3.3 大きな数の割り算

プログラム [Machin.java] に手を加えて、何百桁かの π の値を計算することは、そんなに簡単な話ではない。大きな数を扱うには配列を利用すればよいことは以前学んでいる。問題は大きな数の割り算をすることだ。マチンの公式には 239^{2n-1} が登場するので、これがすぐに大きな数になることは分かる。割り算の計算には掛け算が関わってくるし、さらにその値を足したり引いたりする必要もある。要するに π の計算には、大きな数の四則演算がすべて必要なのである。

準備として、**Java** に $1/239^m$ の計算をさせてみよう。配列を 10 用意すれば 40 桁の数が計算できるので、そのセンでプログラムを組んでみた。

programming list [Div239.java]

```
import java.text.DecimalFormat;

public class Div239 {

    final static int SUP = 5;

    public static void main(String[] args) {
        int[] p = new int[11]; // p[0] is invalid.

        p[1] = p[2] = p[3] = p[4] = p[5] = p[6] = p[7] = p[8] = p[9] = 0;
        p[10] = 10000;

        for(int m = 1; m <= SUP; m++) {
            for(int i = 10; i >= 1; i--) {
                p[i-1] = p[i-1] + (p[i] % 239) * 10000;
                p[i] = p[i] / 239;
            }
        }

        DecimalFormat df = new DecimalFormat("0000");

        System.out.print("0.");
        for(int i = 10; i >= 1; i--) {
            System.out.print(df.format(p[i]));
        }
        System.out.println();
    }
}
```

プログラムはべき乗の計算で作成した [PowerOf2.java] と同じ手法を用いている。

[Div239.java] は割り算をするのだから、一部が違うだけで残りはほとんど同じはずだ。是非 [PowerOf2.java] と見比べてほしい。では、一部の違いを中心に見ていこう。

まず 5:行目で SUP として上限を 5 に設定してあることがあげられる。この上限によって $1/239^5$ が計算されるのだが、上限を上げすぎると計算結果は確実に 0 となってしまうので、差し当たって 5 にしただけだ。ここでは小数点以下 40 位までの計算が可能なので、上限はもう少し上げられる。

8:行目で配列変数を 11 個準備した。以前と違って今度は p[0] がダミーだ。10:行目と 11:行目で配列変数を 0 と 10000 で初期化している。ここでは p[1] から p[9] までを 0 に、そして p[10] を 10000 にセットしている。p[1] がいちばん下の桁で p[10] がいちばん上の桁であることは、[PowerOf2.java] と同じである。そして、あえて p[10] に 10000 を与えたのには意味がある。p[10]...p[1] と並べると、10000...0000 なる数ができるが、ここでは 1.0000...0000 を作ったと解釈してほしい。この豆旅では、各 p[] には 4 桁の数を与えるはずなのに、p[10] に限って 5 桁の 10000 にしたのはそういうことなのだ。

13:–18:行目にかけて、上位の桁から順次 239 で割っているのだが、[PowerOf2.java] と違うのは p[i-1] と p[i] を用いて計算している点だ。もちろんこれには理由がある。割り算は掛け算と違うからだ。掛け算の場合、はみ出た数は繰り上がりとして処理される。p[i] に対する繰り上がりは p[i+1] へ向かう。割り算の場合は、余った数は次の割り算の処理へ回される。だから p[i] に対する余りは p[i-1] へ向かうのだ。

ところで 15:行目では、i = 1 になったとき p[0] が計算されてしまうことになるが、われわれの関心は p[1] までであるから、それ以下の結果はダミーが吸収するわけだ。まあ、そのために最後の桁には誤差が生じることがあるが仕方ない。

そして結果表示だ。表示は上の桁から順に並べればよい。そのために処理は 22:–25:行目までが必要となる。ただし、ここでは 1 を 239^m で割ることを想定した配列を作ったので、配列はすべて小数点以下の値である。よって 22:行目であらかじめ 0. を打ったわけだ。結果は正しく表示されるだろう。

TRY! 5:行目の final int SUP = 5 の値をもう少し大きくしてみよ。どのぐらいまで計算が可能か。

さて、計算結果が正しいかどうか確認するのは難しい。電卓では十分な桁数がないからだ。プロ

グラムが信頼がおけるか調べるには、`SUP` の値を `1` にするとよい。$1/239 = 0.00418410041841\ldots$ だから、確認はすぐにできる。

TRY! $1/239^m$ の計算はできた。では、$1/(2n-1)239^{2n-1}$ の計算をするプログラムを作ってみよ。