3.2 マチンの計略

円周率の値を計算するのに無限級数を用いた。ただ、そこで用いた級数は収束が遅いため計算機向きでなかった。計算機向きの級数に仕立て上げたのはマチン[1]である。詳しい経過を述べることはできないが、マチンは

$$\frac{\pi}{4} = 4\left(\frac{1}{1\cdot 5} - \frac{1}{3\cdot 5^3} + \frac{1}{5\cdot 5^5} - \cdots\right) - \left(\frac{1}{1\cdot 239} - \frac{1}{3\cdot 239^3} + \frac{1}{5\cdot 239^5} - \cdots\right)$$

という式をひねり出した。ここでもまた $\pi/4 = \cdots$ の形である。気になる人にだけ、そっと耳打ちしておこう。グレゴリーの式もマチンの式も $\tan\pi/4 = 1$ であること、すなわち $\pi/4 = \arctan 1$ であることが利用されているからだ。そして

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

である。以上。

え？　なおさら気になっただって？　知りたい人はきちんとした数学の書物を読んでみよう。

さて、今はマチンの式を

$$\pi = 4\left\{\left(\frac{4}{1\cdot 5} - \frac{1}{1\cdot 239}\right) - \left(\frac{4}{3\cdot 5^3} - \frac{1}{3\cdot 239^3}\right) + \left(\frac{4}{5\cdot 5^5} - \frac{1}{5\cdot 239^5}\right) - \cdots\right\} \tag{3.1}$$

と見て、π の値を計算することにしよう。

programming list [machin.cpp]

```
#include <iostream>

double fx(int nn) {
    int i, u = 1, v = 1;
    for(i = 1; i <= nn; i++) {
        u = u * 5;
        v = v * 239;
    }
    return (4. / (nn * u) - 1. / (nn * v));
}

int main() {
    int n, sgn = 1;
    double p = 0.0;
    for(n = 1; n <= 9; n += 2) {
        p = p + sgn * fx(n);
        sgn = -sgn;
    }
    std::cout << 4 * p << std::endl;

```

[1] ジョン・マチン (1685–1751)：イギリスの天文学者。

```
21:     return 0;
22: }
```

先に 12:行目からの main 関数を見ていこう。この部分は [greg.cpp] とほとんど同じだ。細かい点で多少の違いはあるが、本質的に違うのはただ 1ヶ所だけである。それは 16:行目だ。16:行目の sgn は double 型にキャストしていないことに注目してほしい。これは続く関数 fx(n) が double 型の値を返してくるので、sgn の型が何であれ正しい計算になる。そのためにキャストの必要がないのだ。

では、16:行目に使われている関数 fx(n) はどうなっているのだろう。[greg.cpp] では奇数分母の分数を交互に足したり引いたりしていた。ところがマチンの式 (3.1) はそう単純ではない。第 n 項では

$$\frac{4}{(2n-1)5^{2n-1}} - \frac{1}{(2n-1)239^{2n-1}} \tag{3.2}$$

という複雑な式を、交互に足したり引いたりする必要がある。奇数分母の分数と違い、239^n などの計算には繰り返し処理が必要なのだ。そこで関数にしてある。そして、ちょっと複雑になっている。

その複雑な計算を受け持つのが、関数 double fx(int nn) だ。double fx(int nn) は項番号を整数値 nn で受け取り、(3.2) を計算した後、double 型の値を返す。べき乗の計算は指数関数を使わずに 5:–8:行目にかけて for 文で処理している。見てのとおり nn が 3 を受け取れば 5^3 と 239^3 が、nn が 7 を受け取れば 5^7 と 239^7 が計算される仕組みだ。

そして 9:行目で (3.2) にあたる計算結果が返されるのだ。

ところで関数 double fx(int nn) の中身を見ると、整数値だけを扱いながら実数値を返していることに気付くだろう。キャストもしていない。その理由は 9:行目にある。9:行目では整数値でありながら 4. と 1. のような記述をしてある。小数点を記述することで整数値が実数値扱いになる、って前に言ったね。おかげで 9:行目の計算結果は double 型になって、関数 fx() が返す型と不整合が生じないようになっている。

再び main 関数に戻ろう。16:行目の fx(n) は (3.1) のたったひとつの項—ここでは ( ) 内の差をひとつの項と見ている—しか計算しない。π の値を求めるには fx(n) で計算した各項を、足したり引いたりする必要がある。それが main(){}の仕事である。

プログラムは 15:–18:行目にかけて n に奇数値が入る。そのたびに関数 fx(n) が fx(1), fx(3), fx(5), $\cdots$ で呼ばれるのだ。つまりプログラムは構造上、for 文の中に for 文が入れ子になっていることになる。

そして 19:行目でめでたく π の値が画面に現れるのである。

ところで [greg.cpp] では、for 文で n < 30000; を使ったが、ここではわずか n <= 9; までの繰り返しで済んでいる。このことからもマチンの式の収束の速さが分かるというものだ。

TRY! マチンの式で、どこまで精確に π の値に迫れるか試してみよ。