Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

# Лабораторная работа " Графы."

Выполнил студент гр. РИС-24-3б

Жиряков Леонид Антонович

Проверил:

Доцент кафедры ИТАС   
Ольга Андреевна Полякова

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г. Пермь, 2024

Вариант 8.

Постановка задачи:

Общая:

Реализовать алгоритмы для графа, имеющего не менее 6 вершин.

Алгоритмы:

1. Обход в ширину.

2. Обход в глубину.

3. Алгоритм Дейкстры.

4. Алгоритм Флойда.

Требования:

1. Визуализация графа с использованием доступной графической библиотеки SFML.

2. Реализованные алгоритмы должны справляться с любым графом.

3. Необходимо реализовать функции для редактирования графа:

- Создание новой вершины.

- Удаление вершины.

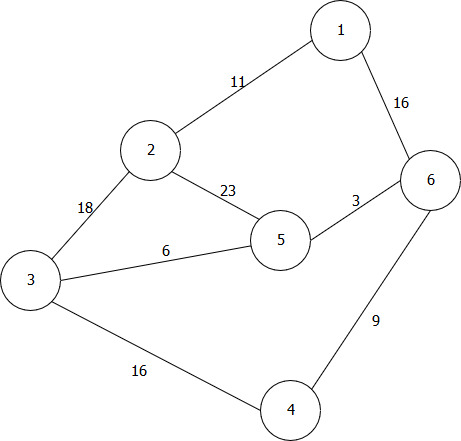
- Добавление и удаление ребра.

- Редактирование весов ребер.

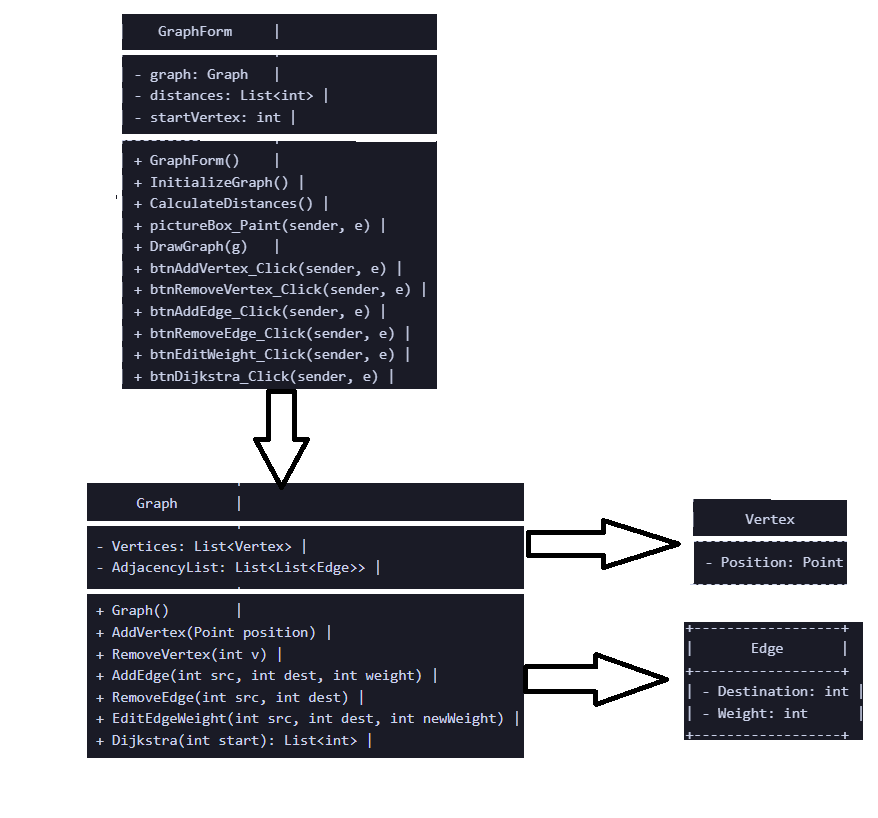
- Редактирование матрицы смежности (или инцидентности в зависимости от реализации).

Персональная:

Реализовать граф, а также алгоритм Дейкстры, выполнив все необходимые действия. Выполнение начать с вершины 4.



UML – диаграмма:



Программная реализация:

Program.cs:

using System;

using System.Windows.Forms;

namespace GraphVisualization

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new GraphForm());

}

}

}

GraphForm.cs:

using System;

using System.Drawing;

using System.Windows.Forms;

namespace GraphVisualization

{

public partial class GraphForm : Form

{

private Graph graph;

public GraphForm()

{

InitializeComponent();

graph = new Graph();

graph.InitializeGraph(); // Инициализация графа с набором данных

}

private void pictureBox\_Paint(object sender, PaintEventArgs e)

{

var g = e.Graphics;

g.Clear(Color.White);

// Рисуем ребра

for (int i = 0; i < graph.Vertices.Count; i++)

{

foreach (var edge in graph.AdjacencyList[i])

{

if (i < edge.Destination) // чтобы не рисовать ребро дважды

{

var p1 = graph.Vertices[i].Position;

var p2 = graph.Vertices[edge.Destination].Position;

g.DrawLine(Pens.Black, p1, p2);

// Рисуем вес ребра в середине

var midX = (p1.X + p2.X) / 2;

var midY = (p1.Y + p2.Y) / 2;

g.DrawString(edge.Weight.ToString(), Font, Brushes.Red, midX, midY);

}

}

}

// Рисуем вершины

for (int i = 0; i < graph.Vertices.Count; i++)

{

var p = graph.Vertices[i].Position;

var rect = new Rectangle(p.X - 15, p.Y - 15, 30, 30);

g.FillEllipse(Brushes.LightBlue, rect);

g.DrawEllipse(Pens.Black, rect);

g.DrawString(i.ToString(), Font, Brushes.Black, p.X - 7, p.Y - 10);

}

}

private void btnAddVertex\_Click(object sender, EventArgs e)

{

if (int.TryParse(txtX.Text, out int x) && int.TryParse(txtY.Text, out int y))

{

graph.AddVertex(new Point(x, y));

pictureBox.Invalidate();

}

else

{

MessageBox.Show("Введите корректные координаты X и Y.");

}

}

private void btnRemoveVertex\_Click(object sender, EventArgs e)

{

if (int.TryParse(txtVertex.Text, out int vertex))

{

graph.RemoveVertex(vertex);

pictureBox.Invalidate();

}

else

{

MessageBox.Show("Введите корректный номер вершины.");

}

}

private void btnAddEdge\_Click(object sender, EventArgs e)

{

if (int.TryParse(txtSrc.Text, out int src) &&

int.TryParse(txtDest.Text, out int dest) &&

int.TryParse(txtWeight.Text, out int weight))

{

graph.AddEdge(src, dest, weight);

pictureBox.Invalidate();

}

else

{

MessageBox.Show("Введите корректные значения для ребра.");

}

}

private void btnRemoveEdge\_Click(object sender, EventArgs e)

{

if (int.TryParse(txtSrc.Text, out int src) &&

int.TryParse(txtDest.Text, out int dest))

{

graph.RemoveEdge(src, dest);

pictureBox.Invalidate();

}

else

{

MessageBox.Show("Введите корректные значения для ребра.");

}

}

private void btnEditWeight\_Click(object sender, EventArgs e)

{

if (int.TryParse(txtSrc.Text, out int src) &&

int.TryParse(txtDest.Text, out int dest) &&

int.TryParse(txtWeight.Text, out int newWeight))

{

graph.EditEdgeWeight(src, dest, newWeight);

pictureBox.Invalidate();

}

else

{

MessageBox.Show("Введите корректные значения для изменения веса.");

}

}

private void btnDijkstra\_Click(object sender, EventArgs e)

{

if (int.TryParse(txtStartVertex.Text, out int startVertex))

{

if (startVertex < 0 || startVertex >= graph.Vertices.Count)

{

MessageBox.Show("Стартовая вершина вне диапазона.");

return;

}

var distances = graph.Dijkstra(startVertex);

string result = $"Расстояния от вершины {startVertex}:\n";

for (int i = 0; i < distances.Count; i++)

{

result += $"До вершины {i}: {(distances[i] == int.MaxValue ? "INF" : distances[i].ToString())}\n";

}

MessageBox.Show(result, "Результаты алгоритма Дейкстры");

}

else

{

MessageBox.Show("Введите корректный номер стартовой вершины.");

}

}

private void btnFloydWarshall\_Click(object sender, EventArgs e)

{

int[,] distances = graph.FloydWarshall();

int V = graph.Vertices.Count;

string result = "Матрица расстояний между всеми парами вершин:\n\n";

for (int i = 0; i < V; i++)

{

for (int j = 0; j < V; j++)

{

if (distances[i, j] == int.MaxValue)

result += "INF".PadLeft(5);

else

result += distances[i, j].ToString().PadLeft(5);

}

result += "\n";

}

MessageBox.Show(result, "Результаты алгоритма Флойда-Уоршелла");

}

}

}

Graph.cs:

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

namespace GraphVisualization

{

public class Vertex

{

public Point Position { get; set; }

}

public class Edge

{

public int Destination { get; set; }

public int Weight { get; set; }

}

public class Graph

{

public List<Vertex> Vertices { get; private set; }

public List<List<Edge>> AdjacencyList { get; private set; }

public Graph()

{

Vertices = new List<Vertex>();

AdjacencyList = new List<List<Edge>>();

}

public void AddVertex(Point position)

{

Vertices.Add(new Vertex { Position = position });

AdjacencyList.Add(new List<Edge>());

}

public void RemoveVertex(int v)

{

if (v < 0 || v >= Vertices.Count) return;

Vertices.RemoveAt(v);

AdjacencyList.RemoveAt(v);

for (int i = 0; i < AdjacencyList.Count; i++)

{

AdjacencyList[i] = AdjacencyList[i]

.Where(edge => edge.Destination != v)

.Select(edge => new Edge

{

Destination = edge.Destination > v ? edge.Destination - 1 : edge.Destination,

Weight = edge.Weight

})

.ToList();

}

}

public void AddEdge(int src, int dest, int weight)

{

if (src < 0 || src >= Vertices.Count || dest < 0 || dest >= Vertices.Count) return;

bool updated = false;

foreach (var edge in AdjacencyList[src])

{

if (edge.Destination == dest)

{

edge.Weight = weight;

updated = true;

break;

}

}

foreach (var edge in AdjacencyList[dest])

{

if (edge.Destination == src)

{

edge.Weight = weight;

updated = true;

break;

}

}

if (!updated)

{

AdjacencyList[src].Add(new Edge { Destination = dest, Weight = weight });

AdjacencyList[dest].Add(new Edge { Destination = src, Weight = weight });

}

}

public void RemoveEdge(int src, int dest)

{

if (src < 0 || src >= Vertices.Count || dest < 0 || dest >= Vertices.Count) return;

AdjacencyList[src] = AdjacencyList[src].Where(edge => edge.Destination != dest).ToList();

AdjacencyList[dest] = AdjacencyList[dest].Where(edge => edge.Destination != src).ToList();

}

public void EditEdgeWeight(int src, int dest, int newWeight)

{

if (src < 0 || src >= Vertices.Count || dest < 0 || dest >= Vertices.Count) return;

foreach (var edge in AdjacencyList[src])

if (edge.Destination == dest)

edge.Weight = newWeight;

foreach (var edge in AdjacencyList[dest])

if (edge.Destination == src)

edge.Weight = newWeight;

}

public List<int> Dijkstra(int start)

{

int V = Vertices.Count;

var dist = Enumerable.Repeat(int.MaxValue, V).ToList();

dist[start] = 0;

var pq = new SortedSet<(int dist, int vertex)>();

pq.Add((0, start));

while (pq.Count > 0)

{

var (d, u) = pq.Min;

pq.Remove((d, u));

if (d > dist[u]) continue;

foreach (var edge in AdjacencyList[u])

{

int v = edge.Destination;

int w = edge.Weight;

if (dist[u] != int.MaxValue && dist[u] + w < dist[v])

{

pq.Remove((dist[v], v));

dist[v] = dist[u] + w;

pq.Add((dist[v], v));

}

}

}

return dist;

}

public int[,] FloydWarshall()

{

int V = Vertices.Count;

int[,] dist = new int[V, V];

// Инициализация расстояний

for (int i = 0; i < V; i++)

{

for (int j = 0; j < V; j++)

{

if (i == j)

dist[i, j] = 0;

else

dist[i, j] = int.MaxValue;

}

}

// Заполнение начальных расстояний из списка смежности

for (int u = 0; u < V; u++)

{

foreach (var edge in AdjacencyList[u])

{

dist[u, edge.Destination] = edge.Weight;

}

}

// Алгоритм Флойда-Уоршелла

for (int k = 0; k < V; k++)

{

for (int i = 0; i < V; i++)

{

for (int j = 0; j < V; j++)

{

if (dist[i, k] != int.MaxValue && dist[k, j] != int.MaxValue)

{

dist[i, j] = Math.Min(dist[i, j], dist[i, k] + dist[k, j]);

}

}

}

}

return dist;

}

// Добавленный метод инициализации графа с набором данных

public void InitializeGraph()

{

// Добавление вершин (координаты подобраны для удобного отображения)

AddVertex(new Point(100, 100)); // 0

AddVertex(new Point(200, 50)); // 1

AddVertex(new Point(100, 200)); // 2

AddVertex(new Point(300, 150)); // 3

AddVertex(new Point(250, 100)); // 4

AddVertex(new Point(300, 200)); // 5

// Добавление рёбер с весами согласно изображению

AddEdge(0, 1, 11); // 1-2

AddEdge(1, 2, 23); // 2-5

AddEdge(1, 4, 16); // 2-6

AddEdge(2, 3, 18); // 3-2

AddEdge(3, 4, 16); // 4-6

AddEdge(3, 5, 6); // 3-5

AddEdge(4, 5, 3); // 6-5

AddEdge(5, 1, 9); // 5-4 (переобозначение, см. ниже)

}

}

}

GraphForm.Designer.cs:

namespace GraphVisualization

{

partial class GraphForm

{

private System.ComponentModel.IContainer components = null;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

private void InitializeComponent()

{

pictureBox = new PictureBox();

btnAddVertex = new Button();

btnRemoveVertex = new Button();

btnAddEdge = new Button();

btnRemoveEdge = new Button();

btnEditWeight = new Button();

btnDijkstra = new Button();

btnFloydWarshall = new Button(); // Добавлено

txtX = new TextBox();

txtY = new TextBox();

txtVertex = new TextBox();

txtSrc = new TextBox();

txtDest = new TextBox();

txtWeight = new TextBox();

txtStartVertex = new TextBox();

label1 = new Label();

label2 = new Label();

label3 = new Label();

label4 = new Label();

label5 = new Label();

label6 = new Label();

label7 = new Label();

((System.ComponentModel.ISupportInitialize)pictureBox).BeginInit();

SuspendLayout();

//

// pictureBox

//

pictureBox.BackColor = Color.White;

pictureBox.Location = new Point(16, 18);

pictureBox.Margin = new Padding(4, 5, 4, 5);

pictureBox.Name = "pictureBox";

pictureBox.Size = new Size(1067, 615);

pictureBox.TabIndex = 0;

pictureBox.TabStop = false;

pictureBox.Paint += pictureBox\_Paint;

//

// btnAddVertex

//

btnAddVertex.Location = new Point(16, 643);

btnAddVertex.Margin = new Padding(4, 5, 4, 5);

btnAddVertex.Name = "btnAddVertex";

btnAddVertex.Size = new Size(133, 35);

btnAddVertex.TabIndex = 1;

btnAddVertex.Text = "Добавить вершину";

btnAddVertex.UseVisualStyleBackColor = true;

btnAddVertex.Click += btnAddVertex\_Click;

//

// btnRemoveVertex

//

btnRemoveVertex.Location = new Point(157, 643);

btnRemoveVertex.Margin = new Padding(4, 5, 4, 5);

btnRemoveVertex.Name = "btnRemoveVertex";

btnRemoveVertex.Size = new Size(133, 35);

btnRemoveVertex.TabIndex = 2;

btnRemoveVertex.Text = "Удалить вершину";

btnRemoveVertex.UseVisualStyleBackColor = true;

btnRemoveVertex.Click += btnRemoveVertex\_Click;

//

// btnAddEdge

//

btnAddEdge.Location = new Point(299, 643);

btnAddEdge.Margin = new Padding(4, 5, 4, 5);

btnAddEdge.Name = "btnAddEdge";

btnAddEdge.Size = new Size(133, 35);

btnAddEdge.TabIndex = 3;

btnAddEdge.Text = "Добавить ребро";

btnAddEdge.UseVisualStyleBackColor = true;

btnAddEdge.Click += btnAddEdge\_Click;

//

// btnRemoveEdge

//

btnRemoveEdge.Location = new Point(440, 643);

btnRemoveEdge.Margin = new Padding(4, 5, 4, 5);

btnRemoveEdge.Name = "btnRemoveEdge";

btnRemoveEdge.Size = new Size(133, 35);

btnRemoveEdge.TabIndex = 4;

btnRemoveEdge.Text = "Удалить ребро";

btnRemoveEdge.UseVisualStyleBackColor = true;

btnRemoveEdge.Click += btnRemoveEdge\_Click;

//

// btnEditWeight

//

btnEditWeight.Location = new Point(581, 643);

btnEditWeight.Margin = new Padding(4, 5, 4, 5);

btnEditWeight.Name = "btnEditWeight";

btnEditWeight.Size = new Size(133, 35);

btnEditWeight.TabIndex = 5;

btnEditWeight.Text = "Изменить вес";

btnEditWeight.UseVisualStyleBackColor = true;

btnEditWeight.Click += btnEditWeight\_Click;

//

// btnDijkstra

//

btnDijkstra.Location = new Point(723, 643);

btnDijkstra.Margin = new Padding(4, 5, 4, 5);

btnDijkstra.Name = "btnDijkstra";

btnDijkstra.Size = new Size(133, 35);

btnDijkstra.TabIndex = 6;

btnDijkstra.Text = "Дейкстра";

btnDijkstra.UseVisualStyleBackColor = true;

btnDijkstra.Click += btnDijkstra\_Click;

//

// btnFloydWarshall

//

btnFloydWarshall.Location = new Point(865, 643); // Позиция кнопки Флойда

btnFloydWarshall.Margin = new Padding(4, 5, 4, 5);

btnFloydWarshall.Name = "btnFloydWarshall";

btnFloydWarshall.Size = new Size(133, 35);

btnFloydWarshall.TabIndex = 21;

btnFloydWarshall.Text = "Флойд";

btnFloydWarshall.UseVisualStyleBackColor = true;

btnFloydWarshall.Click += btnFloydWarshall\_Click; // Обработчик события

//

// txtX

//

txtX.Location = new Point(16, 708);

txtX.Margin = new Padding(4, 5, 4, 5);

txtX.Name = "txtX";

txtX.Size = new Size(65, 27);

txtX.TabIndex = 7;

//

// txtY

//

txtY.Location = new Point(89, 708);

txtY.Margin = new Padding(4, 5, 4, 5);

txtY.Name = "txtY";

txtY.Size = new Size(65, 27);

txtY.TabIndex = 8;

//

// txtVertex

//

txtVertex.Location = new Point(194, 708);

txtVertex.Margin = new Padding(4, 5, 4, 5);

txtVertex.Name = "txtVertex";

txtVertex.Size = new Size(65, 27);

txtVertex.TabIndex = 9;

//

// txtSrc

//

txtSrc.Location = new Point(320, 708);

txtSrc.Margin = new Padding(4, 5, 4, 5);

txtSrc.Name = "txtSrc";

txtSrc.Size = new Size(65, 27);

txtSrc.TabIndex = 10;

//

// txtDest

//

txtDest.Location = new Point(440, 708);

txtDest.Margin = new Padding(4, 5, 4, 5);

txtDest.Name = "txtDest";

txtDest.Size = new Size(65, 27);

txtDest.TabIndex = 11;

//

// txtWeight

//

txtWeight.Location = new Point(606, 708);

txtWeight.Margin = new Padding(4, 5, 4, 5);

txtWeight.Name = "txtWeight";

txtWeight.Size = new Size(65, 27);

txtWeight.TabIndex = 12;

//

// txtStartVertex

//

txtStartVertex.Location = new Point(760, 708);

txtStartVertex.Margin = new Padding(4, 5, 4, 5);

txtStartVertex.Name = "txtStartVertex";

txtStartVertex.Size = new Size(65, 27);

txtStartVertex.TabIndex = 13;

//

// label1

//

label1.AutoSize = true;

label1.Location = new Point(41, 683);

label1.Margin = new Padding(4, 0, 4, 0);

label1.Name = "label1";

label1.Size = new Size(18, 20);

label1.TabIndex = 14;

label1.Text = "X";

//

// label2

//

label2.AutoSize = true;

label2.Location = new Point(107, 683);

label2.Margin = new Padding(4, 0, 4, 0);

label2.Name = "label2";

label2.Size = new Size(17, 20);

label2.TabIndex = 15;

label2.Text = "Y";

//

// label3

//

label3.AutoSize = true;

label3.Location = new Point(194, 688);

label3.Margin = new Padding(4, 0, 4, 0);

label3.Name = "label3";

label3.Size = new Size(73, 20);

label3.TabIndex = 16;

label3.Text = "Вершина";

//

// label4

//

label4.AutoSize = true;

label4.Location = new Point(320, 683);

label4.Margin = new Padding(4, 0, 4, 0);

label4.Name = "label4";

label4.Size = new Size(51, 20);

label4.TabIndex = 17;

label4.Text = "Исход";

//

// label5

//

label5.AutoSize = true;

label5.Location = new Point(427, 683);

label5.Margin = new Padding(4, 0, 4, 0);

label5.Name = "label5";

label5.Size = new Size(94, 20);

label5.TabIndex = 18;

label5.Text = "Назначение";

//

// label6

//

label6.AutoSize = true;

label6.Location = new Point(624, 683);

label6.Margin = new Padding(4, 0, 4, 0);

label6.Name = "label6";

label6.Size = new Size(33, 20);

label6.TabIndex = 19;

label6.Text = "Вес";

//

// label7

//

label7.AutoSize = true;

label7.Location = new Point(723, 683);

label7.Margin = new Padding(4, 0, 4, 0);

label7.Name = "label7";

label7.Size = new Size(147, 20);

label7.TabIndex = 20;

label7.Text = "Стартовая вершина";

//

// GraphForm

//

AutoScaleDimensions = new SizeF(8F, 20F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(1099, 740);

Controls.Add(label7);

Controls.Add(label6);

Controls.Add(label5);

Controls.Add(label4);

Controls.Add(label3);

Controls.Add(label2);

Controls.Add(label1);

Controls.Add(txtStartVertex);

Controls.Add(txtWeight);

Controls.Add(txtDest);

Controls.Add(txtSrc);

Controls.Add(txtVertex);

Controls.Add(txtY);

Controls.Add(txtX);

Controls.Add(btnFloydWarshall); // Добавлено

Controls.Add(btnDijkstra);

Controls.Add(btnEditWeight);

Controls.Add(btnRemoveEdge);

Controls.Add(btnAddEdge);

Controls.Add(btnRemoveVertex);

Controls.Add(btnAddVertex);

Controls.Add(pictureBox);

Margin = new Padding(4, 5, 4, 5);

Name = "GraphForm";

Text = "Graph Visualization";

((System.ComponentModel.ISupportInitialize)pictureBox).EndInit();

ResumeLayout(false);

PerformLayout();

}

#endregion

private System.Windows.Forms.PictureBox pictureBox;

private System.Windows.Forms.Button btnAddVertex;

private System.Windows.Forms.Button btnRemoveVertex;

private System.Windows.Forms.Button btnAddEdge;

private System.Windows.Forms.Button btnRemoveEdge;

private System.Windows.Forms.Button btnEditWeight;

private System.Windows.Forms.Button btnDijkstra;

private System.Windows.Forms.Button btnFloydWarshall; // Добавлено

private System.Windows.Forms.TextBox txtX;

private System.Windows.Forms.TextBox txtY;

private System.Windows.Forms.TextBox txtVertex;

private System.Windows.Forms.TextBox txtSrc;

private System.Windows.Forms.TextBox txtDest;

private System.Windows.Forms.TextBox txtWeight;

private System.Windows.Forms.TextBox txtStartVertex;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.Label label5;

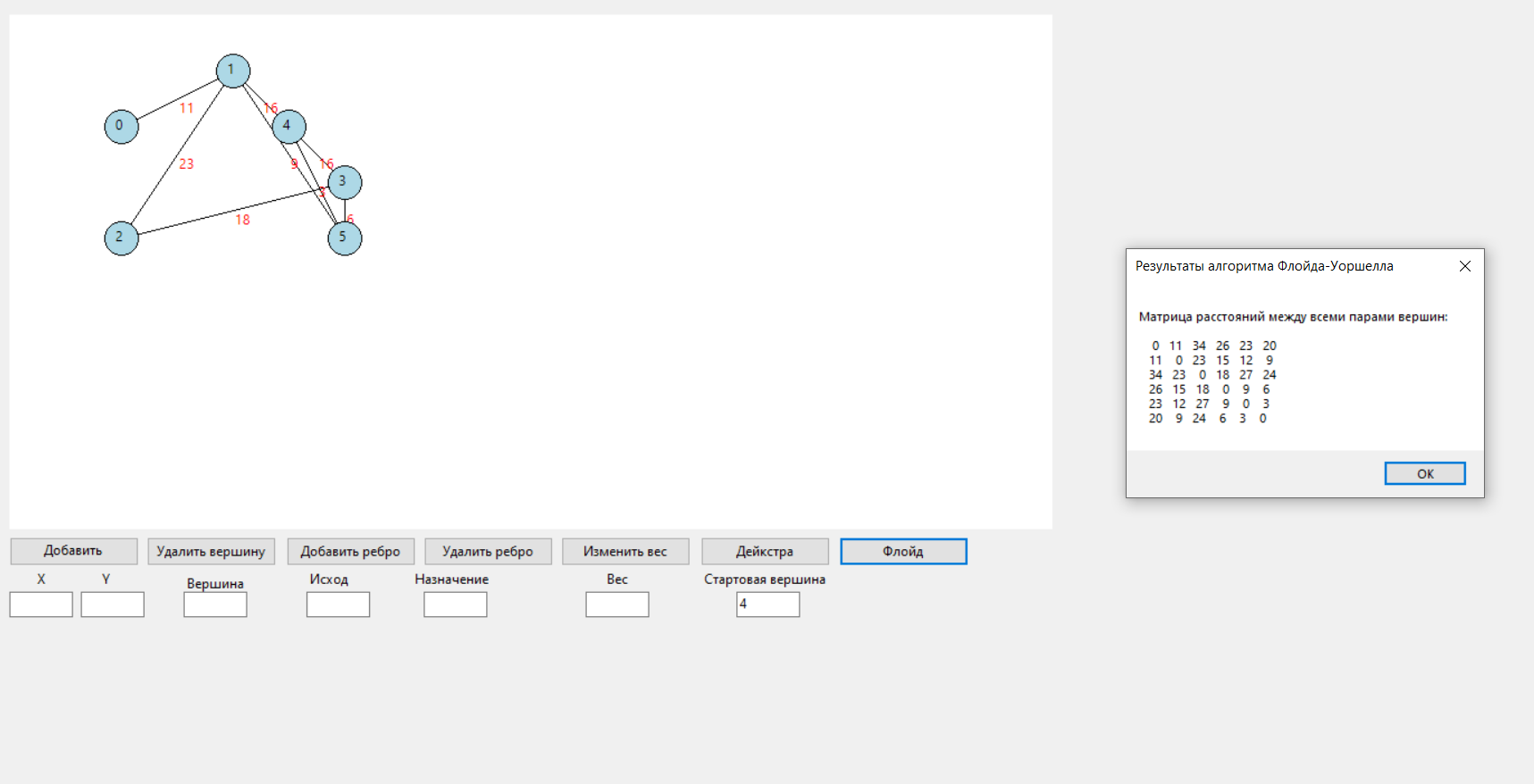
private System.Windows.Forms.Label label6;

private System.Windows.Forms.Label label7;

}

}

Результат работы:



Ссылка на GitHub:  
https://github.com/LeonidZhir/-