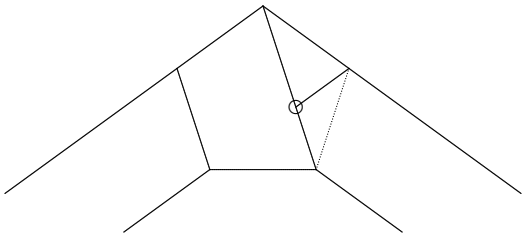
5.5 ここにも黄金比

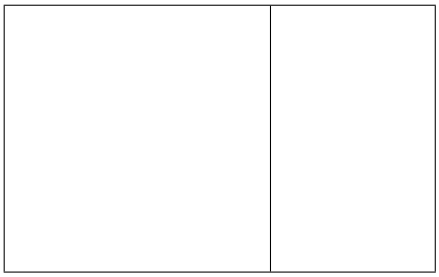
豆旅の途中に少し寄り道をしよう。

正 5 角形は意外と簡単に作れる。それには、適当な長さの紙テープを用意すること。紙テープをくるりと結んで、折り目をきちんとつけると正 5 角形ができる。



そして、ここにも黄金比が隠れている。図中、○印の位置で対角線が黄金分割されているのである。

また長方形には、美しい長方形と美しくない長方形がある。美しい長方形とは、長方形から正方形を切り取ったとき、残りの長方形がもとの長方形と相似になるものを言う。



なぜ美しいかと言えば、残った長方形から同じように正方形を切り取っても、次に残る長方形も再び相似形になるのだ。この操作は寸分たがわず繰り返すことができる。そして、この性質を持つ長方形は、縦・横の比が黄金比なのである。

紙の話題が続いたところで、現在われわれが使っている用紙にも触れておこう。これは、黄金比を持つ長方形より若干横の長さが短い。その比は $1 : \sqrt{2}$ である。しかし、これも均整がとれた長方形だ。なぜなら、この長方形は二つ折りにしたときに、はじめの長方形と相似形になる。すると、いくら半分に折っても形が変わらない利点がある。だから、定型用紙として普及しているわけだ。

二つ折りにすることは、用紙の面積が 1/2 になることである。面積が 1/2 になるということは、

各辺の長さが $1/\sqrt{2}$ になることだ。だから、A3 → A4 の拡大率が $1/\sqrt{2} \approx 0.707$ になっているわけだ。

それでは、用紙の大きさは何が基準になっているか知ってるかい？　基準になっているのは A0 と呼ばれる用紙だ。これを二つ折りにして A1、また二つ折りにして A2、またまた折って A3、... という具合だ。では A0 のサイズは？　実は A0 は、面積が $1\,\mathrm{m}^2$ である。基準の用紙らしい大きさだね。

EX.　A0 の用紙は、縦・横比が $1:\sqrt{2}$ で面積が $1\,\mathrm{m}^2$ である。縦・横の長さはそれぞれ何 cm か。

用紙には A 判の他に B 判がある。B 判の基準は B0 と呼ばれる用紙で、これも基準の面積を持っている。用紙の縦・横比が $1:\sqrt{2}$ だから、基準の面積も $\sqrt{2}\,\mathrm{m}^2$ とすれば整合性があったのに、惜しいかな、$1.5\,\mathrm{m}^2$ の面積となっている。そのせいで、用紙は B0 → A0 → B1 → A1 → B2 → ... の順に小さくなるが、拡大率は 0.816 と 0.866 が交互に出現するのだ。もし B0 の面積が $\sqrt{2}\,\mathrm{m}^2$ であれば、拡大率は常に 0.841 だ。もっとも、B 判の用紙を使わなければ何も面倒なことはないけれどね。

ただ、A 判と B 判の両方を使う機会が多い人は、拡大率の対応表でも作っておくと便利だろう。何も **Java** で表を作ることもないが、配列を利用して一覧表を仕上げてみるもの一興だろう。

programming list [Octavo.java]

```
 1: public class Octavo {
 2:
 3:     public static void main(String[] args) {
 4:         String[][] t = new String[2][6];
 5:
 6:         t[0][0] = "1.000"; t[0][1] = "0.707"; t[0][2] = "0.500";
                    t[0][3] = "1.224"; t[0][4] = "0.866"; t[0][5] = "0.612";
 7:         t[1][0] = "0.816"; t[1][1] = "0.577"; t[1][2] = "0.408";
                    t[1][3] = "1.000"; t[1][4] = "0.707"; t[1][5] = "0.500";
 8:
 9:         for(int i = 0; i <= 1; i++) {
10:             for(int j = 0; j <= 5; j++) {
11:                 System.out.print(t[i][j] + " ");
12:             }
13:             System.out.println();
14:         }
15:     }
16: }
```

このような表は、いわゆる行列の考えに基づいて作られる。[Octavo.java] は

	A_n	A_{n+1}	A_{n+2}	B_n	B_{n+1}	B_{n+2}
A_n	1.000	0.707	0.500	1.224	0.866	0.612
B_n	0.816	0.577	0.408	1.000	0.707	0.500

の表の、数値部分を出力するものである。数値部分は 2 行 6 列の行列だ。

Java で 2 行 6 列の行列を実現するには配列を使うとよい。この場合の配列は `t[0][1]` のように、`[]` を並べて記述することになっている。それに伴ってプログラムでは 4:行目で、`String[][]` という宣言の仕方をしている。配列の値を計算に使うつもりはないので文字列型の配列だ。6:, 7: 行目で各配列に値を代入している。何も 2 行に分けて書く必要はないが、表の出来上がりをイメージするためにも 2 行に分けて書いただけだ。紙面の都合で 4 行になってしまったけれど、コードとしては 2 行である。

表の出力は 9:–14:行目でなされる。ここでは二つの `for` 構文が入れ子になっている。

`for` 構文は外側の `i = 0` から始まるが、先に繰り返しをするのは内側の `for` 構文である。すなわち `i = 0` のまま、`j` の値が `0` から `5` まで変化する。それに伴って 11:行目の `System.out.print;` 文が実行されるので、順に `t[0][0]`, `t[0][1]`, `t[0][2]`, ..., `t[0][5]` が出力される。配列は 6 列だが、列番号は `0` から `5` に対応していることに注意してほしい。

ひと通り `j` に関する `for` 構文が処理されたら、プログラムは次の `i` へ移る。入れ子になった `for` 構文は常にこのパターンで繰り返しが行われるのだ。

ただし、ここでは表に仕立てたいので、一旦 `j` に関する `for` 構文を抜けたら改行しておかなくてはいけない。それが 13:行目である。

TRY! `[Octavo.java]` では項目行などが表示されない。項目行なども表示するよう、プログラムを修正せよ。

おっと、いかん。寄り道が過ぎたようだ。迷子になる前にもとの道へ戻るとしよう。