







//Shrabanti Basu

//April 3, 2016

//Program 7

//The purpose of this program is to find the value of PI using Monte Carlo's method.

#include <iostream>

#include <iomanip>

#include <ctime> //for the time function

#include <cstdlib> //for rand and srand

using namespace std;

//function prototype that generates random numbers between zero and the integer passed as argument

double numGenerator(int);

int main()

{

cout << "Shrabanti Basu\n";

cout << "April 3, 2016\n";

cout << "Program 7\n";

cout << "The purpose of this program is to find the value of PI using MonteCarlo's method.\n\n";

cout << "The value of PI can be can be found as the ratio of areas\n"

<< "of a circle located within a square, of side 2r.\n"

<< "This program generates number of points that are within the square.\n"

<< "We check how many of those points are also inside the circle.\n"

<< "The user can specify the radius of circle and number of trials.\n\n";

cout << "Enter the radius of the circle and the number of trials and\n"

<< "the program will generate the value of PI upto 6 precisions.\n\n";

const double PI = 3.14159265; //we test the value of pi against this value

cout << setprecision(6) << fixed << showpoint;

int radius, trials; //radius of circle and number of trials

double x, y; //x and y co-ordinates

//counters;

//if the point generated is inside the circle, counterC is incremented

//counterS is incremented at each iteration of the circle

int counterS, counterC;

double distance; //distance between origin (0,0) and point represented by (x, y)

double pi; //to store the calculated value of pi from the ratio of points lying inside the circle and square

double accuracy; //to hold accuracy factor between pi and PI

double difference; //to hold the absolute difference between pi and PI

//the seed gets the time value which is different for every run of the program

//use this seed value with the srand function to generate a random number

unsigned seed = time(0);

srand(seed);

//get the radius and number of trials from the user

cout << "Enter the radius of the circle: ";

cin >> radius;

cout << "Enter the number of trials: ";

cin >> trials;

//the outermost loop runs for the number of trials entered by user

for (int i = 1; i <= trials; i++)

{

//counters initialized to zero at each iteration of loop

counterS = 0;

counterC = 0;

// initialize accuracy to 0.1 and difference to 0.2 each time as intial conditions

//for the while loop to start

accuracy = 0.1;

difference = 0.2;

cout << "Trial " << i << endl;

cout << "Accuracy " << "\t" << "Circle count " << "\t" << "Square count " << "\t" << "PI" << endl << endl;

//generate accuracy for 6 precisions

for (int count = 1; count <= 6; count++)

{

while (difference >= accuracy)

{

//call the numGenerator fuction twice and save its return values in x and y co-ordinates

x = numGenerator(radius);

y = numGenerator(radius);

counterS++; //increment counterS for each iteration

distance = sqrt(pow(x, 2.0) + pow(y, 2.0)); //calculate distance

//if the distance is less than the radius only then increment counterC

if (distance <= radius)

counterC++;

//calculate the value of pi from the ratios of counters multiplied by 4

pi = 4 \* ((double)counterC / (counterS));

//calculate absolute difference between given value of PI and calculated pi

difference = abs(PI - pi);

}

//print values for counters, pi when desired accuracy level is reached

cout << accuracy << "\t\t" << counterC << "\t\t" << counterS << "\t\t" << pi << endl << endl;

//increase accuracy level by 10 and run loop again

accuracy /= 10;

}

}

return 0;

}

/\* Definition of numGenerator function.

The function generates random number and sends it back to calling main function.

The function takes an integer value as its argument.

A random number is generated and an alogorithm is used to

transform the random nuumber between zero and the argument passed through num

\*/

double numGenerator(int num)

{

double randNum; //to store a random number generated by rand function and save it as double

double number; //to store the random number that generated by the algorithm

//generate a random number, divide it by built in const variable RAND\_MAX

//now randNum will less than or equal to 1.

//Multiply the variable randNum by num and the value will now be less than or equal to num

//since the origin is (0,0) this will generate a number between 0 and value passed in num

randNum = (double)rand() / RAND\_MAX;

number = randNum \* num;

return number;

}