0.4 濃度の話

それでは、もう少し濃度について語ることにしよう。

偶数や奇数は自然数と一対一の対応がつく、という理由から濃度が等しいことを説明した。今度は分数、すなわち有理数の濃度についてだが、結論を先に言えば、有理数の濃度は自然数の濃度に等しい。つまり有理数は自然数と一対一の対応がつくのだ。表を見てもらいたい。

b \ a	1	2	3	4	5	…
1	$\frac{1}{1}$ →	$\frac{2}{1}$	$\frac{3}{1}$ →	$\frac{4}{1}$	$\frac{5}{1}$ →	…
	↙	↗	↙	↗		
2	$\frac{1}{2}$	$(\frac{2}{2})$	$\frac{3}{2}$	$(\frac{4}{2})$	$\frac{5}{2}$	…
	↓ ↗	↙	↗			
3	$\frac{1}{3}$	$\frac{2}{3}$	$(\frac{3}{3})$	$\frac{4}{3}$	$\frac{5}{3}$	…
	↙	↗				
4	$\frac{1}{4}$	$(\frac{2}{4})$	$\frac{3}{4}$	$(\frac{4}{4})$	$\frac{5}{4}$	…
	↓ ↗					
5	$\frac{1}{5}$	$\frac{2}{5}$	$\frac{3}{5}$	$\frac{4}{5}$	$(\frac{5}{5})$	…
⋮	⋮	⋮	⋮	⋮	⋮	⋱

( ) と矢印を無視しておけば、ここにはすべての有理数があることが分かるだろう。もっとも 0 と負の有理数を省いているが、この先の話の本質を変えることはないし、簡単のために正の有理数だけで説明を続けるとしよう。

有理数の濃度が自然数の濃度に等しいことを示すには、表にあるすべての有理数に 1, 2, 3, … と番号が振れることを示せばよいわけである。それには、表の有理数を矢印の順に沿って番号を付けるだけで済む。( ) で囲った数は、すでに番号が振られている有理数と同じ値なので、飛ばしておく。すると有理数が自然数に一対一に対応していることが分かるはずだ。えー？　そうかな？　このような番号の振り方では、対角線の左上に番号が付いても、それと同じ量の番号の付いてない数が右下に残るんじゃないの？　そうだね。たしかにそんな気がするかもしれない。でも、それはわ

れわれが無限をよく理解できないからである。この番号の付け方で、もれなく有理数には番号が付くのだ。いくら右下に無尽蔵の有理数があっても、自然数だって無尽蔵にあるんだから。

すると今度は小数の番である。小数は有理数と違い、自然数によって番号を振ることができない。カントールは次のように説明している。

いま、すべての小数にもれなく番号を振ることができたと仮定しよう。そしてすべての小数を番号順に

$$\begin{aligned} d_1 &= 0.\alpha_1\alpha_2\alpha_3\alpha_4\dots \\ d_2 &= 0.\beta_1\beta_2\beta_3\beta_4\dots \\ d_3 &= 0.\gamma_1\gamma_2\gamma_3\gamma_4\dots \\ d_4 &= 0.\delta_1\delta_2\delta_3\delta_4\dots \\ &\vdots \end{aligned}$$

と、並べたとする。そして、これらの数をもとに、次の操作で新たな小数 $\dot{x}$ を作る。$\dot{x}$ の小数第 1 位の数字は、d_1 の小数第 1 位の数字と "違う" 数字 $\dot{\alpha}_1$ を選ぶ。また、小数第 2 位の数字は、d_2 の小数第 2 位の数字と "違う" 数字 $\dot{\beta}_2$ を選ぶ。このように、$\dot{x}$ の小数第 n 位の数字には、n 番の番号が付いた数の小数第 n 位とは "違う" 数字を選ぶのである。すると新しい数

$$\dot{x} = 0.\dot{\alpha}_1\dot{\beta}_2\dot{\gamma}_3\dot{\delta}_4\dots$$

が出来上がるが、$\dot{x}$ はこれまでに番号を振られた数の中にない数である。

ちょっと変だぞ。はじめの仮定では、すべての小数にもれなく番号が振られたはずなのに、いま出来上がった $\dot{x}$ は番号を振られた数とは明らかに違っている。この矛盾が生じた理由は、最初にすべての小数に番号を付けることができると仮定したからだ。仮定は間違っている。すなわち、小数には自然数と同じ番号を振ることができない。つまり、小数の集合と自然数の集合では濃度が違うのだ。それも小数の方が "濃い" 濃度を持っていることになる。

集合の濃度といっても無数に**要素**を含んでいるので、その量を数値で表すことなどできない。そこでわれわれは、自然数の集合の濃度を $\aleph_0$—$\aleph$ はヘブライ文字で "アレフ" と読む—で表すことにしよう。すると、これまでに分かったことから、偶数の集合と奇数の集合の濃度も、さらには有理数の集合の濃度も $\aleph_0$ である。しかし小数の集合の濃度は $\aleph_0$ ではない。

おおまかに言って、小数で表すことができる数は**実数**と呼ばれている。もちろん有理数は実数である。なぜなら $1/3 = 0.333\ldots$ や $5 = 5.0$ のように、あらゆる有理数は小数表記にできるからだ。さて、実数の集合の濃度は $\aleph_0$ ではないので、今度はそれを $\aleph$ で表すことにしよう。これまでの話では、どうやら

$$\aleph_0 < \aleph$$

であるようだ。

すると、ちょっとした疑問が湧いてくるだろう。$\aleph$ より "濃い" 濃度を持つ集合はどんな集合だろう。また、$\aleph_0$ と $\aleph$ の間の濃度を持つ集合はあるのだろうか。残念なことに、その話題を続ける力量が私にはない。この先の大秘境に興味がある人は、足を踏み入れてみてはいかが？

濃度の話から少し離れてしまうかもしれないが、数学において一対一の対応は重要な概念である。有理数の濃度が自然数の濃度と同じというのは、有理数が、1 ずつ増える自然数の性質を継承している表れでもある。**Java** は 1 ずつ増える処理をどうしているのだろう。簡単な例で見ておこう。

programming list [CountUp.java]

```
1: public class CountUp {
2:
3:     public static void main(String[] args) {
4:         for(int i = 0; i <= 100; i++) {
5:             System.out.print(i + " ");
6:         }
7:         System.out.println();
8:     }
9: }
```

プログラムは 0 から 100 までを画面に表示するだけのものだが、ここには多くのことが含まれている。いきなりの初登場が 4:行目の `for` 文だ。`for` 文は繰り返しをするときによく使われる。繰り返しは、その後の `{}` 内に書かれた処理を、`for()` 内に書かれた条件に従って行われる。つまりこのプログラムは 5:行目の `System.out.print;` 文を繰り返し実行するものである[1)]。

`for(int i = 0; i <= 100; i++)` は、"`i` を `0` から始めて `100` 以下である限り—つまり `i` が `100` になるまで—`i` を `1` ずつ増やしながら `{}` 内の処理を繰り返せ" と読めばよいだろう。`i` の前に `int` を宣言しているのは、ここで初めて変数 `i` が顔を出したからだ。もし、`for` 構文の前に `int i;` と

1) "System.out.print 文" と呼べばよいのだが、豆旅では以後、文は ; 付きで呼ぶことにする。

宣言しておけば for 構文内で宣言する必要はない[2)]。というより、二度宣言すると二重に宣言したと言って怒られるのがオチだ。

また、一般にキーボードから $\leq$ の記号は入力できないので、"何々以下" を表すときは <= を用いる。当然 "何々以上" なら >= を使う。また i++ は、繰り返しの際の i の増え方を指示している。i++ で 1 ずつ増える意味になる。for 構文はこのような書き方が標準なので、丸覚えしてかまわない。

繰り返される System.out.print; 文の末尾が.println(); でないことに注意してほしい。ln がない場合は復帰改行をしないのだ。だから i + " "として 2 個の空白を付け足しているので、整数は 2 個の空白を入れながら表示されることになる。これにより i が 0 から 100 に変化しながら、そのときの i の値を画面に表示するのだ。

7:行目の System.out.println; 文は、ただ単に復帰改行をさせるためだけで、これがなくても一向に差し支えない。ただ、これまでの出力で改行をしていないので、ちょっとした礼儀のつもりで入れている。不作法なプログラムは嫌われるからね。

TRY! 4:行目の for(i = 0; i <= 100; i++) を for(i = 100; i >= 0; i--) に変えて実行してみよ（>= の向きに注意 !)。

TRY! i++ は i が 1 ずつ増える意味だが、i += 2 とすると i が 2 ずつ増える意味になる。このことを利用し、100 以下の偶数を表示するプログラムにせよ。また、100 未満の奇数も表示させてみよ。

2) for のように {} でグルーピングした文の集まりを、豆旅では以後、**構文**と呼ぶことにする。