**Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки**

**Лабораторна робота №2**

з дисципліни  
«Дискретні структури»

на тему «Характеристики графів»

Виконав: Перевірила:

студент групи ІП-93 Сергієнко А. А.  
Домінський Валентин Олексійович   
номер залікової книжки: 9311

Київ 2020

**Завдання на лабораторну роботу:**

1. Представити напрямлений граф із заданими параметрами так само, як у

лабораторній роботі №1.

Відміна: матриця А напрямленого графа за варіантом формується за

командами Scilab:

rand("seed", п1п2п3п4);

T = rand(n,n) + rand(n,n);

A = floor((1.0 – п3\*0.01 – п4\*0.01 - 0.3)\*T)

Перетворити граф у ненапрямлений.

2. Визначити степені вершин напрямленого і ненапрямленого графів.

Програма на екран виводить степені усіх вершин ненапрямленого графу і

напівстепені виходу та заходу напрямленого графу. Визначити, чи граф є

однорідним та якщо так, то вказати степінь однорідності графу.

3. Визначити всі висячі та ізольовані вершини. Програма на екран

виводить перелік усіх висячих та ізольованих вершин графу.

Варіант 9311:  
*п =* 11  
розміщення вершин: колом

**Текст програми для Scilab**

**clf; //clearing canvas**

**TypeOfGraph = 1; //0 is for non-oriented,1 for oriented**

**N1 = 9; //numbers of My Zalikovka to make figure**

**N2 = 3;**

**N3 = 1;**

**N4 = 1;**

**N = 10 + N3; //number of vertexes**

**//preparing for making canvas**

**//making values for plot2d**

**//x on left is 0**

**ForPlot\_X1 = 0;**

**//x on right is 70**

**ForPlot\_X2 = 70;**

**//y on down is 0**

**ForPlot\_Y1 = 0;**

**//y on upper is 70**

**ForPlot\_Y2 = 70;**

**RadiusOfCircle = 5; //Woah,its radius of circle**

**DiameterOfCircle = 2\*RadiusOfCircle; //Woah,its diameter of circle**

**ColorOfArrow\_1 = 14; //color of arrow\_1 //1 is good**

**ColorOfArrow\_2 = 9; //color of arrow\_2 //3,**

**N1N2N3N4 = N1\*1000 + N2\*100 + N3\*10 + N4**

**//generating matrix for or-graph(from the konspekt)**

**rand("seed", N1N2N3N4);**

**T = rand(N,N) + rand(N,N);**

**A\_oriented = floor((1.0 - N3\*0.01 - N4\*0.01 - 0.3)\*T)**

**//generating matrix for non\_or-graph**

**At=A\_oriented'; //transpose matrix**

**A\_non\_oriented=At+A\_oriented //final non\_oriented matrix(adding regular matrix and transpose)**

**//loop,where we changing 2,which we get by adding**

**//two matrixes,for every element,that = 2**

**for v=1:1:N**

**for v1=1:1:N**

**if A\_non\_oriented(v,v1) == 2**

**A\_non\_oriented(v,v1)=1**

**end**

**end**

**end**

**//choosing matrix and width of arrows**

**if TypeOfGraph == 0**

**Matrix = A\_non\_oriented;**

**WidthOfArrow = 0;**

**elseif TypeOfGraph == 1**

**Matrix = A\_oriented;**

**WidthOfArrow = 30;**

**end**

**//making canvas with our varibles**

**//plot2d([0,200],[0,100],[-1,-1],"022") — формує**

**//прямокутне поле у графічному вікні розміром 100х200 одиниць.**

**plot2d([ForPlot\_X1;ForPlot\_X2],[ForPlot\_Y1;ForPlot\_Y2],0);**

**//looking for a center**

**//(100-0)/2=50**

**X\_O = (ForPlot\_X2-ForPlot\_X1)/2;**

**Y\_O = (ForPlot\_Y2-ForPlot\_Y1)/2;**

**//radius of the big circle,on which we will place our vertexes**

**BigRadius = 2/5\*sqrt((ForPlot\_X2-ForPlot\_X1)\*(ForPlot\_Y2-ForPlot\_Y1))**

**for x=1:1:N//making operations for each element**

**X\_Position = cos(2 \* %pi \* x / N) \* BigRadius + X\_O;//calculating coordinates**

**Y\_Position = sin(2 \* %pi \* x / N) \* BigRadius + Y\_O;**

**//drawing vertexes**

**// xarc(x,y,w,h,a1,a2) — малювання дуги еліпса, який вписаний у прямокутник**

**//з координатами (х,у) верхнього лівого кута і шириною w та висотою h,**

**//яка обмежена променями з кутами а1 і а2, які задаються у 64-х частках градуса.**

**//Наприклад, коло діаметром 4 одиниць з центром у точці (10,20) задається**

**//функцією xarc(8,22,4,4,0,360\*64);**

**xarc(X\_Position,Y\_Position,DiameterOfCircle,DiameterOfCircle,0,360\*64);**

**// xnumb(x,y,data) — вивід числа data починаючи з позиції з координатами**

**//(х,у).**

**xnumb(X\_Position+RadiusOfCircle,Y\_Position-RadiusOfCircle,x);**

**end**

**//need 11 variables for non\_oriented and 22(11 for enter and 11 for exit) for oriented graph**

**Vertex\_1\_Exit=0;**

**Vertex\_2\_Exit=0;**

**Vertex\_3\_Exit=0;**

**Vertex\_4\_Exit=0;**

**Vertex\_5\_Exit=0;**

**Vertex\_6\_Exit=0;**

**Vertex\_7\_Exit=0;**

**Vertex\_8\_Exit=0;**

**Vertex\_9\_Exit=0;**

**Vertex\_10\_Exit=0;**

**Vertex\_11\_Exit=0;**

**Vertex\_1\_Enter=0;**

**Vertex\_2\_Enter=0;**

**Vertex\_3\_Enter=0;**

**Vertex\_4\_Enter=0;**

**Vertex\_5\_Enter=0;**

**Vertex\_6\_Enter=0;**

**Vertex\_7\_Enter=0;**

**Vertex\_8\_Enter=0;**

**Vertex\_9\_Enter=0;**

**Vertex\_10\_Enter=0;**

**Vertex\_11\_Enter=0;**

**Vertex\_1\_Non\_oriented=0;**

**Vertex\_2\_Non\_oriented=0;**

**Vertex\_3\_Non\_oriented=0;**

**Vertex\_4\_Non\_oriented=0;**

**Vertex\_5\_Non\_oriented=0;**

**Vertex\_6\_Non\_oriented=0;**

**Vertex\_7\_Non\_oriented=0;**

**Vertex\_8\_Non\_oriented=0;**

**Vertex\_9\_Non\_oriented=0;**

**Vertex\_10\_Non\_oriented=0;**

**Vertex\_11\_Non\_oriented=0;**

**for i=1:1:N**

**for j=1:1:N**

**if Matrix(i,j) == 1;**

**Position\_X\_I = cos(2 \* %pi \* i / N) \* (BigRadius-RadiusOfCircle) + X\_O;**

**Position\_Y\_I = sin(2 \* %pi \* i / N) \* (BigRadius-RadiusOfCircle) + Y\_O;**

**Position\_X\_J = cos(2 \* %pi \* j / N) \* (BigRadius-RadiusOfCircle) + X\_O;**

**Position\_Y\_J = sin(2 \* %pi \* j / N) \* (BigRadius-RadiusOfCircle) + Y\_O;**

**i1=i;**

**select i1**

**case 1**

**Vertex\_1\_Exit=Vertex\_1\_Exit+1;**

**case 2**

**Vertex\_2\_Exit=Vertex\_2\_Exit+1;**

**case 3**

**Vertex\_3\_Exit=Vertex\_3\_Exit+1;**

**case 4**

**Vertex\_4\_Exit=Vertex\_4\_Exit+1;**

**case 5**

**Vertex\_5\_Exit=Vertex\_5\_Exit+1;**

**case 6**

**Vertex\_6\_Exit=Vertex\_6\_Exit+1;**

**case 7**

**Vertex\_7\_Exit=Vertex\_7\_Exit+1;**

**case 8**

**Vertex\_8\_Exit=Vertex\_8\_Exit+1;**

**case 9**

**Vertex\_9\_Exit=Vertex\_9\_Exit+1;**

**case 10**

**Vertex\_10\_Exit=Vertex\_10\_Exit+1;**

**case 11**

**Vertex\_11\_Exit=Vertex\_11\_Exit+1;**

**end**

**j1=j;**

**select j1**

**case 1**

**Vertex\_1\_Enter=Vertex\_1\_Enter+1;**

**case 2**

**Vertex\_2\_Enter=Vertex\_2\_Enter+1;**

**case 3**

**Vertex\_3\_Enter=Vertex\_3\_Enter+1;**

**case 4**

**Vertex\_4\_Enter=Vertex\_4\_Enter+1;**

**case 5**

**Vertex\_5\_Enter=Vertex\_5\_Enter+1;**

**case 6**

**Vertex\_6\_Enter=Vertex\_6\_Enter+1;**

**case 7**

**Vertex\_7\_Enter=Vertex\_7\_Enter+1;**

**case 8**

**Vertex\_8\_Enter=Vertex\_8\_Enter+1;**

**case 9**

**Vertex\_9\_Enter=Vertex\_9\_Enter+1;**

**case 10**

**Vertex\_10\_Enter=Vertex\_10\_Enter+1;**

**case 11**

**Vertex\_11\_Enter=Vertex\_11\_Enter+1;**

**end**

**if i == j**

**A1=i;**

**A2=j;**

**Position\_X\_I = cos(2 \* %pi \* i / N) \* (BigRadius) + X\_O;**

**Position\_Y\_I = sin(2 \* %pi \* i / N) \* (BigRadius) + Y\_O;**

**Position\_X\_J = cos(2 \* %pi \* j / N) \* (BigRadius) + X\_O;**

**Position\_Y\_J = sin(2 \* %pi \* j / N) \* (BigRadius) + Y\_O;**

**xarc(Position\_X\_I-RadiusOfCircle,Position\_Y\_I+RadiusOfCircle\*3/4,3/2\*DiameterOfCircle,3/2\*DiameterOfCircle,0,190\*64);**

**//xarrows([x1;x2], [y1;y2], w, c) — стрілка з координатами початку**

**//(x1;y1), кінця (x2;y2), розмірами вістря w і кольором с (с=1 –чорний, 2– синій і**

**//т.д.).**

**xarrows([Position\_X\_I-RadiusOfCircle; Position\_X\_J],[Position\_Y\_I-RadiusOfCircle; Position\_Y\_J-RadiusOfCircle], WidthOfArrow, 1);**

**else**

**if Matrix(i,j)==Matrix(j,i) && i>j**

**if TypeOfGraph == 1**

**CentrePosition\_X = (Position\_X\_I+RadiusOfCircle+Position\_X\_J+RadiusOfCircle)/2;**

**CentrePosition\_Y = (Position\_Y\_I-RadiusOfCircle+Position\_Y\_J)/2;**

**// xsegs([x1;y1], [x2;y2], c) — пряма з координатами початку [x1;y1],**

**//кінця [x2;y2] і кольором с.**

**if (CentrePosition\_X<X\_O && CentrePosition\_Y<Y\_O)**

**xsegs([Position\_X\_I+RadiusOfCircle; CentrePosition\_X+RadiusOfCircle],[Position\_Y\_I-RadiusOfCircle; CentrePosition\_Y+DiameterOfCircle],ColorOfArrow\_2);**

**xarrows([CentrePosition\_X+RadiusOfCircle; Position\_X\_J+RadiusOfCircle],[CentrePosition\_Y+DiameterOfCircle; Position\_Y\_J-RadiusOfCircle], WidthOfArrow, ColorOfArrow\_2);**

**elseif (CentrePosition\_X>X\_O && CentrePosition\_Y<Y\_O)**

**xsegs([Position\_X\_I+RadiusOfCircle; CentrePosition\_X+RadiusOfCircle],[Position\_Y\_I-RadiusOfCircle; CentrePosition\_Y+DiameterOfCircle],ColorOfArrow\_2);**

**xarrows([CentrePosition\_X+RadiusOfCircle; Position\_X\_J+RadiusOfCircle],[CentrePosition\_Y+DiameterOfCircle; Position\_Y\_J-RadiusOfCircle], WidthOfArrow, ColorOfArrow\_2);**

**elseif (CentrePosition\_X>X\_O && CentrePosition\_Y>Y\_O)**

**xsegs([Position\_X\_I+RadiusOfCircle; CentrePosition\_X-RadiusOfCircle],[Position\_Y\_I-RadiusOfCircle; CentrePosition\_Y-DiameterOfCircle],ColorOfArrow\_2);**

**xarrows([CentrePosition\_X-RadiusOfCircle; Position\_X\_J+RadiusOfCircle],[CentrePosition\_Y-DiameterOfCircle; Position\_Y\_J-RadiusOfCircle], WidthOfArrow, ColorOfArrow\_2);**

**elseif (CentrePosition\_X<X\_O && CentrePosition\_Y>Y\_O)**

**xsegs([Position\_X\_I+RadiusOfCircle; CentrePosition\_X+RadiusOfCircle],[Position\_Y\_I-RadiusOfCircle; CentrePosition\_Y-DiameterOfCircle],ColorOfArrow\_2);**

**xarrows([CentrePosition\_X+RadiusOfCircle; Position\_X\_J+RadiusOfCircle],[CentrePosition\_Y-DiameterOfCircle; Position\_Y\_J-RadiusOfCircle], WidthOfArrow, ColorOfArrow\_2);**

**end**

**else**

**xarrows([Position\_X\_I+RadiusOfCircle; Position\_X\_J+RadiusOfCircle],[Position\_Y\_I-RadiusOfCircle; Position\_Y\_J-RadiusOfCircle], WidthOfArrow, ColorOfArrow\_1);**

**end**

**elseif TypeOfGraph == 0**

**else**

**xarrows([Position\_X\_I+RadiusOfCircle; Position\_X\_J+RadiusOfCircle],[Position\_Y\_I-RadiusOfCircle; Position\_Y\_J-RadiusOfCircle], WidthOfArrow, ColorOfArrow\_1);**

**end**

**end**

**end**

**end**

**end**

**if TypeOfGraph == 1**

**Vertex\_1\_Non\_oriented=Vertex\_1\_Exit+Vertex\_1\_Enter**

**Vertex\_2\_Non\_oriented=Vertex\_2\_Exit+Vertex\_2\_Enter**

**Vertex\_3\_Non\_oriented=Vertex\_3\_Exit+Vertex\_3\_Enter**

**Vertex\_4\_Non\_oriented=Vertex\_4\_Exit+Vertex\_4\_Enter**

**Vertex\_5\_Non\_oriented=Vertex\_5\_Exit+Vertex\_5\_Enter**

**Vertex\_6\_Non\_oriented=Vertex\_6\_Exit+Vertex\_6\_Enter**

**Vertex\_7\_Non\_oriented=Vertex\_7\_Exit+Vertex\_7\_Enter**

**Vertex\_8\_Non\_oriented=Vertex\_8\_Exit+Vertex\_8\_Enter**

**Vertex\_9\_Non\_oriented=Vertex\_9\_Exit+Vertex\_9\_Enter**

**Vertex\_10\_Non\_oriented=Vertex\_10\_Exit+Vertex\_10\_Enter**

**Vertex\_11\_Non\_oriented=Vertex\_11\_Exit+Vertex\_11\_Enter**

**elseif TypeOfGraph == 0**

**Vertex\_1\_Non\_oriented=Vertex\_1\_Exit**

**Vertex\_2\_Non\_oriented=Vertex\_2\_Exit**

**Vertex\_3\_Non\_oriented=Vertex\_3\_Exit**

**Vertex\_4\_Non\_oriented=Vertex\_4\_Exit**

**Vertex\_5\_Non\_oriented=Vertex\_5\_Exit**

**Vertex\_6\_Non\_oriented=Vertex\_6\_Exit**

**Vertex\_7\_Non\_oriented=Vertex\_7\_Exit**

**Vertex\_8\_Non\_oriented=Vertex\_8\_Exit**

**Vertex\_9\_Non\_oriented=Vertex\_9\_Exit**

**Vertex\_10\_Non\_oriented=Vertex\_10\_Exit**

**Vertex\_11\_Non\_oriented=Vertex\_11\_Exit**

**end**

**//making matrixes of variables to make it easier**

**//Power of vertexes**

**//Exit Enter**

**MatrixOfOrVertexes = [Vertex\_1\_Exit Vertex\_1\_Enter;Vertex\_2\_Exit Vertex\_2\_Enter;Vertex\_3\_Exit Vertex\_3\_Enter;Vertex\_4\_Exit Vertex\_4\_Enter;Vertex\_5\_Exit Vertex\_5\_Enter;Vertex\_6\_Exit Vertex\_6\_Enter;Vertex\_7\_Exit Vertex\_7\_Enter;Vertex\_8\_Exit Vertex\_8\_Enter;Vertex\_9\_Exit Vertex\_9\_Enter;Vertex\_10\_Exit Vertex\_10\_Enter;Vertex\_11\_Exit Vertex\_11\_Enter];**

**//Sum**

**MatrixOfNonOrVertexes = [Vertex\_1\_Non\_oriented;Vertex\_2\_Non\_oriented;Vertex\_3\_Non\_oriented;Vertex\_4\_Non\_oriented;Vertex\_5\_Non\_oriented;Vertex\_6\_Non\_oriented;Vertex\_7\_Non\_oriented;Vertex\_8\_Non\_oriented;Vertex\_9\_Non\_oriented;Vertex\_10\_Non\_oriented;Vertex\_11\_Non\_oriented];**

**//Exit Enter Sum**

**//not using right now**

**MatrixOfEveryVertexes = cat(2,MatrixOfOrVertexes,MatrixOfNonOrVertexes);**

**//printing variables**

**if TypeOfGraph == 1**

**disp("Exit Enter")**

**MatrixOfOrVertexes**

**elseif TypeOfGraph == 0**

**disp("Sum")**

**MatrixOfNonOrVertexes///2 and delete Vertex\_1\_Non\_oriented=Vertex\_1\_Exit ^**

**end**

**//checking if graph is homogeneous**

**if Vertex\_1\_Non\_oriented==Vertex\_2\_Non\_oriented && Vertex\_2\_Non\_oriented==Vertex\_3\_Non\_oriented && Vertex\_3\_Non\_oriented==Vertex\_4\_Non\_oriented && Vertex\_4\_Non\_oriented==Vertex\_5\_Non\_oriented && Vertex\_5\_Non\_oriented==Vertex\_6\_Non\_oriented && Vertex\_6\_Non\_oriented==Vertex\_7\_Non\_oriented && Vertex\_7\_Non\_oriented==Vertex\_8\_Non\_oriented && Vertex\_8\_Non\_oriented==Vertex\_9\_Non\_oriented && Vertex\_9\_Non\_oriented==Vertex\_10\_Non\_oriented && Vertex\_10\_Non\_oriented==Vertex\_11\_Non\_oriented// && Vertex\_11\_Non\_oriented==Vertex\_1\_Non\_oriented**

**homogeneousGraph=1;**

**disp('The graph is homogeneous and its power = ', Vertex\_1\_Non\_oriented)**

**else**

**homogeneousGraph=0;**

**disp('The graph is not homogeneous')**

**end**

**////identifying if vertexes isolated or floppy**

**for x = 1:1:N**

**if MatrixOfNonOrVertexes(x)==1**

**disp('Vertex',x,'is floppy')**

**elseif MatrixOfNonOrVertexes(x)==0**

**disp('Vertex',x,'is isolated')**

**end**

**end**

**Згенеровані матриці суміжності**

Орграф:

0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0.

0. 0. 0. 1. 0. 0. 0. 1. 0. 1. 0.

0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0.

0. 0. 0. 0. 0. 1. 0. 1. 0. 1. 0.

0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

1. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

Неорграф:

0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 0.

0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0.

1. 1. 0. 0. 1. 0. 0. 0. 1. 1. 0.

0. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0.

0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0.

0. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0.

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

0. 1. 0. 1. 0. 0. 0. 0. 0. 1. 0.

1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.

1. 1. 1. 1. 0. 1. 0. 1. 0. 0. 0.

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

**Таблиця степенів вузлів:**

3.

4.

5.

4.

2.

3.

0.

3.

2.

6.

0.

**Таблиця напівстепенів вузлів:**

Вихід та вхід

2. 1.

3. 1.

2. 3.

3. 1.

1. 1.

1. 2.

0. 0.

1. 2.

0. 2.

3. 3.

0. 0.

**Ізольовані вершини:**

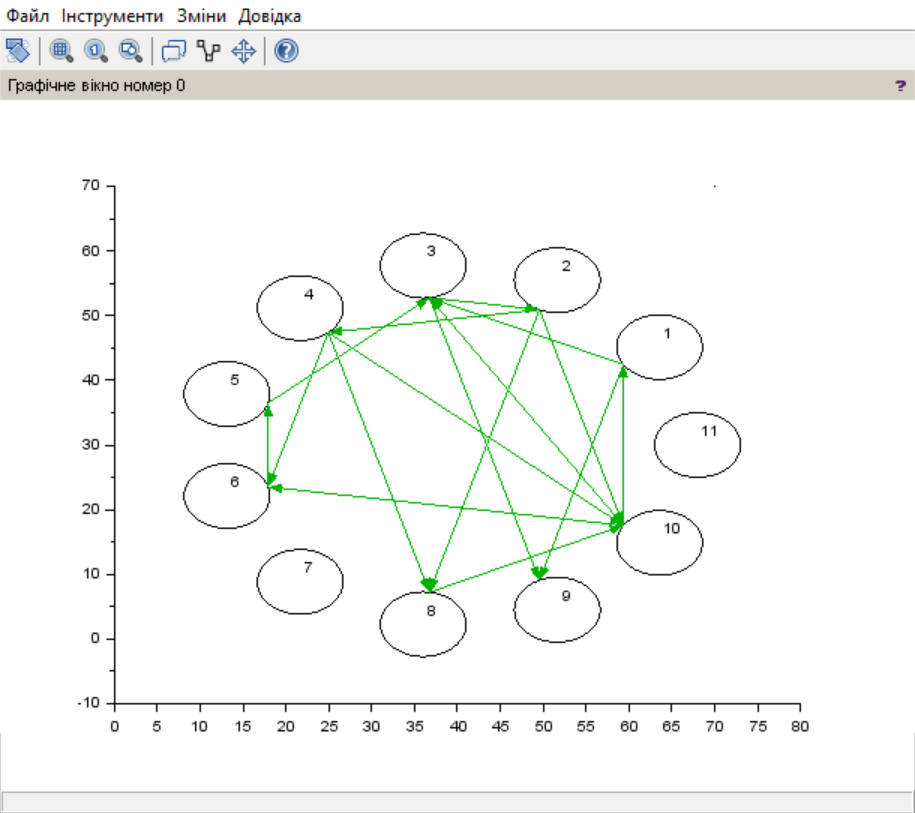
7-ма та 11-та вершини

**Висячі вершини:**

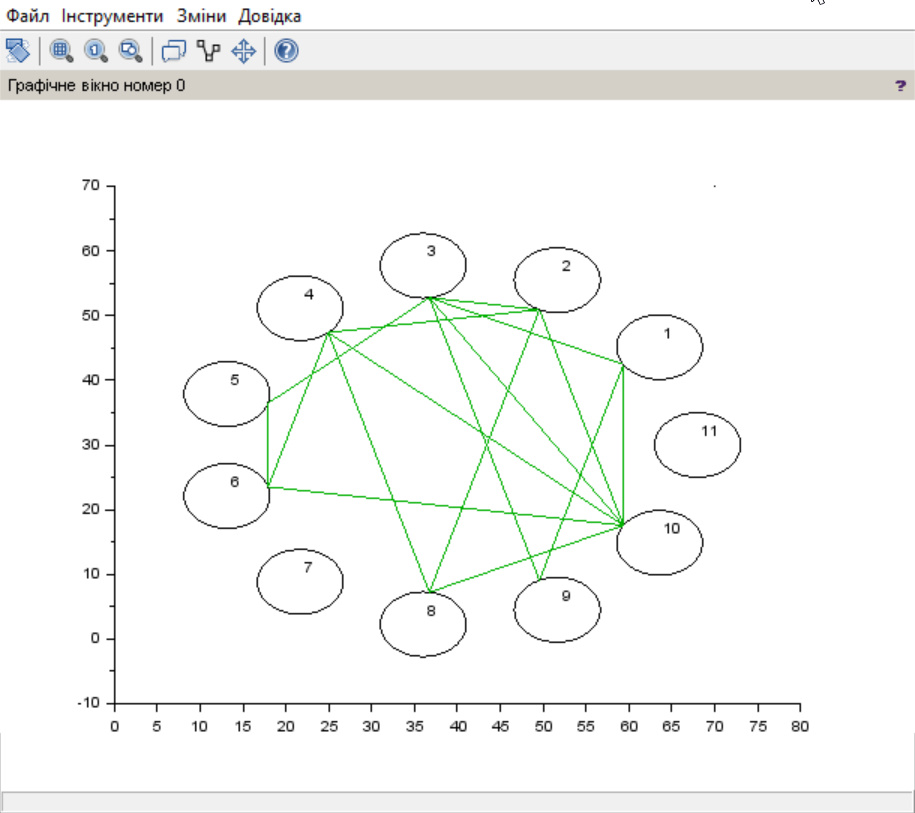
Немає

**Результати виконання програми:**

Орграф:



Неорграф:



**Висновки:**

Я навчився досліджувати характеристики графів та визначати їх на конкретних прикладах, шукати висячі та ізольовані вершини,шукати таблиці степенів та напівстепенів вузлів та визначати,чи є граф однорідним.