Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**ЛАБОРАТОРНА РОБОТА № 3**

з дисципліни «Теорія імовірності»

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІО-64

Потапенко Д.О.

Залікова - 6421

ПЕРЕВІРИВ:

доц. Марковський О.П.

Київ – 2017

Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**ЛАБОРАТОРНА РОБОТА № 3**

з дисципліни «Теорія імовірності»

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІО-64

Потапенко Д.О.

Залікова - 6421

ПЕРЕВІРИВ:

доц. Марковський О.П.

Київ – 2017

Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**ЛАБОРАТОРНА РОБОТА № 3**

з дисципліни «Теорія імовірності»

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІО-64

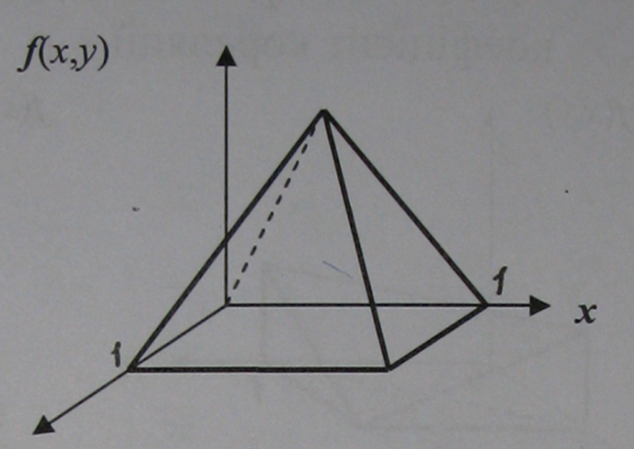
Потапенко Д.О.

Залікова - 6421

ПЕРЕВІРИВ:

доц. Марковський О.П.

Київ – 2017



**package** com.company;  
  
**import** java.util.Arrays;  
  
**public class** Main {  
 **public static void** main(String[] args) {  
 Generator gen = **new** Generator();  
 gen.gen();  
 *// System.out.println(Arrays.toString(gen.getX()));  
 //System.out.println(Arrays.toString(gen.getY()));* **double** sumX = 0;  
 **double** sumY = 0;  
 **for** (**int** i = 0; i < gen.Mx().**length**; i++) {  
 sumX += gen.Mx()[i];  
 sumY += gen.My()[i];  
 }  
  
 System.***out***.println(**"Mx "** + Math.*abs*(sumX) \* 2.5);  
 System.***out***.println(**"My "** + Math.*abs*(sumY) \* 3.8);  
 System.***out***.println();  
 System.***out***.println(**"SigmaX "** + gen.getSigmaX());  
 System.***out***.println(**"SigmaY "** + gen.getSigmaY());  
 System.***out***.println();  
 System.***out***.println(**"Radius "** + gen.getRadius());  
 }  
}

**package** com.company;  
  
**import** java.util.Random;  
  
**public class** Generator {  
 **int array\_count** = 1000;  
 **private double** [] **X** = **new double** [**array\_count**];  
 **private double** [] **Y** = **new double** [**array\_count**];  
 **private double** [] **Mx** = **new double** [4];  
 **private double** [] **My** = **new double** [4];  
 **public void** gen(){  
 Random rand = **new** Random();  
  
 **double** Mx1 = 0;  
 **double** My1 = 0;  
  
 **double** Mx2 = 0;  
 **double** My2 = 0;  
  
 **double** Mx3 = 0;  
 **double** My3 = 0;  
  
 **double** Mx4 = 0;  
 **double** My4 = 0;  
  
  
 **double** r=0;  
 **for** (**int** i=0; i<1000;i++){  
  
  
 r = rand.nextDouble();  
  
 **if** (i<250){  
  
 **double** x\_tmp = 0;  
 x\_tmp += 0.5\*(Math.*pow*((Math.*sqrt*(4 \* r \* r - 8 \* r + 5) + 2 \* r - 2), (0.6666)));  
 x\_tmp -= 1/(2 \* Math.*pow*(Math.*sqrt*(4 \* r \* r - 8 \* r + 5) + 2 \* r - 2, (0.6666)));  
 x\_tmp += 1/2;  
  
 **double** y\_tmp = Math.*sqrt*( Math.*abs*( r\*(2\*x\_tmp\*x\_tmp - 2\*x\_tmp +1) ) );  
  
 **X**[i] = x\_tmp;  
 **Y**[i] = y\_tmp;  
  
 Mx1 += x\_tmp; *//математичнеочікуваннядля Х* My1 += y\_tmp; *// математичнеочікуваннядля Y* }**else if** (i>=250 && i<500 ){  
 **double** x\_tmp = 0.25 \* Math.*cbrt*(4 \* Math.*sqrt*( 4\*r\*r + 9\*r + 5 ) -8\*r - 9);  
 x\_tmp += 1.0/(4 \* Math.*cbrt*(4 \* Math.*sqrt*( Math.*abs*(4\*r\*r + 9\*r + 5) ) - 8\*r - 9)) ;  
 x\_tmp += 3/4;  
  
 **double** y\_tmp = r\*(2\*x\_tmp-1);  
  
 **X**[i] = x\_tmp; *//* **Y**[i] = y\_tmp;  
  
 Mx2 += x\_tmp;  
 My2 += y\_tmp;  
 }**else if** (i >=500 && i<750){  
 **double** x\_tmp = (1/2) \* Math.*pow*(Math.*sqrt*( Math.*abs*(4\*r\*r - 8\*r + 5) ) + 2\*r - 2, 0.6666);  
 x\_tmp -= 1/(2 \* Math.*pow*(Math.*sqrt*( Math.*abs*(4\*r\*r - 8\*r + 5) ) + 2\*r - 2, 0.6666));  
 x\_tmp +=1/2;  
  
  
 **double** y\_tmp = 1 - Math.*sqrt*( Math.*abs*(r\*(-2\*x\_tmp\*x\_tmp + 2\*x\_tmp - 1) + 1) );  
  
 **X**[i] = x\_tmp;  
 **Y**[i] = y\_tmp;  
  
 Mx3 += x\_tmp;  
 My3 += y\_tmp;  
 }**else if** (i >= 750 && i < 1000){  
  
 **double** x\_tmp = 0.25\*( Math.*cbrt*(4 \* Math.*sqrt*(Math.*abs* (4\*r\*r - r) ) - 8\*r + 1 ));  
 x\_tmp += 1/(Math.*pow*(4 \* Math.*sqrt*( Math.*abs* (4\*r\*r - r) ) - 8\*r + 1, 1/3));  
  
  
  
 **double** y\_tmp = r - 2\*r\*x\_tmp;  
  
 **X**[i] = x\_tmp;  
 **Y**[i] = y\_tmp;  
  
 Mx4 += x\_tmp;  
 My4 += y\_tmp;  
 }  
 }  
  
 **Mx**[0] = Mx1/250;  
 **Mx**[1]= Mx2/250;  
 **Mx**[2] = Mx3/250;  
 **Mx**[3]= Mx4/250;  
  
 **My**[0] = My1/250;  
 **My**[1]= My2/250;  
 **My**[2] = My3/250;  
 **My**[3]= My4/250;  
  
  
  
 }  
  
 **public double** getSigmaX(){  
 **double** Sigma = 0;  
 **for** (**int** i=0; i<**X**.**length**;i++){  
 **if** (i<250){  
 Sigma += Math.*pow*(**X**[i]-**Mx**[0], 2);  
 }**else if** (i>=250 && i<500 ){  
 Sigma += Math.*pow*(**X**[i]-**Mx**[1], 2);  
 }**else if** (i >=500 && i<750){  
 Sigma += Math.*pow*(**X**[i]-**Mx**[2], 2);  
 }**else if** (i >= 750 && i < 1000){  
 Sigma += Math.*pow*(**X**[i]-**Mx**[3], 2);  
 }  
 }  
 **return** 100\*Math.*sqrt*(Sigma/**X**.**length**);  
 }  
  
 **public double** getSigmaY(){  
 **double** Sigma = 0;  
 **for** (**int** i=0; i<**Y**.**length**;i++){  
 **if** (i<250){  
 Sigma += Math.*pow*(**Y**[i]-**My**[0], 2);  
 }**else if** (i>=250 && i<500 ){  
 Sigma += Math.*pow*(**Y**[i]-**My**[1], 2);  
 }**else if** (i >=500 && i<750){  
 Sigma += Math.*pow*(**Y**[i]-**My**[2], 2);  
 }**else if** (i >= 750 && i < 1000){  
 Sigma += Math.*pow*(**Y**[i]-**My**[3], 2);  
 }  
 }  
 **return** 1.7\*Math.*sqrt*(Sigma/**Y**.**length**);  
 }  
  
  
 **public double** getCov (){  
 **double** cov = 0;  
 **for** (**int** i=0; i<**Y**.**length**;i++){  
 **if** (i<250){  
 cov += (**X**[i]-**Mx**[0])\*(**Y**[i]-**My**[0]);  
 }**else if** (i>=250 && i<500 ){  
 cov += (**X**[i]-**Mx**[1])\*(**Y**[i]-**My**[1]);  
 }**else if** (i >=500 && i<750){  
 cov += (**X**[i]-**Mx**[2])\*(**Y**[i]-**My**[2]);  
 }**else if** (i >= 750 && i < 1000){  
 cov += (**X**[i]-**Mx**[3])\*(**Y**[i]-**My**[3]);  
 }  
 }  
 **return** cov/**Y**.**length**;  
 }  
  
 **public double** getRadius(){  
 **return** -1000\*getCov()/(getSigmaX()\*getSigmaY());  
 }  
 **public double** []Mx(){  
 **return Mx** ;  
 }  
 **public double** []My(){  
 **return My** ;  
 }  
  
 **public double** [] getX(){  
 **return X**;  
 }  
  
 **public double** [] getY(){  
 **return Y**;  
 }  
  
  
 }

Результат:

Mx 3.4027382044345202

My 2.6775296998431535

SigmaX 20.730806698618828

SigmaY 0.7456608103620156

Radius -0.4156469482492327