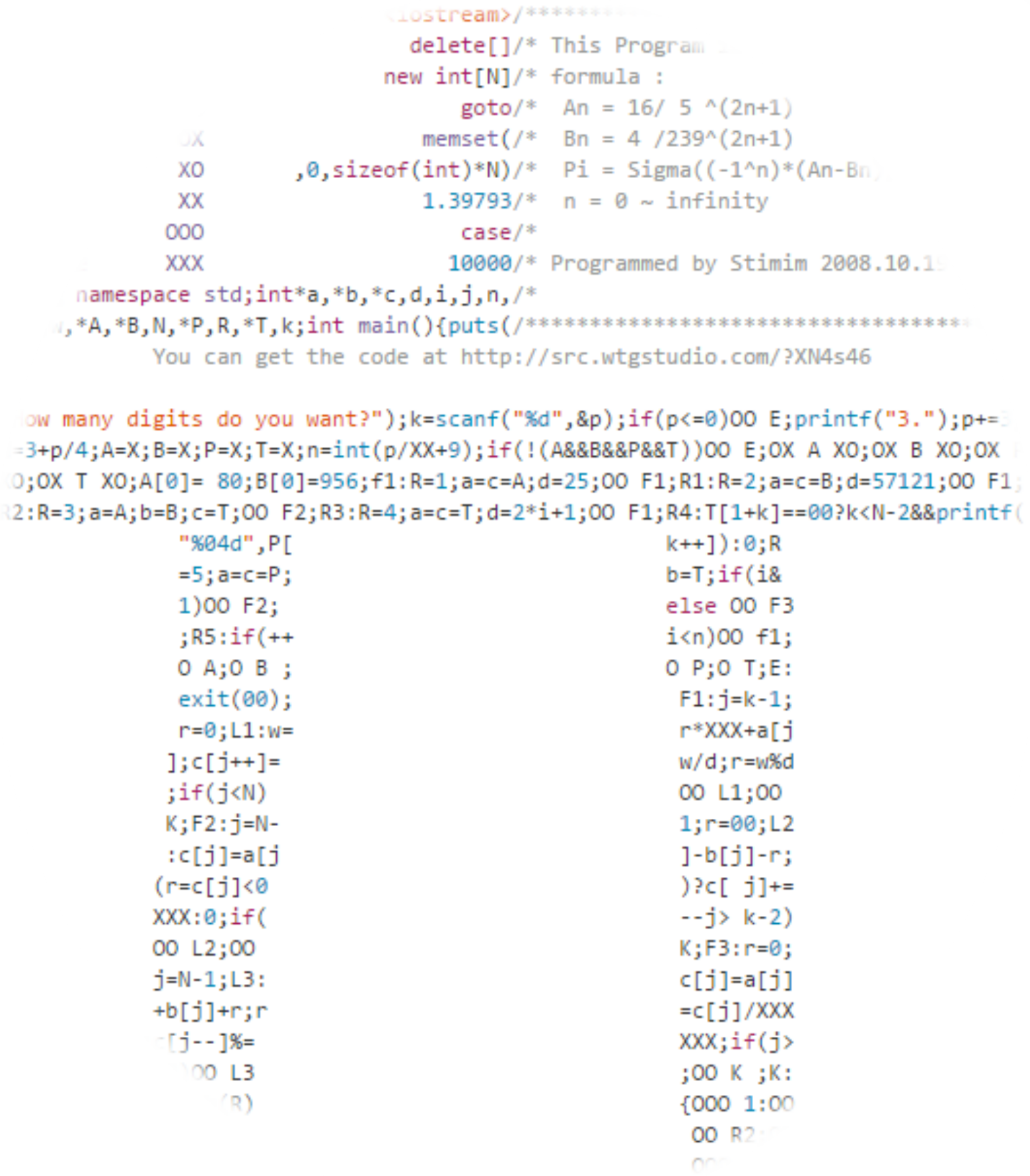
# Hello threads,

That’s study on co-op

# Compiling and running

- gcc and g++
  - >> gcc **-pthread** test_pthread.c -o good.out
  - Input: test_pthread.c
  - Output: good.out
- Run the program
  - >> **./**good.out

$\pi =$ 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 ......



```
                              iostream>/*********
                               delete[]/* This Program
                             new int[N]/* formula :
                                   goto/*  An = 16/ 5 ^(2n+1)
             OX                 memset(/*  Bn = 4 /239^(2n+1)
             XO       ,0,sizeof(int)*N)/*  Pi = Sigma((-1^n)*(An-Bn)
             XX                 1.39793/*  n = 0 ~ infinity
            OOO                    case/*
            XXX                   10000/* Programmed by Stimim 2008.10.1
     namespace std;int*a,*b,*c,d,i,j,n,/*
   w,*A,*B,N,*P,R,*T,k;int main(){puts(/********************************
           You can get the code at http://src.wtgstudio.com/?XN4s46

 ow many digits do you want?");k=scanf("%d",&p);if(p<=0)OO E;printf("3.");p+=
=3+p/4;A=X;B=X;P=X;T=X;n=int(p/XX+9);if(!(A&&B&&P&&T))OO E;OX A XO;OX B XO;OX
O;OX T XO;A[0]= 80;B[0]=956;f1:R=1;a=c=A;d=25;OO F1;R1:R=2;a=c=B;d=57121;OO F1
2:R=3;a=A;b=B;c=T;OO F2;R3:R=4;a=c=T;d=2*i+1;OO F1;R4:T[1+k]==00?k<N-2&&printf(
             "%04d",P[                            k++]):0;R
             =5;a=c=P;                            b=T;if(i&
             1)OO F2;                             else OO F3
             ;R5:if(++                            i<n)OO f1;
             O A;O B ;                            O P;O T;E:
             exit(00);                             F1:j=k-1;
             r=0;L1:w=                             r*XXX+a[j
            ];c[j++]=                              w/d;r=w%d
            ;if(j<N)                               OO L1;OO
            K;F2:j=N-                              1;r=00;L2
            :c[j]=a[j                              ]-b[j]-r;
           (r=c[j]<0                               )?c[ j]+=
           XXX:0;if(                               --j> k-2)
           OO L2;OO                                K;F3:r=0;
           j=N-1;L3:                               c[j]=a[j]
           +b[j]+r;r                               =c[j]/XXX
           c[j--]%=                                XXX;if(j>
             OO L3                                 ;OO K ;K:
                (R)                                {OOO 1:OO
                                                    OO R2
                                                    OO
```

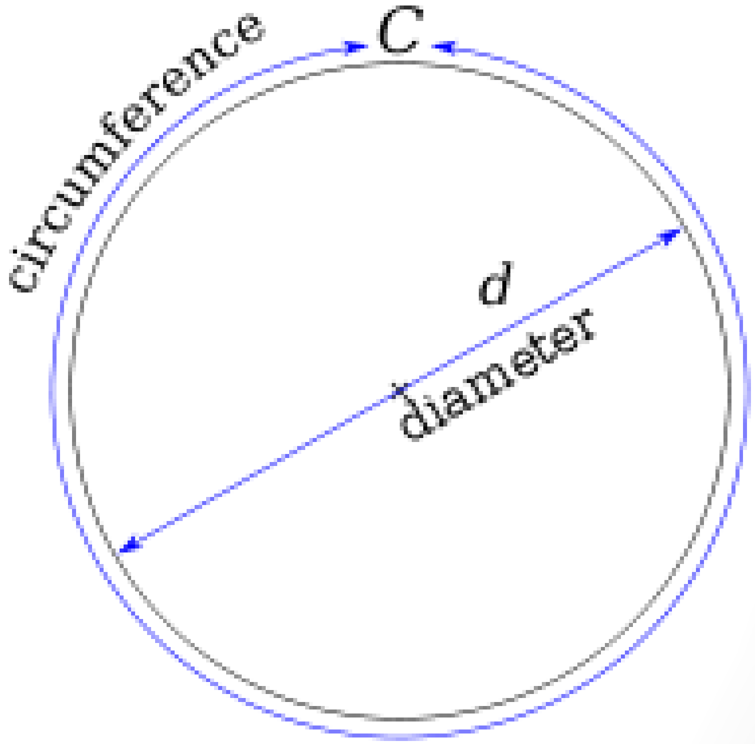
# God-like

```
int a=10000,b,c=2800,d,e,f[2801],g;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);}
```

```
3141592653589793238462643383279502884197169399375105820974944592307816
4062862089986280348253421170679821480865132823066470938446095505822317
2535940812848111745028410270193852110555964462294895493038196442881097
5665933446128475648233786783165271201909145648566923460348610454326648
2133936072602491412737245870066063155881748815209209628292540917153643
6789259036001133053054882046652138414695194151160943305727036575959195
3092186117381932611793105118548074462379962749567351885752724891227938
1830119491298336733624406566430860213949463952247371907021798609437027
7053921717629317675238467481846766940513200056812714526356082778577134
2757789609173637178721468440901224953430146549585371050792279689258923
5420199561121290219608640344181598136297747713099605187072113499999983
7297804995105973173281609631859
```

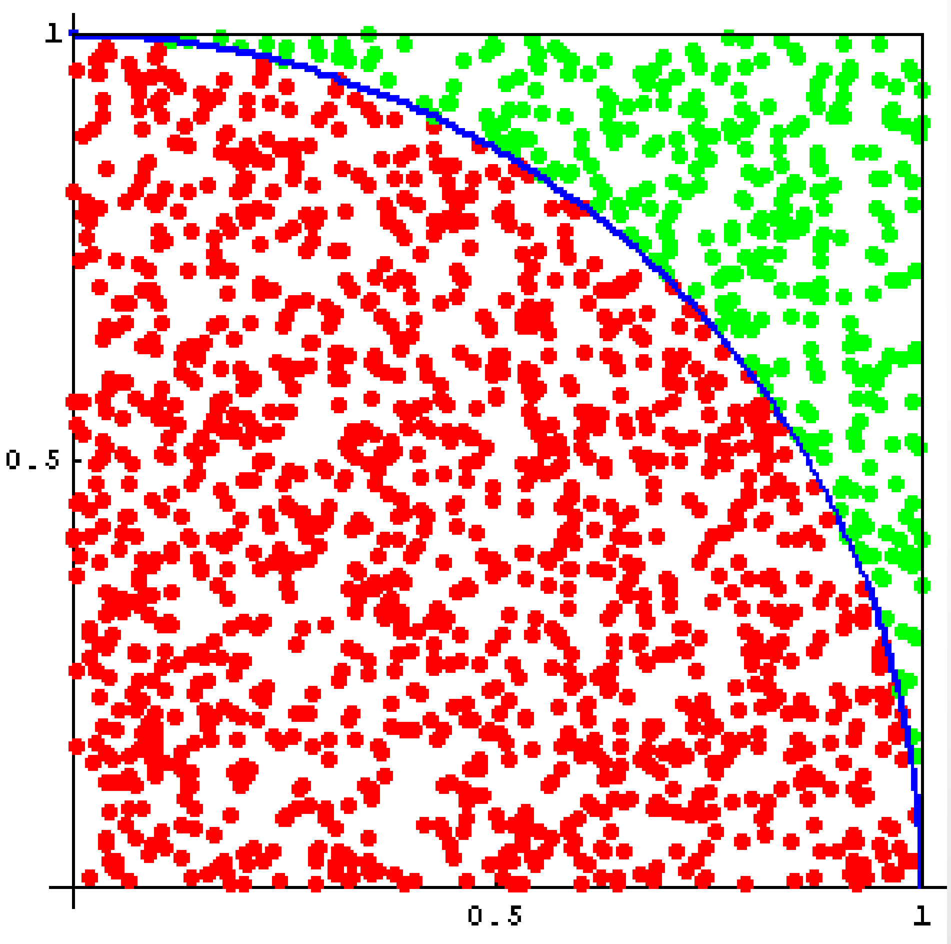
# Define

$$\pi = C/d$$



# Define

- Monte-Carlo method



# Define

- PI == 4 * arctan(1)
- PI == 4 * ( arctan(1/2) + arctan(1/3) )
- …
- John Machin’s formula
  - http://en.wikipedia.org/wiki/John_Machin

$$\frac{\pi}{4} = 4 \arctan \frac{1}{5} - \arctan \frac{1}{239}$$

- Don’t forget to use Power series and Taylor series
- to solve arctan(x)
  - [http://en.wikipedia.org/wiki/Power_series](http://en.wikipedia.org/wiki/Power_series)
  - [http://en.wikipedia.org/wiki/Taylor_series](http://en.wikipedia.org/wiki/Taylor_series)

# Define

- Wallis Product
  - http://en.wikipedia.org/wiki/Wallis_product

$$\prod_{n=1}^{\infty}\left(\frac{2n}{2n-1}\cdot\frac{2n}{2n+1}\right)=\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\frac{8}{7}\cdot\frac{8}{9}\cdots=\frac{\pi}{2}$$

  - http://mathworld.wolfram.com/WallisFormula.html

- More, more and more…
  - http://mathworld.wolfram.com/PiFormulas.html

# Problem is…

- How to divide your work into multiple thread?
  - Except the main thread, you need to divide your work into 4 threads to do the calculation (it should be controllable)
- How many digits of precision you made?
- How to count and control computing time?

- And, how to use pthread XD
  - http://dragonspring.pixnet.net/blog/post/32963482
  - http://en.wikipedia.org/wiki/Fork%E2%80%93join_model

# Homework

- ./s1234567_pi.out [argument] [M threads]

  Pregame name input Use M threads to calc.

- There are three possible types:
- Argument = [N digits] //print N digits
  - The best way, full scores.
- Argument = [time(ms)]//calculate in N microsecond
  - 80% scores
- Argument = [iteration] //perform N iteration
  - 70% scores

  define by yourself

# Requirement

- Divide your work into multiple threads – REQUIRED
  - No multi-tasking, No points
  - Share data between your threads
  - Please use Pthread or std::thread to do this assignment
- Let the number of thread become controllable
- Count and control your computing time
  - The time used by outputting could be ignored.

# Performance analysis

- #include <time.h>

  - clock_t **start_time**, **end_time**;
  - float **total_time** = 0;

  - **start_time** = clock(); /* mircosecond */
  - … do something …
  - **end_time** = clock();

  - **total_time** = (float)(**end_time** - **start_time**)/**CLOCKS_PER_SEC**;
  - printf("Time : %f sec \n", **total_time**);