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 №5**

По дисципліні «Теорія ймовірності»

Виконав: Перевірив:

Студент групи ІО-21 Викладач

Коноз А.О. Марковський О. П.

Дата здачі\_\_\_\_\_\_\_\_\_\_\_\_\_

Захищено з балом\_\_\_\_\_

Київ 2013

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

**public** **class** SolveSLAR {

**private** **double** slar [][];

**private** **double** solve[];

**private** **double** solution[];

/\*\*

\* Метод зчитує матрицю СЛАР з файлу,

\* попередньо зчитавши розмір її з файлу

\*/

**private** **void** reader () **throws** Exception {

BufferedReader readerForLines = **new** BufferedReader(**new** InputStreamReader(

**new** FileInputStream("c:\\lab.txt")));

String vector [];

**int** size = Integer.*parseInt*(readerForLines.readLine());

slar = **new** **double** [size][size];

solve = **new** **double**[size];

solution = **new** **double**[size];

solve[0] = 1.0;

**for** (**int** i = 0; i < size; i++) {

vector = readerForLines.readLine().split(" ");

**for** (**int** j = 0; j < vector.length; j++) {

slar[i][j] = Double.*parseDouble*(vector[j]);

}

}

readerForLines.close();

}

/\*\*

\* Приведення матриці до трикутного вигляду

\*/

**private** **void** triangle(){

**double** koef;

**int** intel = 0;

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

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

koef = slar[j][intel]/slar[i][intel];

**for** (**int** j2 = 0; j2 < slar.length; j2++) {

slar[j][j2] = -slar[i][j2]\*koef + slar[j][j2];

}

solve[j] = -solve[i]\*koef + solve[j];

}

intel++;

}

}

/\*\*

\* Метод гауса.

\*/

**public** **void** Gaus() **throws** Exception{

reader();

triangle();

**double** sum;

**for** (**int** i = slar.length - 1; i >= 0; i--) {

sum = solve[i];

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

sum = sum - slar[i][j]\*solution[j];

}

solution[i] = sum/slar[i][i];

}

System.*out*.println("Теоретичний метод:");

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

System.*out*.println("P" + (i + 1) + " = " + solution[i] );

}

}

}

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

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

**public** **class** Graph {

**private** **double** matrix [][];

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

**private** **double** time[];

**private** **double** ways[];

**private** **int** k = 1000;

**private** **void** reader () **throws** Exception {

BufferedReader readerForLines = **new** BufferedReader(**new** InputStreamReader(

**new** FileInputStream("c:\\matrix.txt")));

String vector [];

**int** size = Integer.*parseInt*(readerForLines.readLine());

matrix = **new** **double** [size][size];

time = **new** **double**[size];

ways = **new** **double**[size];

Arrays.*fill*(ways, 0);

**for** (**int** i = 0; i < size; i++) {

vector = readerForLines.readLine().split(" ");

**for** (**int** j = 0; j < vector.length; j++) {

matrix[i][j] = Double.*parseDouble*(vector[j]);

}

}

}

**private** **void** countTime(**int** i){

**double** min = 0.0;

**int** key = 0;

**while** (k > 0){

**for** (**int** j = 0; j < matrix.length; j++) {

**if**(matrix[i][j] != 0){

ways[j] = (-1)/matrix[i][j]\*Math.*log*(random.nextDouble());

}

}

min = Double.*MAX\_VALUE*;

**for** (**int** j = 0; j < ways.length; j++) {

**if** ((min > ways[j]) && (ways[j] !=0)){

min = ways[j];

key = j;

}

}

Arrays.*fill*(ways, 0);

time[i] += min;

i = key;

k--;

}

}

**private** **void** showP(){

**double** mainTime = 0.0;

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

mainTime += time[i];

}

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

System.*out*.println("P" + (i + 1) + " = " + time[i]/mainTime);

}

}

**private** **void** countP() **throws** Exception{

System.*out*.println("Практичний метод:");

reader();

countTime(0);

showP();

}

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

Graph graph = **new** Graph();

SolveSLAR solve = **new** SolveSLAR();

graph.countP();

solve.Gaus();

}

}