5.2 漸化式

では、フィボナッチ数列に戻ろう。

よく数列の n 番目の項を a_n などと表記することがある。すると次の $n+1$ 番目の項は a_{n+1}、$n+2$ 番目の項は a_{n+2} という書き方になるだろう。フィボナッチ数列は、直前の 2 項の和が次の項になっているので、一般に

$$a_{n+2} = a_{n+1} + a_n$$

という書き方ができる。特にいまは 1, 1 から数列を始めているので、$a_1 = 1$, $a_2 = 1$ の条件が等式に付け加わることになる。

これだけの条件と式があれば、この先の項が順次計算できるのだ。このような形式で与えられる式を**漸化式**と呼ぶ。数学の世界に限らず、前後の関係は分かっているのだが、そのものズバリを与える式が不明であることは多い。実はフィボナッチ数列もそんなもののひとつだ。数列の前後の関係は漸化式により明確に分かっている。では、フィボナッチ数列のズバリ第 n 項を求める式は何だろう。これはちょっと複雑な計算をするので、詳しいことは省くが、フィボナッチ数列の第 n 項 a_n は

$$a_n = \frac{1}{\sqrt{5}}\left\{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right\}$$

で与えられる。

実際に計算すると大変だが $n = 1, 2, 3, \ldots$ を代入すれば

$$1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ \ldots$$

が現れるのだ。式には $\sqrt{5}$ があるのに、フィボナッチ数列には一切登場しない点が興味深い。近似値でよければ、{ } 内の前の項が $(1.618033\cdots)^n$ で、後ろの項が $(0.618033\cdots)^n$ であることを利用しよう。n が大きいほど $(0.618033\cdots)^n$ は 0 とみなせるので、第 a_n 項は近似的に $\dfrac{1}{\sqrt{5}}(1.618033\cdots)^n$ と考えてよい。電卓を使えば、$n = 9$ のときは $a_n \approx 33.994116$ であることが分かる。確かに第 9 項の 34 に一致している。

TEX には残念なことに、漸化式をうまく扱える仕組みが備わっていない。以前、フィボナッチ数列を生成したマクロを覚えているね。そして君たちはそれをもとに、整数 n を与えて n 番目のフィボナッチ数を出力するマクロを書いただろう。え？　書いてない？　そりゃそうだ。ここで目にするマクロでさえ実行するのがはばかられるのに、その上、別のマクロを書くなんてどうかしてるしね。まあ、いいや。ここでは当時とほぼ同じ動作をするものを、漸化式風の仕組みにして実現してみよう。次のマクロは、整数 n を与えると n 番目のフィボナッチ数を出力する。

```
\def\I{\advance\f1}
\def\II{\advance\f1}
\def\III{\II\I}
\def\IV{\III\II}
\def\V{\IV\III}
\def\VI{\V\IV}
\def\VII{\VI\V}
\def\VIII{\VII\VI}
\def\IX{\VIII\VII}
\def\X{\IX\VIII}
\def\XI{\X\IX}
\def\XII{\XI\X}
\def\XIII{\XII\XI}
\def\XIV{\XIII\XII}
\def\XV{\XIV\XIII}
\def\XVI{\XV\XIV}
\def\XVII{\XVI\XV}
\def\XVIII{\XVII\XVI}
\def\XIX{\XVIII\XVII}
\def\XX{\XIX\XVIII}
\def\XXI{\XX\XIX}
\def\XXII{\XXI\XX}
\def\XXIII{\XXII\XXI}
\def\XXIV{\XXIII\XXII}
\def\XXV{\XXIV\XXIII}
\def\XXVI{\XXV\XXIV}
\def\XXVII{\XXVI\XXV}
\def\XXVIII{\XXVII\XXVI}
\def\XXIX{\XXVIII\XXVII}
\def\XXX{\XXIX\XXVIII}
\newcommand\fib[1]{%
  \newcount\f
  \ifcase#1%case0 is null
    \or\I\or\II\or\III\or\IV\or\V
    \or\VI\or\VII\or\VIII\or\IX\or\X
    \or\XI\or\XII\or\XIII\or\XIV\or\XV
    \or\XVI\or\XVII\or\XVIII\or\XIX\or\XX
    \or\XXI\or\XXII\or\XXIII\or\XXIV\or\XXV
    \or\XXVI\or\XXVII\or\XXVIII\or\XXIX\or\XXX
  \fi{\number\f}%
}
```

アスキーアート（テキスト文字によるイラスト）じゃないからね。これで、『10 番目のフィボナッチ数は`\fib{10}`である。』と書けば『10 番目のフィボナッチ数は 55 である。』が、『30 番目のフィボナッチ数は`\fib{30}`である。』と書けば『30 番目のフィボナッチ数は 832040 である。』が出力される。

分かってるよ、言いたいことは。そもそも、こんな蝶道に迷い込んでること自体、馬鹿げてるん

だからね。でも、これでもフィボナッチ数が計算できるんだ。

仕組みを説明するために、具体的に#1 に 4 が与えられたとして処理を追ってみよう。

まず\ifcase は、判別される 0 以上の整数—この例では引数\#1 の値—によって分岐する命令である。\fi までにたくさんの\or があるけれど、#1 の値 0, 1, 2, 3, . . . , 30 に応じて\or で区切られた処理が選択される。\ifcase#1 の直後には#1=0 のときに行われる処理が書かれるが、いまは何も記述されていないので 0 に対応する処理はない。続けて\or\I 等が列挙されているので、#1 の 1, 2, 3, . . . に応じて\I, \II, \III, . . . が選択される。\ifcase は\or を追加すれば、好きなだけ分岐を増やすことができる。

いま#1 には 4 が与えられたとしたので、4 番目の\or にある\IV が実行される。\IV はあらかじめ\def\IV{\III\II}によって\III\II と定義されているので、\III\II に展開され実行される。しかしながら、\III と\II も同様に\II\I と\advance\f1 に展開される。そして最終的に\IV は\advance\f1\advance\f1\advance\f1 に展開されるのである。これは\f の値を 1 ずつ増やすことを 3 回するので、\f には結局 3 が代入されることになる。したがって{\number\f}によって、4 番目のフィボナッチ数 3 が出力されるのである。

何をしているかというと、いわゆる**再帰呼び出し**風の処理を再現したのだ。再起呼び出しは、前後関係がはっきりしていて、一般の式を求める必要がないときなどに有効である。ただし、再帰呼び出しは計算量が爆発的に増加してしまう。たとえば\fib{6}の計算でも

```
\fib{6}  =  \VI
         =  \V\IV
         =  \IV\III \III\II
         =  \III\II \II\I \II\I \advance\f1
         =  \II\I\advance\f1 ... ... \advance\f1
         =  \advance\f1\advance\f1 ... ... \advance\f1
```

のように、\VI が\V\IV に展開されを、\V と\IV がそれぞれ\IV\III と\III\II に展開され、さらにそれぞれが. . . 。何ということだ。展開が倍々に増えているじゃないか。

倍々に増えると言っても\II と\I は 1 個の命令に展開されるだけなので、単純に\fib{30}の計算が 2^{30} 回行われるわけではない。実際は 160 万回ほどだ。TEX は展開処理を高速に行うようだが、160 万回の展開は “一瞬で” 結果を出力できる処理回数でもない。再帰呼び出しをうかつに利用するのは要注意である。もっとも、TEX で再起呼び出しをする価値があるかを考えた方がよい。

4

ここでは`\ifcase`で30までしか対応できないが、`\fib{30}`でさえ処理に相応の時間が必要なはずだ。だから釘を刺しておこう。`\XXXI`以降を定義し、`\ifcase`にも`\XXXI`以降を追加して、たとえば`\fib{40}`なんて実行しないように。労力の割に得るものは少ないだろうから。