Code:

**import** java.io.\*;

**import** java.util.\*;

**import** java.math.\*;

**public** **class** cxz {

**static** **long** *x*=0l,*y*=0l;

**static** **long** *max*=1000000000l;

**static** Random *random* = **new** Random();

**static** **double** *res*=0;

**static** Scanner *in*=**new** Scanner( System.***in***);

**public** **static** **void** main(String[] args){

**int** total = 0;

**for**(**long** i = 1;;i++){

*x* =*random*.nextLong()%(*max*+1);

*y* =*random*.nextLong()%(*max*+1);

**if**(*x*\**x*+*y*\**y* <*max*\**max*)

total++;

**if**(i%10000000==0)

System.***out***.println(4\*((**double**)total/i)+" "+total+" "+i);

}

}

}

思路：

在运算pi的值的时候，时获取随机数字后，确定求得对应（x，y）是否在四分之一单位圆中。

优点：

相较于原本的从（0，1）中获得小数，通过整型数字我们可以令取的随机数范围（0，max）这样就相当于在单位1的方格中取了精度为（1/max）的一个小数。我们可以通过增加max的值，进而提高我们计算出来的pi值精度。

注意事项：

由于计算机的“不足之处”我们无法肆无忌惮的将max取至无穷大。不难发现，在运算中我们要进行乘方运算，这是若进行乘方运算的数字大于int的最大值，呢么他的平方将超出long的最大值。

玄学理论：

由于随机数是由一个确定的数字通过公式变换等方法演算而来，所以我们的随机数并不一定是随机的。而我们在对代码进行大量测试时，也发现当精度趋近于3.1415后，就无法进一步精确。但根据我们的想法，我们的精度应为1/（(2^31-1）^2)两者显然相差很远，这佐证了我们对随机数不随机的想法。

猜测：虽然我们的随机数范围看起来越大精确度越高，但由于随机数的存在，我们的高精度并不能完全的体现其价值。

综上：求pi值的最好的方法还应该是依靠数学的公式等，进行严格的逼近，这样才能更加准确地 /\*且不用担心答案是不是真正的pi值\*/ 获得答案

求pi正道是沧桑

#include "stdafx.h"

long a = 10000, b, c = 2800, d, e, f[2801], g;

int main() {

for (; b - c;) f[b++] = a / 5;

for (; d = 0, g = c \* 2; c -= 14, printf("%.4d", e + d / a), e = d % a)

for (b = c; d += f[b] \* a, f[b] = d % --g, d /= g--, --b; d \*= b);

getchar();

return 0;

}//来自百度

求pi正道：https://www.cnblogs.com/PegasusWang/archive/2013/03/17/2965189.html