Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №3**

По дисципліні «Теорія ймовірності»

Виконав: Перевірив:

Студент групи ІО-21 Викладач

Коноз А.О. Марковський О. П.

Дата здачі\_\_\_\_\_\_\_\_\_\_\_\_\_

Захищено з балом\_\_\_\_\_

Київ 2013

**package** ClassTB;

**import** java.util.Random;

**public** **class** ClassTB {

**double** array[][] = **new** **double**[1000][2];

**double** vectory []= **new** **double** [1000];

**double** vectorx [] = **new** **double** [1000];

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

**public** ClassTB(){

**double** dy = 2/1000d;

**double** y = 0.0d;

**double** s = 0.0d;

**for** (**int** i = 1; i < array.length; i++) {

s += getS(y, dy);

y += dy;

array[i][0] = s;

array[i][1] = y;

}

}

**public** **double** getS(**double** y, **double** dy){

**return** ((2/Math.*PI*)\*Math.*sqrt*((1-Math.*pow*((y - 1), 2)))) \* dy;

}

**public** **double** nextY(){

**double** ri = random.nextDouble();

**double** y = 0.0;

**for** (**int** i = 0; i < array.length - 1; i++) {

**if** ((array[i][0] < ri) && ( ri < array[i + 1][0])){

y = array[i][1];

**break**;

}

}

**double** result = y;

**return** result;

}

**private** **void** fillingy(){

**for** (**int** i = 0; i < vectory.length; i++) {

vectory[i] = nextY();

}

}

**private** **double** getM(**double** array[]){

**double** s = 0.0d;

**for** (**int** i = 0; i < array.length; i++) {

s += array[i];

}

**double** m = s/1000d;

**return** m;

}

**private** **double** getMx(**double** array[]){

**double** s = 0.0d;

**for** (**int** i = 0; i < array.length; i++) {

s += array[i];

}

**double** m =1- s/1000d;

**return** m;

}

**public** **double** getDy(**double** my){

**double** dy = 0.0d;

**double** sum = 0.0d;

**for** (**int** i = 0; i < vectory.length; i++) {

sum += Math.*pow*((vectory[i] - my),2);

}

dy = sum/1000d;

**return** dy;

}

**public** **double** getDx(**double** mx){

**double** dy = 0.0d;

**double** sum = 0.0d;

**for** (**int** i = 0; i < vectorx.length; i++) {

sum += Math.*pow*((vectorx[i] - mx),2);

}

dy = sum/1000d;

**return** dy;

}

**private** **double** getQ(**double** du){

**return** Math.*pow*(du, 0.5);

}

**private** **void** fillingx(){

**double** x = 0.0d;

**double** s = 0.0d;

**double** partX = 0.0;

**for** (**int** i = 0; i < vectory.length; i++) {

x = 1 - Math.*sqrt*(1 - (vectory[i] - 1)\*(vectory[i] - 1));

s = (2.0d/Math.*PI*)\*(1 - x);

partX = (s\*random.nextDouble())/((2.0d/Math.*PI*));

vectorx[i] = x + partX;

}

}

**private** **double** getCov(**double** mx, **double** my){

**double** cov = 0.0d;

**for** (**int** i = 0; i < vectory.length; i++) {

cov += (vectory[i] - my)\*(vectorx[i] - mx);

}

**return** cov/1000d;

}

**private** **double** getP(**double** cov, **double** qx, **double** qy){

**return** cov/(qx \* qy);

}

**public** **void** showDimenshion(){

**double** mx;

**double** my;

**double** dy;

**double** dx;

**double** qx;

**double** qy;

**double** cov;

**double** p;

fillingy();

fillingx();

my = getM(vectory);

dy = getDy(my);

qy = getQ(dy);

mx = getM(vectorx);

dx = getDx(mx);

qx = getQ(dx);

cov = getCov(mx, my);

p = getP(cov, qx, qy);

System.*out*.println("dy = " + dy);

System.*out*.println("my = " + my);

System.*out*.println("qy = " + qy);

System.*out*.println();

System.*out*.println("mx = " + mx);

System.*out*.println("dx = " + dx);

System.*out*.println("qx = " + qx);

System.*out*.println();

System.*out*.println("cov = " + cov);

System.*out*.println("p = " + p );

}

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

ClassTB ob = **new** ClassTB();

ob.showDimenshion();

}

}