Chudnovskyアルゴリズム

$$\frac{1}{\pi} = 12 \sum_{n=0}^{\infty} \frac{(-1)^n (6n)! (A + B_n)}{(3n)! (n!) C^{3n+3/2}}$$

多倍長ライブラリgmp.hがインストールされていない場合、n=1を直接計算で検証

```
/* chudnovsky のアルゴリズム n = 1 の単純計算で検証 */
#include <stdio.h>
#include <math.h>
#define PI 3.14159265358979323846264338327950288
//              3.141592653589793238... 18桁が限界か
//Chudnovskyの公式
//n=0; 1/π~12*(13591409/640320^3/2)
//n=1; 1/π~12*((13591409/640320^3/2)-(6!*13591409+545140134)/3!*640320^9/2))

int main(void) {
long double pi_result;
long double inverted_pi;
long double pow_result1;
long double pow_result2;

/* 3/2乗、9/2乗の計算 */
pow_result1 = powl(640320L, 3.0L/2.0L);
printf("power_result1= %.40Lf\n",pow_result1);

pow_result2 = powl(640320L, 9.0L/2.0L);
printf("power_result2= %.20Lf\n",pow_result2);

inverted_pi = 12.0L * ((13591409.0L/pow_result1) - 120.0L * (13591409.0L + 545140134.0L)/
pow_result2 ) ;
printf("inverted_pi =%.40Lf\n", inverted_pi);

pi_result = 1/ inverted_pi;
printf("pi_result =%.50Lf\n", pi_result);

return 0;
}
```

計算結果:

```
power_result1= 512384047.99600074981572106480598449707031250000000
power_result2= 134519982239273124875993088.00000000000000000000
```

inverted_pi =0.318309886183790671538174771315610200872 5
pi_result =3.141592653589793238512808959406186204432742670178 41

結果は18桁まで

Chundnovskyアルゴリズムでn=14で、円周率を100桁まで計算する。

以下、http://bellard.org/pi/pi2700e9からソース引用

```
/* Calculate pi based on Chudnovsky algorithm, using GMP */
/* [1] Computation of 2700 billion decimal digits of Pi using a Desktop Computer,
      Fabrice Bellard, Feb 11 2010 (4th revision),
      http://bellard.org/pi/pi2700e9/pipcrecord.pdf */
#include <stdio.h>
#include <gmp.h>
#include <math.h>
 int main (int argc, char* argv[]) {
 int digits = 100;
 int prec = digits * log2(10);
 int i;
 int n = digits / 14;
 mpf_t pi, p, q, a, P, Q, T, A, B, C;
 /* Initialize Floating point numbers */
 mpf_set_default_prec(prec);
 mpf_init(pi);
 mpf_init(p);
 mpf_init(q);
 mpf_init(a);
 mpf_init(P);
 mpf_init(Q);
 mpf_init(T);
 mpf_init(A);
 mpf_init(B);
 mpf_init(C);
 /* Assignment */
 mpf_set_str(A, "13591409", 10);
 mpf_set_str(B, "545140134", 10);
 mpf_set_str(C, "640320", 10);
 mpf_set_ui(P, 1);
 mpf_set_ui(Q, 1);
 mpf_set_ui(T, 0);
 /* Main loop. about 14 * n digits precision */
 for (i = 1; i <= n; i++) {
   /* p = (2 * i - 1) * (6 * i - 5) * (6 * i - 1) */
   mpf_set_ui(p, (2 * i - 1) * (6 * i - 5) * (6 * i - 1));
```

```
    /* q = C^3 * i^3 / 24 */
    mpf_set_ui(q, i * i * i);
    mpf_mul(q, q, C);
    mpf_mul(q, q, C);
    mpf_mul(q, q, C);
    mpf_div_ui(q, q, 24);
    /* a = (-1)^i * (A + B * i) */
    mpf_mul_ui(a, B, i);
    mpf_add(a, a, A);
    if (i & 1) {
      mpf_neg(a, a);
    }
    /* P(0, N) = P(0, N - 1) p */
    mpf_mul(P, P, p);
    /* Q(0, N) = Q(0, N - 1) q */
    mpf_mul(Q, Q, q);
    /* T(0, N) = T(0, N - 1) * q + a * P */
    mpf_mul(T, T, q);
    mpf_mul(a, a, P);
    mpf_add(T, T, a);
  }
  /* pi = C ^ (1 / 2)     */
  mpf_sqrt(pi, C);
  /*      * C             */
  mpf_mul(pi, pi, C);
  /*      * Q             */
  mpf_mul(pi, pi, Q);
  /*      / (T + A * Q)   */
  mpf_mul(Q, Q, A);
  mpf_add(Q, Q, T);
  mpf_div(pi, pi, Q);
  /*      / 12            */
  mpf_div_ui(pi, pi, 12);
  mpf_out_str(stdout, 10, digits, pi);
  mpf_clear(pi);
  mpf_clear(p);
  mpf_clear(q);
  mpf_clear(a);
  mpf_clear(P);
  mpf_clear(Q);
  mpf_clear(T);
  mpf_clear(A);
  mpf_clear(B);
  mpf_clear(C);
  return 0;
}
```

MAC OSでの計算結果：

```
$ gcc pi.c -lgmp && ./a.out
0.31415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253421170 68e1
```

references:
http://itchyny.hatenablog.com/entry/20120304/1330870932
http://bellard.org/pi/pi2700e9からソース引用