2. . . の蝶道

2.1　指数関数

数学に**関数**は付きものである。関数は特別難しいものではない。記述の仕方に慣れさえすれば、関数はむしろ便利で使いやすいのである。

関数 $f(x) = x^2$ があるとする。これは、f という名の関数に (何がし)^2 の機能を持たせている。そのため関数 f に 5 を与えると、25 という値が返ってくるのだ。また関数 $g(x) = 2^x$ は、g という名の関数に $2^{(\text{何がし})}$ の機能を持たせている。よって関数 g に 10 を与えると、1024 という値が返ってくるのだ。

ここでちょっと変な感じを持った人がいるだろうか？　私が関数 f, g と言ったことに対してだ。だが、これでよい。$f(x)$ と書くのは、関数 f が変数 x を使うことを明らかにするためなのだ。もし皆が皆、変数には x しか使わないと取り決めていたら、$f = x^2$, $g = 2^x$ で十分である。しかし、変数には x 以外の文字を使うことはよくあるし、変数が 2 つ以上ある関数だってある。さらには文字定数を利用する場合があることを思えば、関数名に続けて ( ) を付け加えることが不可欠なのである。

関数の例をあと 2 つ提示しよう。

最初は $f(x,\ y) = x^2 - 3y$ でどうだろう。関数 f は変数に $x,\ y$ を使う。その結果返ってくるのが $x^2 - 3y$ で計算された値だ。だから $f(5,\ 10)$ と書けば、$x = 5$, $y = 10$ を代入して計算した値 -5 が返ってくることになる。一般に数学で使う関数は、実数値を受け取って実数値を返すので、$f(0.8,\ 0.1)$ でも計算可能で 0.34 が返ってくる。

2 つ目の関数は $g(x) = \lfloor x \rfloor$ でどうだろう。見慣れない記号 $\lfloor\ \rfloor$ はフロア記号と呼ばれる。フロア記号を簡単に説明すると、与えられた値を超えない整数のうち、最大のものを返す機能を持っている。むむ、難しい表現だ。具体的には $g(3.14)$ と書けば 3 が返され、$g(-3.14)$ と書けば -4 が返されるのだ。いわゆる “切り捨て” である。フロア（floor）の名称から想像できるように、これは建物の “床” を意味する。3.14 階という表現はおかしいが、3 階の部屋の中空に 3.14 階があると思えば、その床は 3 階だから $\lfloor 3.14 \rfloor = 3$ とみなすのである。

フロアと対（つい）になる関数はシーリング（ceiling）と呼ばれ、$\lceil x \rceil$ で表される。$\lceil x \rceil$ は、x を下回らない最小の整数である。具体的には $\lceil 3.14 \rceil = 4$、$\lceil -3.14 \rceil = -3$ などである。

いま 2 つの関数を例に出してみたが、値を求めるためにする記述は、いずれも $f(5,\ 10)$ や $g(3.14)$ のように簡便だ。にもかかわらず -5 や 3 の値が返ってくるのは、裏方で f や g がせっせと $5^2 - 3 \times 10$ や $\lfloor 3.14 \rfloor$ の計算をしているからである。また、$f(3,\ 8)$ のように別の値を与えれば、それに応じた値が返ってくる。これは、関数がする仕事がきちんと決められているからである。

何のことはない。関数とは、与えられた値を別の値に加工する手続きなのだ。ということは、これまでに這い回ってきた蝶道でも、関数を目にしてきたはずだ。そう、引数を取る命令は関数である。

この道では**指数関数**を話題に取り上げる。はじめに、もっとも基本的な $f(x) = 2^x$ にしておこう。x の値を入力すると 2^x の値を出力するマクロである。

```
\newcount\n \newcount\p \newcount\c
\newcommand\twopow[1]{%
  \p=#1 \n=1 \c=0
  \loop
    \multiply\n2
    \advance\c1
  \ifnum\c<\p \repeat
}
```

これで、『2^8 は予想通り`\twopow{8}\number\n` である。』と書けば『2^8 は予想通り 256 である。』が出力される。`\twopow` は引数に 8 を与えて 256 を返すので、立派な関数である。変数の定義が関数の外に書かれていることと、いままでなら関数の中に書いていた`\number\n` を文章中で用いていることに注意してもらいたい。この理由は`\twopow` を別のところで使いたいからである。と言っても、作為的な例の中でだけれど。

普通 2^p というのは 2 を p 回繰り返し掛けるので、繰り返しのために`\loop`〜`\repeat` を用いている。同じことは`\@whilenum` を使ってもできるので、`\@whilenum` には、この先で手本を示してもらうことにする。ところで本当はこのマクロに変数`\c` は要らない。`\advance\p-1`、`\ifnum\p>0` とすれば`\p` をカウンタに使えるからだ。`\c` にしゃしゃり出てもらったのは、このあとの説明の都合があるからだ。

さて、掛ける回数は引数から分かるので問題ないが、掛け始めは何だろう。2^p だから 2 が掛け始めと考えてもよいけれど、ここでは土台に 1 を選んでみた。そうすると掛け始めは 0 回目と数えることになる。`\n=2` から始めたら`\c=1` から数えればよい。どちらがよいかは好みの問題と思われる。

で、関数の話だっけ？　関数のよいところは使い回しが利くことだろう。たとえば、2^p の値は急速に大きくなるが、平均してどの程度の増加率なのだろう。それには $\dfrac{2^p}{p}$ を計算すればよい。ところで、2^p を求める関数はたったいま書いたばかりなので、目的の関数は次のようにすればよいはずだ。

```
\makeatletter
\newcommand\powavg[1]{%
  \twopow{#1}%
  \dimendiv(\n, #1)%
  \strip@pt\dimen@
}\makeatother
```

これで、『2^6 の平均増加率は`\powavg{6}`、2^8 の平均増加率は`\powavg{8}`である。』と書けば『2^6 の平均増加率は 10.66678、2^8 の平均増加率は 32 である。』が出力される。2^6 の方の計算がちょっと誤差を含んでいるけれど、理由は少しあとの道で説明しよう。`\powavg` は引数を 1 個取るのだが、これはそのまま`\twopow` の引数にもなっている。`\twopow` が実行された際の`\n` の値は次の割り算でも使用され、`\dimendiv` の割り算の結果が`\dimen@`に保持される仕様であったことから、正しく平均の増加率を計算できている。

`\twopow` はあとで使うことを前提に作ったために、変数の定義や結果の出力は関数の本体に記述しなかったことを思い出してほしい。変数を関数の本体に記述せず、この章の文中で、まるで文章の一部であるかのように記述していたはずだ。そのように書くと、変数は TEX が処理するこの文書全体で定義されたことに等しい。したがって、`\powavg` において変数を定義しなかったとしても、しっかり使えるのである。

ところで`\twopow` には、マクロ中に`\c=0` が記述されていたはずだ。実は、`\twopow` を使うのが 1 回限りなら`\c=0` がなくても正常に動く。しかし、`\twopow` を関数にして繰り返し使うなら話は別だ。試しに`\c=0` を無効にして`\powavg` を実行してみよう。

```
\newcommand\twopowE[1]{%
  \p=#1 \n=1 %\c=0
  \loop
    \multiply\n2
    \advance\c1
  \ifnum\c<\p \repeat
}
\makeatletter
\newcommand\powavgE[1]{%
  \twopowE{#1}%
  \dimendiv(\n, #1)%
  \strip@pt\dimen@
}\makeatother
```

すると、『2^6 の平均増加率は\powavgE{6}、2^8 の平均増加率は\powavgE{8}だろうか。』と書いてみると『2^6 の平均増加率は 0.33339、2^8 の平均増加率は 0.24998 だろうか。』が出力される。エラーになるマクロなので、マクロ名はそれぞれ\twopowE、\powavgE とした。

明らかにおかしな値が出力されている。それは、この文書のどこかで使われた\c の値が初期化されないままになっているためだ。この場合、文書のどこかというのは、これ以前に\powavg{8}を処理したときである。このとき\c の値は 8 になっている。ここで\powavgE{6}が実行されると\n=1 に初期化され、\towpowE は\n を 2 にして\c を 9 にする。このとき\p=6 であるから\c<\p は偽となって、このまま\dimendiv(2, 6) が実行されてしまい 0.33339 が出力されるのだ。2^8 も同じ状況である。ただし、いずれも若干の誤差を含んで表示されているのは\dimendiv のせいである。

このことからも変数を初期化する大切さが分かるというものだ。変数を定義するとき 0 で初期化されることに甘えてはいけない。\c=0 は明示しておく必要がある。蛇足ながら\towpowE に記述されている%\c=0 はマクロに何の影響も与えない。“%” は空白を出力させない機能があるため、改行による空白の出力を押さえることができる。このことは “%” 以降が無効になることを意味し、それはコメントの記述に利用できるということである。したがって\twopowE においては、%\c=0 はコメントアウトされたことになっているのである。

蝶道に入る前に断りを入れた通り、TEX をプログラミング言語の代わりに使うのは普通じゃない。一般的なプログラミング言語なら、こんなことはあまり起こらないだろう。しかし教訓にはなるはずだ。変数の初期化を手抜きしてはいけない。