KRZYSZTOF SOKÓŁ-SZOŁTYSEK

PROGRAM 2

GRUPA PONIEDZIAŁKOWA

Program oblicza dekompozycję LU metodą Doolittle'a dla dowolnej macierzy 2x2, 3x3, 4x4. Uzyskane wyniki są poprawne(sprawdzanie mnożeniem wynikowych macierzy L i U) poza przypadkami, kiedy należy użyć pivotingu(program jest podatny na dzielenie przez zero)

PRZYKŁADOWY INPUT&OUTPUT (3X3 I 4X4)

```
run:
Podaj wymiar n tabeli
3

7 2 3
1 4 3
2 3 1

Macierz L
1.0 0.0 0.0
 0.14285714285714285 1.0 0.0
 0.2857142857142857 0.6538461538461539 1.0

Macierz U
7.0 2.0 3.0
 0.0 3.7142857142857144 2.5714285714285716
 0.0 0.0 -1.5384615384615388

 U*L:
 7.0 2.0 3.0
 1.0 4.0 3.0
 2.0 3.0 1.0
```

```
run:
Podaj wymiar n tabeli
4

3 4 6 2
1 3 5 3
7 4 2 4
9 6 7 5
Macierz L
 1.0 0.0 0.0 0.0
 0.3333333333333333 1.0 0.0 0.0
 2.3333333333333335 -3.2 1.0 0.0
 3.0 -3.5999999999999996 0.08333333333333383
1.0

 Macierz U
 3.0 4.0 6.0 2.0
 0.0 1.6666666666666667 3.0 2.3333333333333335
 0.0 0.0 -2.3999999999999986 6.800000000000001
 0.0 0.0 0.0 6.833333333333333

 U*L:
 3.0 4.0 6.0 2.0
 1.0 3.0 5.0 3.0
 7.0 4.0 2.0 4.0
 9.0 6.0 7.0 5.0
```

LISTING

http://pastebin.com/t1zevQN6

```
import
java.util.Sca
nner; public
class Main{
 public static void main(String
 args[])
 { System.out.println("Podaj
 wymiar n tabeli"); Scanner sc
 = new Scanner(System.in);
 int n = sc.nextInt();
 double[][] mac =
 new double[n][n];
 /*int[] piv = new
 int[n];
 for (int i =
 0; i < n;
 i++)
 { piv[i] = i;
 }
 int
piv
sig
n =
1;
*/
 for (int i = 0; i
 < n; i++) for
 (int j = 0; j <
 n; j++)
 mac[i][j] =
 sc.nextDouble
 (); if (n == 2)
 {
 /* pivot do
 dokonczenia
 for (int i =
 j+1; i < m;
 i++) {
 if (Math.abs() >
 Math.abs()) { p
 = i;
 }
 }
 if (p != j) {
 for (int k = 0; k < n; k++) {
 }
 int k = piv[p]; piv[p]
 = piv[j]; piv[j] = k;
 pivsign = -pivsign;
 }*/
 double[][] l =
 new double[n][n];
 l[0][0] = l[1][1] =
 1;
 l[0][1] = 0;
 double[][] u =
 new double[n][n];
 u[1][0] = 0;
u[0][0] = mac[0][0]; u[0][1] = mac[0][1];
 l[1][0] = mac[1][0] / mac[0][0]; u[1][1] = mac[1][1] -
 (l[1][0] * u[0][1]); System.out.println("Macierz L:");
 for (int i = 0; i < n; i++) { for (int j = 0; j < n; j++)
 System.out.print(" " + l[i][j]); System.out.println();
 }
 System.out.println("Macierz U"); for (int i = 0; i < n;
 i++) {
 for (int j = 0; j < n; j++) System.out.print(" " +
 u[i][j]); System.out.println();
 }
 UxL(u,l,2);
 }
 if (n == 3) {
 double[][] l = new double[n][n]; l[0][0] = l[1][1] =
 l[2][2] = 1; l[0][1] = l[0][2] = l[1][2] = 0; double[][] u
 = new double[n][n]; u[1][0] = u[2][0] = u[2][1] = 0;
 u[0][0] = mac[0][0];
 u[0][1] = mac[0][1]; u[0][2] = mac[0][2];
 l[1][0] = mac[1][0] / mac[0][0]; u[1][1] = mac[1][1] -
 (l[1][0] * u[0][1]); u[1][2] = mac[1][2] - (l[1][0] * u[0][2]);
 l[2][0] = mac[2][0] / u[0][0];
 l[2][1] = (mac[2][1] - l[2][0] * u[0][1]) / u[1][1];
 u[2][2] = mac[2][2] - (l[2][0] * u[0][2]) - (l[2][1] * u[1][2]);
System.out.println("Macierz L");
 for (int i = 0; i < n; i++) { for (int j = 0; j < n; j++)
 System.out.print(" " + l[i][j]); System.out.println();
 }
 System.out.println("Macierz U"); for (int i = 0; i < n;
 i++) {
 for (int j = 0; j < n; j++) System.out.print(" " +
 u[i][j]); System.out.println();
}
UxL(u,l,3);
}




if (n == 4) {
double[][] l = new double[n][n];
l[0][0] = l[1][1] = l[2][2] =
l[3][3] = 1;
l[0][1] = l[0][2] = l[0][3] = l[1][2] = l[1][3]
= l[2][3] = 0; double[][] u = new
double[n][n];
u[1][0] = u[2][0] = u[2][1] = u[3][0] = u[3][1] = u[3][2] = 0;



//ok
```

```
u[0][0]            =
mac[0][0];
u[0][1]            =
mac[0][1];
u[0][2]            =
mac[0][2];
u[0][3]            =
mac[0][3];
l[1][0]   =   mac[1][0]   /
u[0][0];      l[2][0]      =
mac[2][0]    /    u[0][0];
l[3][0]   =   mac[3][0]   /
u[0][0];
u[1][1]  =  mac[1][1]  -  (l[1][0]  *
u[0][1]);  u[1][2]  =  mac[1][2]  -
(l[1][0]   *   u[0][2]);   u[1][3]   =
mac[1][3] - (l[1][0] * u[0][3]);
l[2][1] = (mac[2][1] - l[2][0] * u[0][1]) /
u[1][1]; l[3][1] = (mac[3][1] - u[0][1] *
l[3][0])/ u[1][1];
u[2][2] = mac[2][2] - (l[2][1] * u[1][2] + l[2][0]
* u[0][2]); u[2][3] = mac[2][3] - (l[2][0]* u[0][3]
+ l[2][1] * u[1][3]);
l[3][2] = (mac[3][2] - (u[1][2] * l[3][1] + u[0][2] *
l[3][0]))/u[2][2];
u[3][3] = mac[3][3] - (u[2][3] * l[3][2] + u[1][3] * l[3][1] +
u[0][3] * l[3][0]); System.out.println("Macierz L");
for (int i = 0; i < n; i++)
{ for (int j = 0; j < n;
j++) System.out.print("
" + l[i][j]);
System.out.println();
}
System.out.println("Macierz
U"); for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
System.out.print(" " +
u[i][j]);
System.out.println();
}
UxL(u,l,4);
```

```
}









}
public static void
UxL(double[][]u,double[][]l, int n ) {
double[][] UL = new
double[n][n]; for (int i = 0; i
< n; i++) {
for (int j = 0; j < n;
j++) { double temp
= 0;
for (int w = 0; w < n;
w++) { temp += l[i][w]
* u[w][j];
}
UL[i][j] = temp;
}
}
System.out.println("U*L:
"); for (int i = 0; i < n;
i++) { for (int j = 0; j <
n; j++)
System.out.print(" " +
UL[i][j]);
System.out.println();
}
}
}
```