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 $\pi = 16\arctan(1/5) - 4\arctan(1/239)$. 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 $\pi = 32\arctan(1/10) - 4\arctan(1/239) - 16\arctan(1/515)$. 

57.1 $\pi$

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

\[
\pi = 3.1415926535 8979323846 2643383279 5028841971 6939937510 5820974944 \\
5923078164 0628620899 8626034825 3421170679 8214088651 3282306647 \\
0938446095 5058223172 5359408128 4811174501 9171536436 7892590360 \\
6466229489 5493081996 4428810975 6659334461 2847564823 3786783165 \\
2712019091 4564856692 3460348610 4543266482 1339360726 0249141273 \\
7245870666 0331558817 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 8464781846 7669405132 \\
0005681271 4526356082 7785717342 7577896091 7363717872 1468440901 \\
2249534301 4654958537 150792279 6892589235 4201995611 2129021960 \\
8640344181 5981362977 4771309960 5187072113 4999998937 2978049951 \\
0597317328 1609631859 5024459455 3469083026 4252230825 334685035 \\
2619311881 7101000313 7838752886 5875332083 8142061717 7669147303 \\
5982549049 2875546873 1159562863 8823537875 9375195778 1857780532 \\
1712268066 1300192787 6611195909 2164201989 3809525720 1064585863 \\
2788659361 5338182796 8230301952 0353018582 6899577362 2599413891 \\
2497217752 8347913151 5574857242 4541506959 5082953311 6861727855 \\
8890750983 8175463746 4939319255 0604009277 0167113900 9848824012 \\
8583616035 3670766010 4710181942 9555961989 4676783744 9448255379 \\
7747268471 0404753464 6208046684 2590694912 9331367702 8989152104 \\
7521620569 6602405803 8150193511 2533824300 3558764024 7496473263 \\

57.2 Program

The following C program is one of the implementations to find \( \pi \). 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 \( \pi \)! I have compared the output of this program with official-accepted value of \( \pi \). This program gives right \( \pi \) value upto 3199 decimal places; from 3200th decimal place onwards the accuracy is lost. Anyhow this is a good program!

```c
#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 )
```c
{
    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( )----------*/
```