順に見れば

$$(1-1)+(1-1)+(1-1)+(1-1)+\cdots \quad = \quad 0+0+0+0+\cdots \quad = \quad 0?$$

のように思える。でも、くくり方をずらせば

$$1+(-1+1)+(-1+1)+(-1+1)+(-1+\cdots \quad = \quad 1+0+0+0+\cdots \quad = \quad 1?$$

とも思える。どっちが正しいのか分からないけれど

$$1-1+1-1+1-1+1-1+1-1+1-1+\cdots = x$$

とおけば、これを

$$\begin{aligned} 1-(1-1+1-1+1-1+1-1+1-1+\cdots) &= x \\ 1-x &= x \end{aligned}$$

と見直すことさえ可能だ。だったら $x=\frac{1}{2}$?。

さらには

$$\begin{array}{ccccccccc} 1 & +1 & +1 & (+1 & (+1 & (+1 & (+1 & \ldots \\ \to & \to & \to & -1) & -1) & -1) & -1) & \ldots \end{array}$$

みたいに、–1 の項を右へ 3 個ずらせば合計は 3 になるんじゃない？　じゃあ、ずらし方によっては好きな値にできるってこと？　左にずらせば負の値にもなるよね。無限の足し算には普通の計算法則が成り立たないのか、無限の計算だから答も無数に出るのか ... 。いずれにせよ、従来の考えが成立しないことは確かである。夢見鳥路を彷徨（さまよ）って行き倒れになっても困るので、ちょっと立ち止まってみよう。

無限の数は必ずしも扱いにくいものではない。なぜなら $0.333\ldots = \frac{1}{3}$ だから、小数点以下が無限であっても平気な場合もある。このようにわれわれは、端数を表すために**分数**や**小数**を使ってい

る。身近に分数や小数を使っていると、それらの数は自然発生的に生まれたように感じてしまうものだ。しかし、分数や小数はわれわれ人間が作り出した数なのである。

分数や小数はどちらも数と名乗っているものの、性格には大きな違いがある。いまでは $\frac{1}{8}$ と 0.125 は、どちらも同じ量を表す数の扱いを受けている。けれども $\frac{1}{8}$ は、8 の量に対する 1 の量という比を意味し 1 : 8 と表す。すなわち $\frac{1}{8} = 1 : 8$ だ。一方 0.125 は、1 ほどに大きくない量の 0.1 に、0.1 ほどに大きくない 0.02 を加え、さらに 0.01 ほどに大きくない 0.005 を加え合わせた量である。どっちも同じじゃないかって？　そう、量という点ではね。

結局のところ、分数でも小数でも細かい量—これは精密な量と言ってもかまわない—を表せることに変わりがない。たとえば $\frac{10}{3}$ や 3 という値は、**円周率**をおおまかに表している。もちろん円周率というのは、直径に対する円周の比のことだ。でも、これよりも $\frac{22}{7}$ や 3.14 とすることで、さらに $\frac{355}{113}$ や 3.141592 とすることで、円周率をより精密に表せる。ここで注目してもらいたいのが、分数の作り方である。分数はあくまでも自然数の比で表している。一方、小数は数字を増やすことで対処しているが、これは小数に小数を足していく操作になっている。つまり、分数が自然数を継承して作られる数であるのに対し、小数は自分自身を継承している点に注意してほしいのだ。ここに分数と小数の性格の違いを見ることができるわけだ。

今日では、われわれは $a : b$ の比は $\frac{a}{b}$ という分数で表すのが一般的になっている。このように整数 a, b（ただし $b \neq 0$）を用いて $\frac{a}{b}$ で表すことができるすべての数を**有理数**と呼ぶことにする。有理数の呼称は慣れないと戸惑うかもしれない。$\frac{a}{b}$ で表した数は a rational number（比の数）と呼ぶが、日本語訳が有理数となっているだけのことなのだ。すると、この約束に従えば -5 は $\frac{-5}{1}$、0 は $\frac{0}{1}$ と表すことができる数なので、整数も有理数であると言えるのである。

有理数として表せない数は**無理数**と呼ばれる。円周率は**近似値**として分数で表せるが、真の値は整数比で表せない。この先の路で体験するはずだが、LuaTEX は有理数までなら工夫を凝らして正確な値で処理できる。しかし無理数は正確に表せない。え？　コンピュータは $\frac{355}{113}$ だって 3.141592 みたいな小数で表すんだから、正確な値といっても限度があるんじゃないの？　こう思ってるかもしれないね。でも、そうじゃないんだ。そのことは、ずっと先まで翔び回って知ることになるだろう。