アルゴリズムとデータ構造
演習第 4 回
再帰

再帰（アルゴリズムＣ 第１巻 p.59） に関する演習です。 再帰のコツは、ループと同様に、まず最初に終了条件を考えることです。

問題 1 ［印刷用 PostScript］

(1) 次のような数列がある。これらの数列の漸化式を書きなさい。

- 階乗 f(n) = n! の漸化式
- 初項 1、等比 2 の等比級数 $f(n) = 2^0 + 2^1 + 2^2 + \ldots + 2^n$ の漸化式
- フィボナッチ数列 1, 1, 2, 3, 5, 8, 13, 21, 34, ... 第 n 項 f(n) の漸化式
- n 次多項式 $f(x,n) = a_0 x^n + a_1 x^{n-1} + \ldots a_{n-1} x + a_n$ のホーナー法を使っての漸化式

(2) 以下のような関数 rule() がある（教科書とは若干異なる）。 関数 mark() は定規に線を引く関数であり、 例えば mark(13, 3) は 13 の位置に高さ 3 の線を引く。 ここで、rule(4, 12, 3); を実行した場合、 定規にはどのように線が引かれていくかを、順番に図示しなさい。

```
void rule(int left, int right, int height){
  int middle = (left+right)/2;
  if( height>0 ){
    mark(middle, height);
    rule(middle, right, height-1);
    rule(left, middle, height-1);
  }
}
```

(1) は漸化式を作る問題です。例えば、初項 1、等差 1 の 等差級数 f(n) = n + (n-1) + ... + 2 + 1 の漸化式は このようになります。

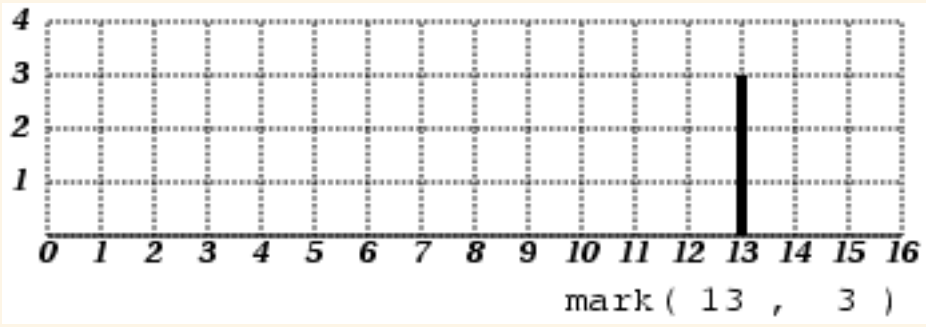
?

ホーナー法は高次多項式を (ax + b) の形に展開していく方法です。 例えば、 $f(x) = 1x^4 + 2x^3 + 3x^2 + 4x + 5$
をこの方法で展開すると次のようになります。

Algorithms and Data Structures

$$
\begin{aligned}
& 1x^4 + 2x^3 + 3x^2 + 4x + 5 \\
= \quad & (1x^3 + 2x^2 + 3x + 4)x + 5 \\
= \quad & ((1x^2 + 2x + 3)x + 4)x + 5 \\
= \quad & (((1x + 2)x + 3)x + 4)x + 5
\end{aligned}
$$

(2) は再帰で実行される順番を考える問題 （アルゴリズムＣ 第１巻 p.62） です。 例えば mark(13,3) を実行すると、次のような線が 引かれることになります。



この mark() 関数の引数が、どういう順番で何になるかを考えて、 図に書いてみてください。

問題２

次の計算を行なう関数を再帰で書きなさい。

- 階乗
- 等比級数
- フィボナッチ数列の第 n 項の値
- 4 次多項式 $f(x) = 1x^4 + 2x^3 + 3x^2 + 4x + 5$
  （ホーナー法を用いる）

また、main() 関数でそれらの関数を呼び出し、 以下の値を計算させて表示させなさい。

- 10!
- 初項 1、等比 2 の等比数列の第 10 項までの等比級数
- フィボナッチ数列の第 20 項目
- x = 2 のときの 4 次多項式 $f(x) = 1x^4 + 2x^3 + 3x^2 + 4x + 5$

実行例：

```
% ./a.out
10! = 3628800
2^0 + 2^1 + 2^2 + ... + 2^9 = 1023
fibonacci 20: 6765
f(2) = 57.00
```

関数を作る場合、問題 1 で考えた漸化式が参考になると思います。 例えば、このような

漸化式

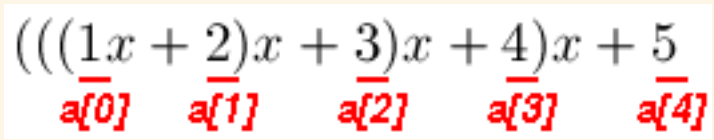
を関数にすると、次のようになります。

```
int tousa(int n){
  if( n==0 ){
    return 0;
  }else{
    return n + tousa(n-1);
  }
}
```

同様にして、他の関数も作ってみてください。 また、ホーナー法を用いる関数では簡単のため、 係数 a_n は次のようにグローバルな配列として宣言しても良い。

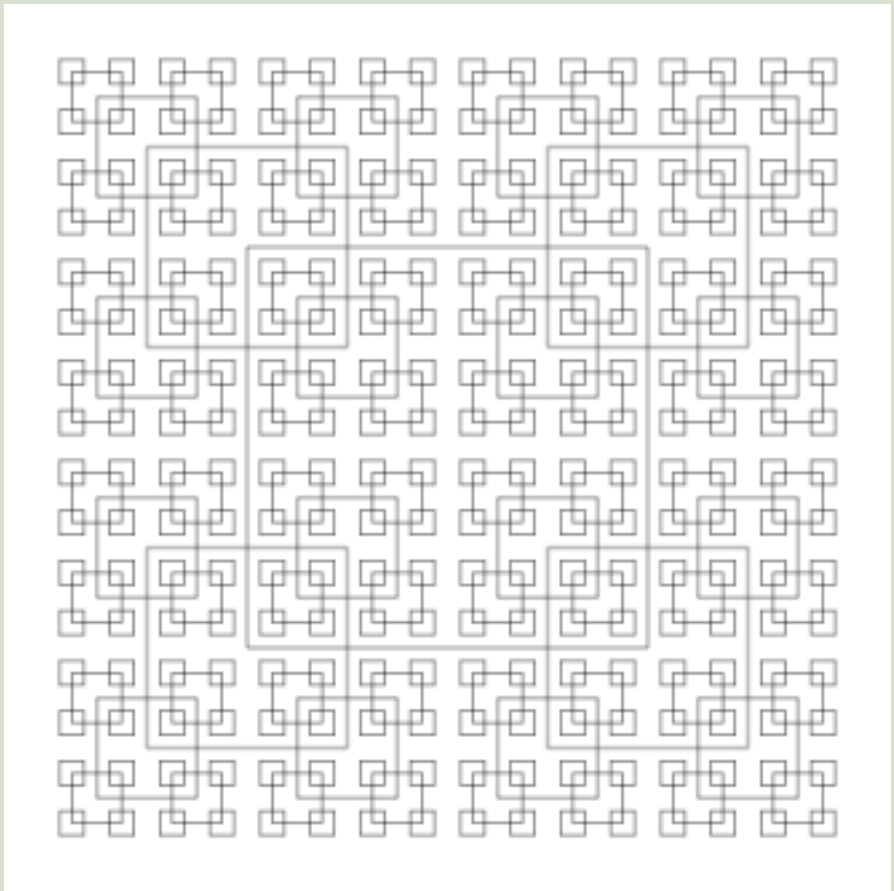
```
#define N 4
double a[N+1] = { 1, 2, 3, 4, 5 };
```

配列 a と展開式はこのように対応します。

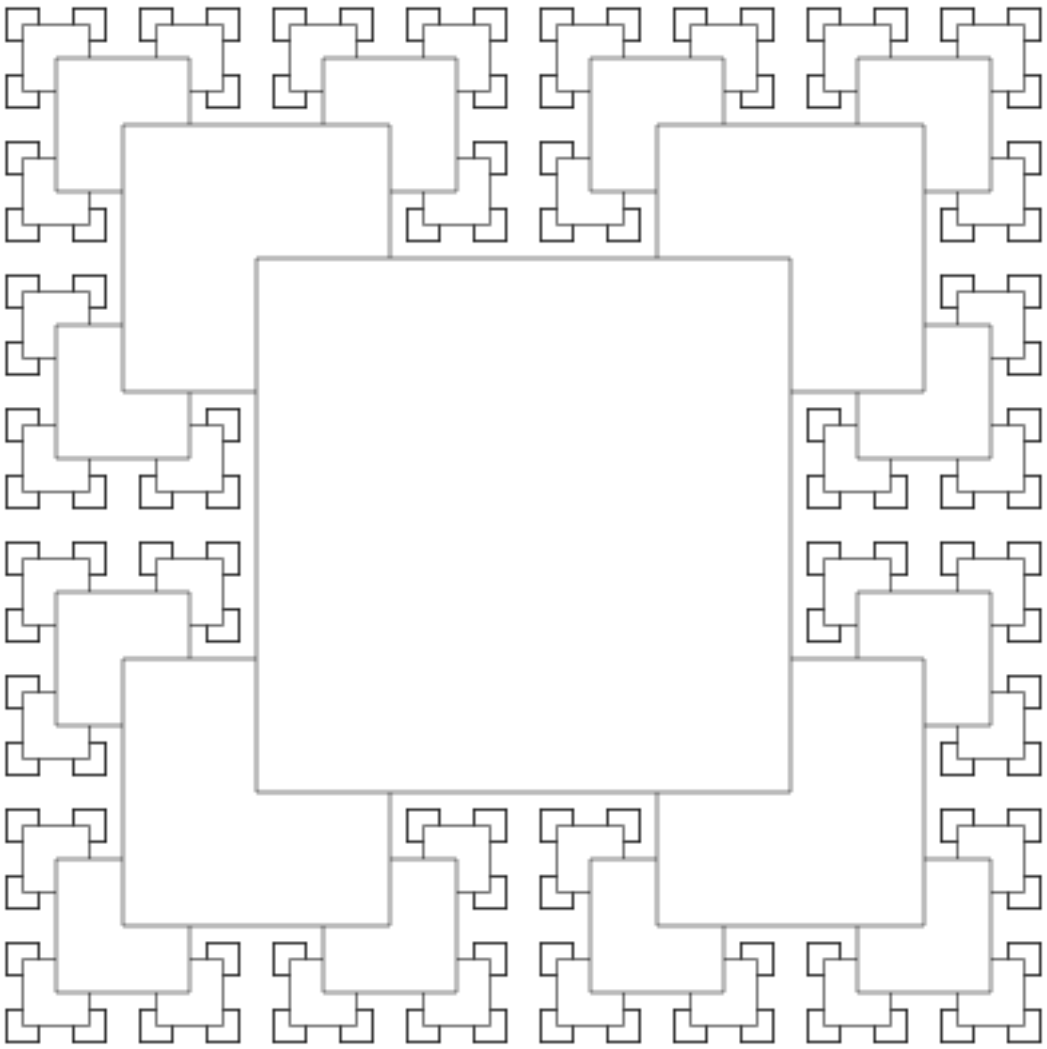


問題 3

ex04-3-skel.c は、実行すると star.ps という PostScript ファイルを生成する。 これを表示すると次のような画像になっている。



このプログラムを変更して、 このような画像 にしなさい。



ex04-3-skel.c の中で、描画しているのは star 関数 （アルゴリズム C 第 1 巻 p.68） です。 この関数を変更してください。この関数の中で使われている line 関数は 線を引く関数で、例えば `line(1,2,3,4)` とした場合は、 座標 (1,2) から (3,4) までの線が引かれます。

これ、結構難しいです。 できた人はコッホ曲線にも挑戦してみてください。

Written by わかまつなおき