57 PI

“Whoever is your servant is the greatest among you.”

π is an irrational number. To find out π with enough precision, many people have contributed since 2000BC. Before the invention of computers, the calculation of π was really hard. Even with the computers, the calculation of π is really a tough job. The problem with π is that it is defined as the ratio between perimeter and diameter of a circle. The value of π is not exactly 22/7, but it is approximately 22/7. And so you need more precision. First computer calculation of π was carried on ENIAC (Electronic Numerical Integrator and Computer) at Ballistic Research labs in September 1949. It took about 70 hours to calculate π to 2,037 decimal places! It was programmed to use Machine’s formula π = 16arctan(1/5) – 4arctan(1/239). It took almost 4000 years to find out π with good precision. Yes, in 1981AD only Kazunori Miyoshi and Kazuhiko Nakayama in Japan calculated π to 20,00,000 decimal places. They used an efficient portable program from the formula π = 32 arctan(1/10) – 4 arctan(1/239) – 16 arctan(1/515).

57.1 π

Officially accepted value of π to 3,200 decimal places is listed below. This listing would be very useful, if you want to work on this research-oriented program!

```
π = 3. 1415926535 8979323846 2643383279 5028841971 6939937510 5820974944
       5923078164 0628620899 8628034825 3421170679 8214808651 3282306647
       0938446095 5058223172 5359408128 4811174502 8410270193 8521105559
       6446229489 5493038196 4428810975 6659334461 2847564823 3786783165
       2712019091 4564856692 3460348610 4543266482 1339360726 0249141273
       7245870066 0631558817 4881520920 9628292540 9171536436 7892590360
       0113305305 4882046652 1384146951 9415116094 3305727036 5759591953
       0921861173 8193261179 3105118548 0744623799 6274956735 1885752724
       8912279381 8301194912 9833673362 4406566430 8602139494 6395224737
       1907021798 6094370277 0539217176 2931767523 8467481846 7669405132
       0005681271 4526356082 7785771342 7577896091 7363717872 1468440901
       2249534301 4654958537 1050792279 6892589235 4201995611 2129021960
       8640344181 5981362977 4771309960 5187072113 4999999837 2978049951
       0597317328 1609631859 5024459455 3469083026 4252230825 3344685035
       2619311881 7101000313 7838752886 5875332083 8142061717 7669147303
       5982534904 2875546873 1159562863 8823537875 9375195778 1857780532
       1712268066 1300192787 6611195909 2164201989 3809525720 1065485863
       2788659361 5338182796 8230301952 0353018529 6899577362 2599413891
       2497217752 8347913151 5574857242 4541506959 5082953311 6861727855
       8890750983 8175463746 4939319255 0604009277 0167113900 9848824012
       8583616035 6370766010 4710181942 9555961989 4676783744 9448255379
       7747268471 0404753464 6208046684 2590694912 9331367702 8989152104
       7521620569 6602405803 8150193511 2533824300 3558764024 7496473263
```

```
9141992726 0426992279 6782354781 6360093417 2164121992 4586315030
2861829745 5570674983 8505494588 5869269956 9092721079 7509302955
3211653449 8720275596 0236480665 4991198818 3479775356 6369807426
5425278625 5181841757 4672890977 7727938000 8164706001 6145249192
1732172147 7235014144 1973568548 1613611573 5255213347 5741849468
4385233239 0739414333 4547762416 8625189835 6948556209 9219222184
2725502542 5688767179 0494601653 4668049886 2723279178 6085784383
8279679766 8145410095 3883786360 9506800642 2512520511 7392984896
0841284886 2694560424 1965285022 2106611863 0674427862 2039194945
0471237137 8696095636 4371917287 4677646575 7396241389 0865832645
9958133904 7802759009 9465764078 9512694683 9835259570 9825822620
5224894077 2671947826 8482601476 9909026401 3639443745 5305068203
4962524517 4939965143 1429809190 6592509372 2169646151 5709858387
4105978859 5977297549 8930161753 9284681382 6868386894 2774155991
8559252459 5395943104 9972524680 8459872736 4469584865 3836736222
6260991246 0805124388 4390451244 1365497627 8079771569 1435997700
1296160894 4169486855 5848406353 4220722258 2848864815 8456028506
0168427394 5226746767 8895252138 5225499546 6672782398 6456596116
3548862305 7745649803 5593634568 1743241125 1507606947 9451096596
0940252288 7971089314 5669136867 2287489405 6010150330 8617928680
9208747609 1782493858 9009714909 6759852613 6554978189 3129784821
6829989487 2265880485 7564014270 4775551323 7964145152 3746234364
5428584447 9526586782 1051141354 7357395231 1342716610 2135969536
2314429524 8493718711 0145765403 5902799344 0374200731 0578539062
1983874478 0847848968 3321445713 8687519435 0643021845 3191048481
0053706146 8067491927 8191197939 9520614196 6342875444 0643745123
7181921799 9839101591 9561814675 1426912397 4894090718 6494231961
5679452080 9514655022 5231603881 9301420937 6213785595 6638937787
0830390697 9207734672 2182562599 6615014215 0306803844 7734549202
6054146659 2520149744 2850732518 6660021324 3408819071 0486331734
6496514539 0579626856
```

57.2 Program

The following C program is one of the implementations to find π. Once someone else provided me this program. I don’t know who is the real author of this program. On Pentium III machine, it just took fraction of seconds to calculate π! I have compared the output of this program with official-accepted value of π. This program gives right π value upto 3199 decimal places; from 3200th decimal place onwards the accuracy is lost. Anyhow this is a good program!

```
#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>

long kf, ks;
long far *mf, far *ms;
long cnt, n, temp, nd;
long i;
long col, col1;
long loc, arr[21];
```

```
void Shift( long far *l1, long far *l2, long lp, long lmod )
{
   long k;
   k = (*l2) > 0 ? (*l2) / lmod: -(-(*l2) / lmod) - 1;
   *l2 -= k * lmod;
   *l1 += k * lp;
} /*--Shift( )---------*/

void YPrint( long m )
{
   if ( cnt<n )
     {
       if ( ++col == 11 )
        {
           col = 1;
           if ( ++col1 == 6 )
             {
              col1 = 0;
              printf( "\n" );
              printf("%4ld",m%10);
             }
            else
              printf("%3ld",m%10);
        }
       else
            printf("%ld",m);
            ++cnt;
     }
} /*--YPrint( )-----------*/

void XPrint( long m )
{
   long ii, wk, wk1;
   if ( m < 8 )
      {
        for( ii = 1; ii <= loc;  )
            YPrint( arr[(int)(ii++)] );
        loc = 0;
      }
     else if ( m > 9 )
      {
          wk = m / 10;
          m %= 10;

    for( wk1 = loc; wk1 >= 1; --wk1 )
            {
              wk += arr[(int)wk1];
              arr[(int)wk1] = wk % 10;
              wk /= 10;
            }
      }
     arr[(int)(++loc)] = m;
} /*--XPrint( )---------*/

int main( int argc, char *argv[] )
{
   int i=0;
   char *endp;
   arr[i++] = 0;
   if ( argc < 2 )
      {
         printf( "Syntax: PI digits \n\a");
         exit(1);
      }
   n = strtol( argv[1], &endp, 10 );
   if ( (mf = farcalloc( n + 3L, (long)sizeof(long)) ) == NULL )
      {
         printf( "Error: Memory not sufficient! \n\a" );
         exit(1);
       }
   if ( (ms = farcalloc( n + 3L, (long)sizeof(long)) ) == NULL )
      {
         printf( "Error: Memory not sufficient! \n\a" );
         farfree( mf );
         exit(1);
      }
   printf( "\nApproximation of PI to %ld digits\n", (long)n );
   cnt = 0;
   kf = 25;
   ks = 57121L;
   mf[1] = 1;
   for( i = 2; i <= n; i += 2 )
      {
         mf[i] = -16;
         mf[i+1] = 16;
      }
   for( i = 1; i <= n; i += 2 )
      {
         ms[i] = -4;
         ms[i+1] = 4;
      }
   printf( "\n 3." );
   while( cnt < n )
```

```
      {
       for( i = 0; ++i <= n - cnt;  )
          {
            mf[i] *= 10;
            ms[i] *= 10;
          }
       for( i = (int)(n - cnt + 1); --i >= 2;  )
          {
            temp = 2 * i - 1;
            Shift( &mf[i - 1], &mf[i], temp - 2, temp * kf );
            Shift( &ms[i - 1], &ms[i], temp - 2, temp * ks );
          }
       nd = 0;
       Shift( (long far *)&nd, &mf[1], 1L, 5L );
       Shift( (long far *)&nd, &ms[1], 1L, 239L );
       XPrint( nd );
      }
   printf( "\n\nCalculations Completed!\n" );
   farfree( ms );
   farfree( mf );
   return(0);
} /*--main( )----------*/
```