Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Кубанский государственный университет»

Кафедра вычислительных технологий

**ОТЧЕТ**

о выполнении лабораторной работы № 15

по дисциплине «Конструирование алгоритмов и структур данных»

Выполнил: ст. гр. 26/1
Аванесов Р.А.
Проверил: доц.каф. ВТ
Полетайкин А.Н.

Краснодар
2023

**Тема:** Обработка графов в C#. Двойная буферизация графики.

**Цель:** освоение навыков составления программ с графическим интерфейсом Windows для работы с простыми и ориентированными графами, с реализацией дополнительного буфера для обеспечения плавности анимации.

## Задание

1. Написать программную реализацию класса графа с возможностью добавления элементов во множество вершин и во множество ребер. Способ представления ребер зависит от варианта индивидуальной задачи (списки смежности, матрица смежности, список ребер). Реализовать графический интерфейс с двойной буферизацией, в котором элементы графа можно свободно перемещать по области рисования.

2. Разработать метод класса для обработки данных графа согласно индивидуальному заданию из табл. 15.1. Результат обработки отражать графически и выводить посредством элементов управления TextBox согласно индивидуальному заданию из табл. 15.1.

3. Операции загрузки, сохранения и обработки графа инициировать посредством элементов управления Button.

4. Разработать модульный тест для метода обработки графа. При проверке читать граф из файла G.grf.

5. При программировании задачи выполнять обработку исключительных ситуаций.

6. Представить результаты выполнения программы и сделать выводы по работе.

7. При программировании задачи выполнять обработку исключительных ситуаций.

Индивидуальное задание из таблицы 15.1

| № варианта | Задание | Вид графа | Способ представления | Метод обхода |
|---|---|---|---|---|
| 1 | Топологическая сортировка. Вывести последовательно названия вершин в TextBox через запятую. | О | Список смежности | DFS |

## Ход работы

Код программы:

**Файл “Node.cs”:**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab15
{
    public class Node
    {

        public List<Edge> edges;

        public Point position {  get; set; }


        public Int32 Id { get; set; }

        public Node(Int32 id)
        {
            edges = new List<Edge>();
            Id = id;
        }

        public void AddEdgeTo(Node node)
        {
            Edge edge = new Edge(this, node);
            edges.Add(edge);
        }

        public void AddEdgeFrom(Node node)
        {
            Edge edge = new Edge(node, this);
            node.edges.Add(edge);
        }

    }
}
```

**Файл “Edge.cs”:**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab15
{
    public class Edge
    {
        public Node Source { get; set; }
        public Node Target { get; set; }
```

```
        public Edge(Node source, Node target)
        {
            Source = source;
            Target = target;
        }
    }
}
```

**Файл “Graph.cs”:**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms.VisualStyles;
using System.Xml.Linq;

namespace Lab15
{
    public class Graph
    {

        public List<Node> nodes;

        public Graph()
        {
            nodes = new List<Node>();
        }

        public void AddNode(Node node)
        {
            nodes.Add(node);
        }

        private char[] colors;
        private bool[] visited;

        private void AcyclicGraph_DFS(Node currentNode)
        {
            colors[currentNode.Id - 1] = 'G';
            foreach (Edge edge in currentNode.edges)
            {
                if (colors[edge.Target.Id - 1] == 'W')
AcyclicGraph_DFS(edge.Target);
                if (colors[edge.Target.Id - 1] == 'G') isAcyclic = false;
            }
            colors[currentNode.Id - 1] = 'B';
        }

        private bool isAcyclic = true;

        private void TopoSort_DFS(Node node, List<Node> sorted_graph)
        {
            if (visited[node.Id - 1] == true) { return; }
            foreach (Edge edge in node.edges)
            {
                TopoSort_DFS(edge.Target, sorted_graph);
```

```
            }
            visited[node.Id - 1] = true;
            sorted_graph.Insert(0, node);
        }

        public List<Node> TopoSort()
        {

            colors = new char[nodes.Count];
            visited = new bool[nodes.Count];
            for (int i = 0; i < nodes.Count; ++i)
            {
                colors[i] = 'W';
            }
            AcyclicGraph_DFS(nodes[0]);

            if (!isAcyclic) { return null; }

            string result = string.Empty;

            for (int i = 0; i < nodes.Count; ++i)
            {
                visited[i] = false;
            }

            var sortedGraph = new List<Node>();
            foreach (Node node in nodes)
            {
                TopoSort_DFS(node, sortedGraph);
            }
            return sortedGraph;
        }

        public Int32 GetGlobalId()
        {
            int maxId = 0;
            foreach (Node node in nodes)
            {
                if (node.Id > maxId) { maxId = node.Id; }
            }
            return maxId;
        }

        public void SaveToFile(StreamWriter sw)
        {
            try
            {
                if (sw == null) { throw new
NullReferenceException("Проблема при сохранении в файл."); }
                foreach (Node node in nodes)
                {
                    string line = $"{node.Id}";
                    foreach(var edge in node.edges)
                    {
                        line += $" {edge.Target.Id}";
                    }
                    sw.WriteLine(line);
                }

            }
```

```
            catch(Exception ex)
            {
                throw;
            }
        }

        static public Graph LoadFromFile(StreamReader sr)
        {
            try
            {
                if (sr == null) { throw new
NullReferenceException("Проблема при открытии файла."); }
                Graph newGraph = new Graph();

                while (!sr.EndOfStream)
                {
                    string[] line = sr.ReadLine().Split(' ');


                    Node firstNode = new Node(int.Parse(line[0]));
                    bool nodeIsAlreadyExist = false;
                    foreach (var node in newGraph.nodes)
                    {
                        if (node.Id == firstNode.Id) { nodeIsAlreadyExist
= true; break; }
                    }
                    if (!nodeIsAlreadyExist) newGraph.AddNode(firstNode);


                    for (int i = 1; i < line.Length; ++i)
                    {
                        nodeIsAlreadyExist = false;
                        int newNodeId = int.Parse(line[i]);

                        Node currentNode = null;
                        foreach (var node in newGraph.nodes)
                        {
                            if (node.Id == newNodeId) {nodeIsAlreadyExist
= true; currentNode = node; break; }
                        }
                        if (nodeIsAlreadyExist)
firstNode.AddEdgeTo(currentNode);
                        else
                        {
                            currentNode = new Node(newNodeId);
                            newGraph.AddNode(currentNode);
                            firstNode.AddEdgeTo(currentNode);
                        }

                    }

                }

                foreach (var node in newGraph.nodes)
                {
                    Random rnd = new Random();
                    node.position = new Point(rnd.Next(0, 1000),
rnd.Next(0, 500));
                }
```

```
                return newGraph;
            }
            catch (Exception ex)
            {
                throw;
            }
        }

    }
}
```

**Файл “Form1.cs”:**

```
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Net;
using System.Security.Cryptography;
using System.Xml.Linq;

namespace Lab15
{
    public partial class Form1 : Form
    {
        Graph graph { get; set; } = new Graph();


        Pen nodePen = new Pen(Color.Black, 4);
        Pen edgePen = new Pen(Color.Black, 2);
        Brush nodeBackGroundBrush = Brushes.Black;

        Node currentlyDraggableNode = null;
        Node currentSourceNode = null;

        const int node_size = 25;

        Graphics g;

        Point newEdgePoint;


        public Form1()
        {
            InitializeComponent();
        }

        private void addNodeButton_Click(object sender, EventArgs e)
        {
            Node newNode = new Node(graph.GetGlobalId() + 1);
            Random rnd = new Random();
            newNode.position = new Point(rnd.Next(0, 1000), rnd.Next(0,
500));
            graph.AddNode(newNode);
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            this.DoubleBuffered = true;
        }

        private void Form1_Paint(object sender, PaintEventArgs e)
```

```
        {
            pictureBox1.Width = Width;
            pictureBox1.Height = Height;


            Bitmap buffer = new Bitmap(Width, Height);
            g = Graphics.FromImage(buffer);
            g.Clear(Color.White);

            foreach (var node in graph.nodes)
            {
                foreach (var edge in node.edges)
                {
                    var first_point = new Point(edge.Source.position.X +
12, edge.Source.position.Y + 12);
                    var second_point = new Point(edge.Target.position.X +
12, edge.Target.position.Y + 12);

                    g.DrawLine(edgePen, first_point, second_point);


                    float angle = (float)(Math.Atan2(second_point.Y -
first_point.Y, second_point.X - first_point.X) * 180 / Math.PI);

                    float arrowSize = 10;
                    float endX = second_point.X - arrowSize *
(float)Math.Cos(angle * Math.PI / 180);
                    float endY = second_point.Y - arrowSize *
(float)Math.Sin(angle * Math.PI / 180);


                    Pen arrowPen = new Pen(Color.Black);
                    AdjustableArrowCap arrowCap = new
AdjustableArrowCap(arrowSize, arrowSize, true);
                    arrowPen.CustomEndCap = arrowCap;
                    g.DrawLine(arrowPen, first_point, new
Point((int)endX, (int)endY));

                }
            }

            foreach (var node in graph.nodes)
            {

                g.DrawEllipse(nodePen, node.position.X, node.position.Y,
node_size, node_size);
                g.FillEllipse(nodeBackGroundBrush, node.position.X,
node.position.Y, node_size, node_size);
                g.DrawString($"{node.Id}", new Font("Segoe UI", 12F,
FontStyle.Regular, GraphicsUnit.Point), Brushes.White, new
Point(node.position.X + 6, node.position.Y + 3));
            }

            if (currentSourceNode != null)
            {
                var temp = new Point(currentSourceNode.position.X + 8,
currentSourceNode.position.Y + 8);
                g.DrawLine(edgePen, temp, newEdgePoint);
            }
```

```
            pictureBox1.Image = buffer;
        }

        private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
        {
            var mouse_position = e.Location;
            if (currentlyDraggableNode != null && e.Button ==
MouseButtons.Left)
            {
                currentlyDraggableNode = null;
            }
            if (currentSourceNode != null && e.Button ==
MouseButtons.Right)
            {
                foreach (var node in graph.nodes)
                {
                    if (new Rectangle(node.position.X, node.position.Y,
node_size, node_size).Contains(mouse_position))
                    {
                        node.AddEdgeFrom(currentSourceNode);
                    }
                }
                currentSourceNode = null;
            }
        }

        private void pictureBox1_MouseDown(object sender, MouseEventArgs
e)
        {
            var mouse_position = e.Location;

            {
                foreach (var node in graph.nodes)
                {
                    if (new Rectangle(node.position.X, node.position.Y,
node_size, node_size).Contains(mouse_position))
                    {
                        if (currentlyDraggableNode == null && e.Button ==
MouseButtons.Left)
                        {
                            currentlyDraggableNode = node;
                        }
                        if (currentSourceNode == null && e.Button ==
MouseButtons.Right)
                        {
                            currentSourceNode = node;
                        }
                    }
                }
            }

        }

        private void pictureBox1_MouseMove(object sender, MouseEventArgs
e)
        {
            var mouse_position = e.Location;
            if (currentlyDraggableNode != null)
            {
```

```
                currentlyDraggableNode.position = new
Point(mouse_position.X - 12, mouse_position.Y - 12);
            }
            if (currentSourceNode != null)
            {
                newEdgePoint = mouse_position;
            }
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            Refresh();
        }

        private void saveButton_Click(object sender, EventArgs e)
        {
            try
            {
                StreamWriter streamWriter = new
StreamWriter("C:\\Users\\romanavanesov\\source\\repos\\Lab15\\Lab15\\G.gr
f");
                graph.SaveToFile(streamWriter);
                streamWriter.Close();


                MessageBox.Show(
                    "Graph successfully saved.",
                    "Saving",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information,
                    MessageBoxDefaultButton.Button1,
                    MessageBoxOptions.DefaultDesktopOnly);

            }
            catch (Exception ex)
            {
                MessageBox.Show(
                    ex.Message,
                    "Error",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error,
                    MessageBoxDefaultButton.Button1,
                    MessageBoxOptions.DefaultDesktopOnly);
                throw;
            }


        }

        private void loadButton_Click(object sender, EventArgs e)
        {
            try
            {
                StreamReader streamReader = new
StreamReader("C:\\Users\\romanavanesov\\source\\repos\\Lab15\\Lab15\\G.gr
f");
                graph = Graph.LoadFromFile(streamReader);
                streamReader.Close();

                MessageBox.Show(
```

```
                    "Graph successfully loaded.",
                    "Loading",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information,
                    MessageBoxDefaultButton.Button1,
                    MessageBoxOptions.ServiceNotification);

            }
            catch (Exception ex)
            {
                MessageBox.Show(
                    ex.Message,
                    "Error",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error,
                    MessageBoxDefaultButton.Button1,
                    MessageBoxOptions.ServiceNotification);
                throw;
            }
        }

        private void sortButton_Click(object sender, EventArgs e)
        {
            var sortedGraph = graph.TopoSort();

            if (sortedGraph == null) { sortedTextBox.Text = "Graph is no
acyclic!"; return;  }

            var result = "";
            for (int i = 0; i < sortedGraph.Count; ++i)
            {
                if (i != sortedGraph.Count - 1) result +=
$"{sortedGraph[i].Id},";
                else result += $"{sortedGraph[i].Id}";
            }
            sortedTextBox.Text = result;
        }
    }
}
```

**Файл “Form1.Designer.cs”:**

```
namespace Lab15
{
    partial class Form1
    {
        /// <summary>
        ///  Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        ///  Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
```

```
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        ///  Required method for Designer support - do not modify
        ///  the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            components = new System.ComponentModel.Container();
            addNodeButton = new Button();
            addNodeLable = new Label();
            pictureBox1 = new PictureBox();
            timer1 = new System.Windows.Forms.Timer(components);
            label1 = new Label();
            label2 = new Label();
            label3 = new Label();
            saveButton = new Button();
            loadButton = new Button();
            sortedTextBox = new TextBox();
            sortButton = new Button();

((System.ComponentModel.ISupportInitialize)pictureBox1).BeginInit();
            SuspendLayout();
            //
            // addNodeButton
            //
            addNodeButton.Font = new Font("Segoe UI", 12F,
FontStyle.Regular, GraphicsUnit.Point);
            addNodeButton.Location = new Point(77, 31);
            addNodeButton.Margin = new Padding(2, 1, 2, 1);
            addNodeButton.Name = "addNodeButton";
            addNodeButton.Size = new Size(81, 29);
            addNodeButton.TabIndex = 0;
            addNodeButton.Text = "Add";
            addNodeButton.UseVisualStyleBackColor = true;
            addNodeButton.Click += addNodeButton_Click;
            //
            // addNodeLable
            //
            addNodeLable.AutoSize = true;
            addNodeLable.Font = new Font("Segoe UI", 12F,
FontStyle.Regular, GraphicsUnit.Point);
            addNodeLable.Location = new Point(62, 9);
            addNodeLable.Margin = new Padding(2, 0, 2, 0);
            addNodeLable.Name = "addNodeLable";
            addNodeLable.Size = new Size(110, 21);
            addNodeLable.TabIndex = 1;
            addNodeLable.Text = "Add new node";
            addNodeLable.TextAlign = ContentAlignment.MiddleCenter;
            //
            // pictureBox1
            //
            pictureBox1.Location = new Point(3, 72);
            pictureBox1.Margin = new Padding(2, 1, 2, 1);
            pictureBox1.Name = "pictureBox1";
```

```
            pictureBox1.Size = new Size(1330, 510);
            pictureBox1.TabIndex = 2;
            pictureBox1.TabStop = false;
            pictureBox1.MouseDown += pictureBox1_MouseDown;
            pictureBox1.MouseMove += pictureBox1_MouseMove;
            pictureBox1.MouseUp += pictureBox1_MouseUp;
            //
            // timer1
            //
            timer1.Enabled = true;
            timer1.Interval = 1;
            timer1.Tick += timer1_Tick;
            //
            // label1
            //
            label1.AutoSize = true;
            label1.Font = new Font("Segoe UI", 12F, FontStyle.Regular,
GraphicsUnit.Point);
            label1.Location = new Point(1097, 9);
            label1.Margin = new Padding(2, 0, 2, 0);
            label1.Name = "label1";
            label1.Size = new Size(86, 21);
            label1.TabIndex = 3;
            label1.Text = "Save to file";
            //
            // label2
            //
            label2.AutoSize = true;
            label2.Font = new Font("Segoe UI", 12F, FontStyle.Regular,
GraphicsUnit.Point);
            label2.Location = new Point(1208, 9);
            label2.Margin = new Padding(2, 0, 2, 0);
            label2.Name = "label2";
            label2.Size = new Size(107, 21);
            label2.TabIndex = 4;
            label2.Text = "Load form file";
            //
            // label3
            //
            label3.AutoSize = true;
            label3.Font = new Font("Segoe UI", 12F, FontStyle.Regular,
GraphicsUnit.Point);
            label3.Location = new Point(524, 9);
            label3.Margin = new Padding(2, 0, 2, 0);
            label3.Name = "label3";
            label3.Size = new Size(103, 21);
            label3.TabIndex = 5;
            label3.Text = "Sorted Graph";
            //
            // saveButton
            //
            saveButton.Font = new Font("Segoe UI", 12F,
FontStyle.Regular, GraphicsUnit.Point);
            saveButton.Location = new Point(1097, 32);
            saveButton.Margin = new Padding(2, 1, 2, 1);
            saveButton.Name = "saveButton";
            saveButton.Size = new Size(81, 30);
            saveButton.TabIndex = 6;
            saveButton.Text = "Save";
            saveButton.UseVisualStyleBackColor = true;
```

```
            saveButton.Click += saveButton_Click;
            //
            // loadButton
            //
            loadButton.Font = new Font("Segoe UI", 12F,
FontStyle.Regular, GraphicsUnit.Point);
            loadButton.Location = new Point(1221, 32);
            loadButton.Margin = new Padding(2, 1, 2, 1);
            loadButton.Name = "loadButton";
            loadButton.Size = new Size(81, 30);
            loadButton.TabIndex = 7;
            loadButton.Text = "Load";
            loadButton.UseVisualStyleBackColor = true;
            loadButton.Click += loadButton_Click;
            //
            // sortedTextBox
            //
            sortedTextBox.Font = new Font("Segoe UI", 12F,
FontStyle.Regular, GraphicsUnit.Point);
            sortedTextBox.Location = new Point(203, 32);
            sortedTextBox.Margin = new Padding(2, 1, 2, 1);
            sortedTextBox.Name = "sortedTextBox";
            sortedTextBox.Size = new Size(740, 29);
            sortedTextBox.TabIndex = 8;
            sortedTextBox.TextAlign = HorizontalAlignment.Center;
            //
            // sortButton
            //
            sortButton.Font = new Font("Segoe UI", 12F,
FontStyle.Regular, GraphicsUnit.Point);
            sortButton.Location = new Point(947, 32);
            sortButton.Margin = new Padding(2, 1, 2, 1);
            sortButton.Name = "sortButton";
            sortButton.Size = new Size(81, 30);
            sortButton.TabIndex = 9;
            sortButton.Text = "Sort";
            sortButton.UseVisualStyleBackColor = true;
            sortButton.Click += sortButton_Click;
            //
            // Form1
            //
            AutoScaleDimensions = new SizeF(7F, 15F);
            AutoScaleMode = AutoScaleMode.Font;
            ClientSize = new Size(1332, 580);
            Controls.Add(sortButton);
            Controls.Add(sortedTextBox);
            Controls.Add(loadButton);
            Controls.Add(saveButton);
            Controls.Add(label3);
            Controls.Add(label2);
            Controls.Add(label1);
            Controls.Add(pictureBox1);
            Controls.Add(addNodeLable);
            Controls.Add(addNodeButton);
            DoubleBuffered = true;
            Margin = new Padding(2, 1, 2, 1);
            Name = "Form1";
            Text = "Graph";
            Paint += Form1_Paint;
```

```
((System.ComponentModel.ISupportInitialize)pictureBox1).EndInit();
            ResumeLayout(false);
            PerformLayout();
        }

        #endregion

        private Button addNodeButton;
        private Label addNodeLable;
        private PictureBox pictureBox1;
        private System.Windows.Forms.Timer timer1;
        private Label label1;
        private Label label2;
        private Label label3;
        private Button saveButton;
        private Button loadButton;
        private TextBox sortedTextBox;
        private Button sortButton;
    }
}
```

**Файл “Program.cs”:**

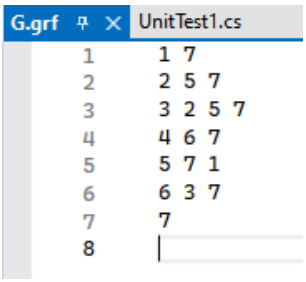
```
namespace Lab15
{
    internal static class Program
    {
        /// <summary>
        ///  The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            // To customize application configuration such as set high
DPI settings or default font,
            // see https://aka.ms/applicationconfiguration.
            ApplicationConfiguration.Initialize();
            Application.Run(new Form1());
        }
    }
}
```

**Файл “UnitTest1.cs”:**

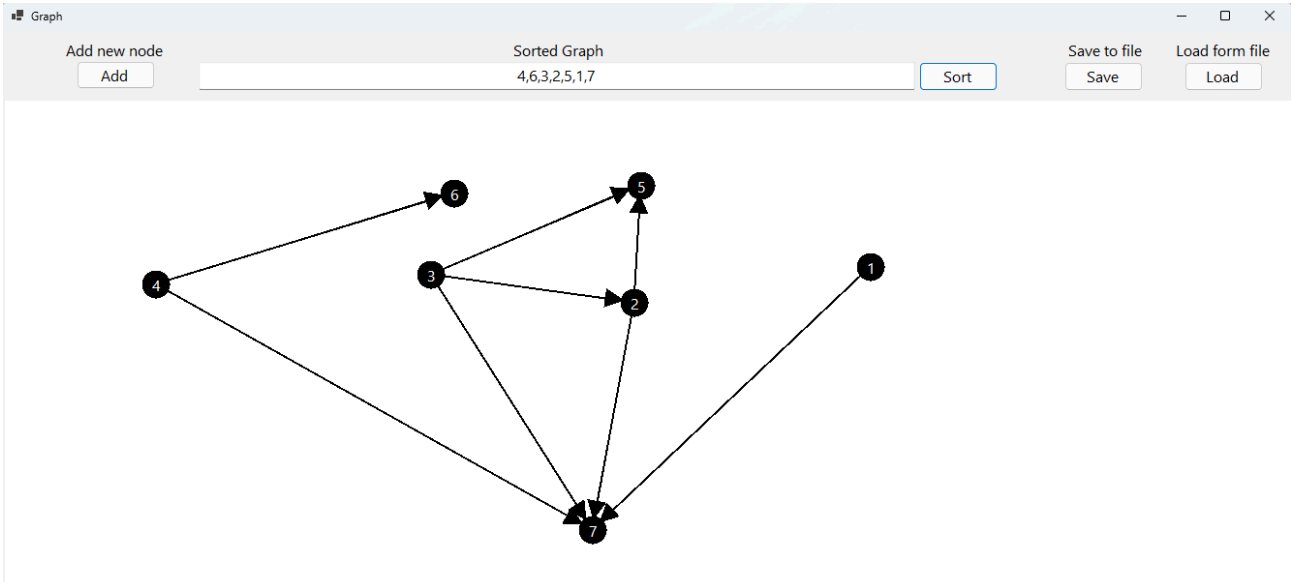
```
namespace Lab15
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void Test_TopoSort_AssertAcyclicSorted()
        {

            StreamReader streamReader = new
StreamReader("C:\\Users\\romanavanesov\\source\\repos\\Lab15\\Lab15\\G.gr
f");
            Graph graph = Graph.LoadFromFile(streamReader);
```

```
            List<Node> sortedGraph = graph.TopoSort();
            List<int> ints = new List<int> { 4, 6, 3, 2, 5, 1, 7};

            for (int i = 0; i < ints.Count; i++)
            {
                Assert.AreEqual(sortedGraph[i].Id, ints[i]);
            }
        }

        [TestMethod]
        public void Test_TopoSort_AssertCyclic()
        {
            Graph graph = new Graph();
            graph.AddNode(new Node(1));
            graph.AddNode(new Node(2));
            graph.AddNode(new Node(3));
            graph.AddNode(new Node(4));

            graph.nodes[0].AddEdgeTo(graph.nodes[1]);
            graph.nodes[1].AddEdgeTo(graph.nodes[2]);
            graph.nodes[2].AddEdgeTo(graph.nodes[0]);

            Assert.AreEqual(graph.TopoSort(), null);
        }
    }
}
```

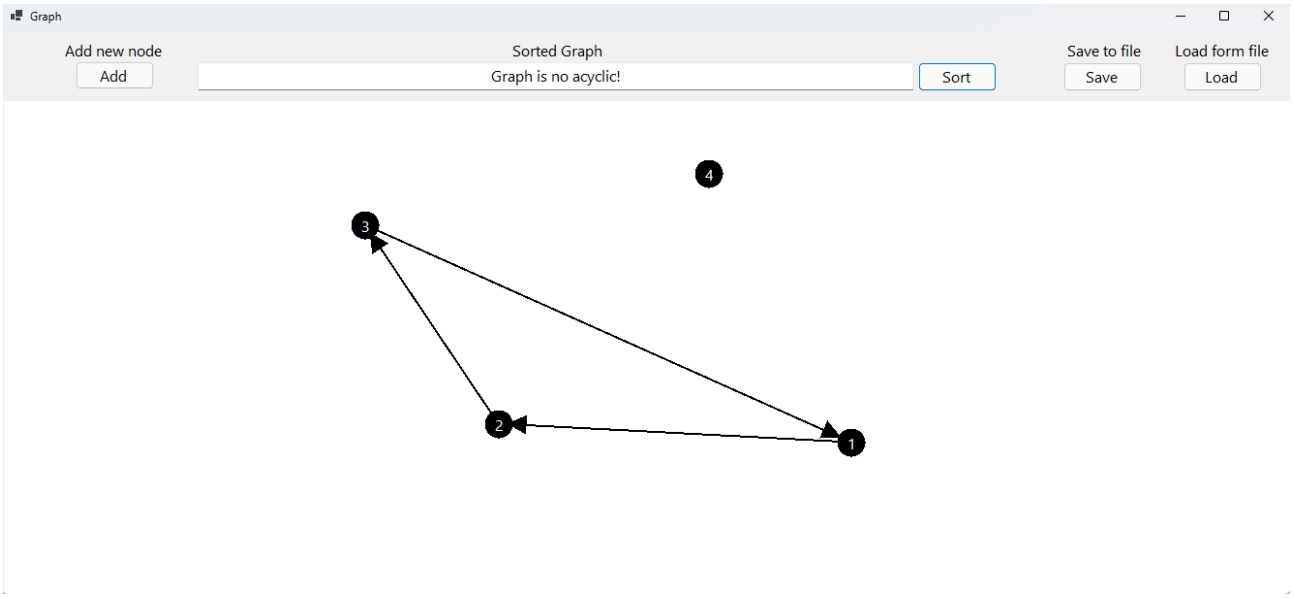
**Тестирование**:
Граф для метода `Test_TopoSort_AssertAcyclicSorted()`:



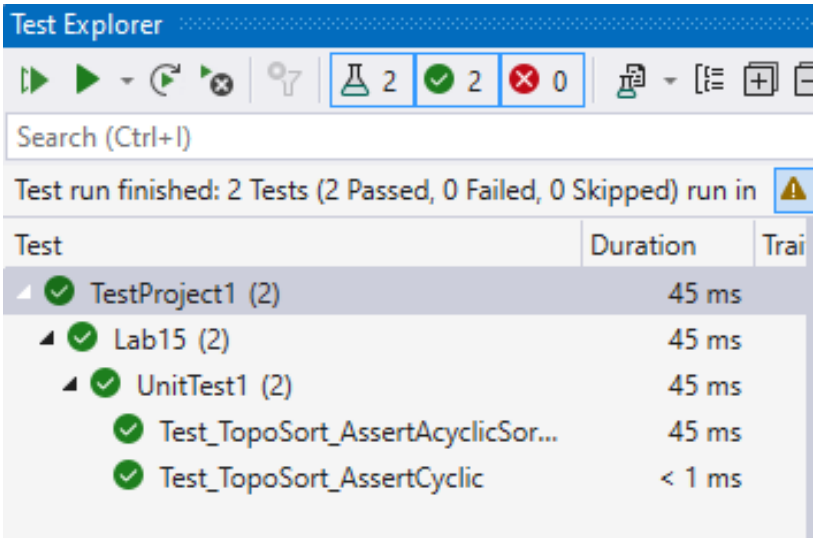

Граф для этого метода в графическом виде и с примененной сортировкой:



Граф для метода `Test_TopoSort_AssertCyclic()`:

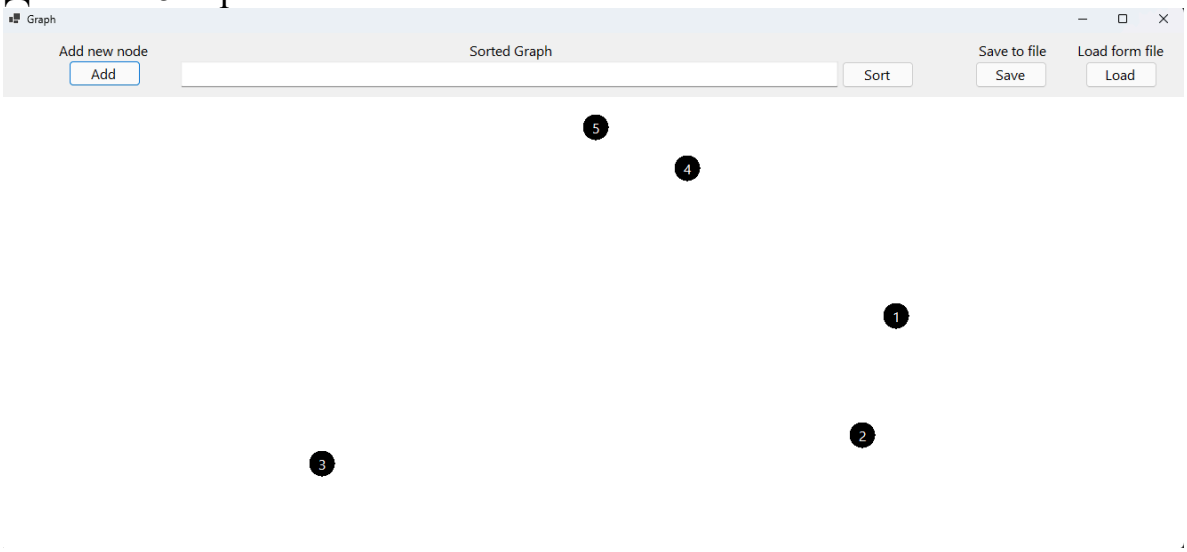



Данный граф невозможно отсортировать, поскольку он имеет цикл.

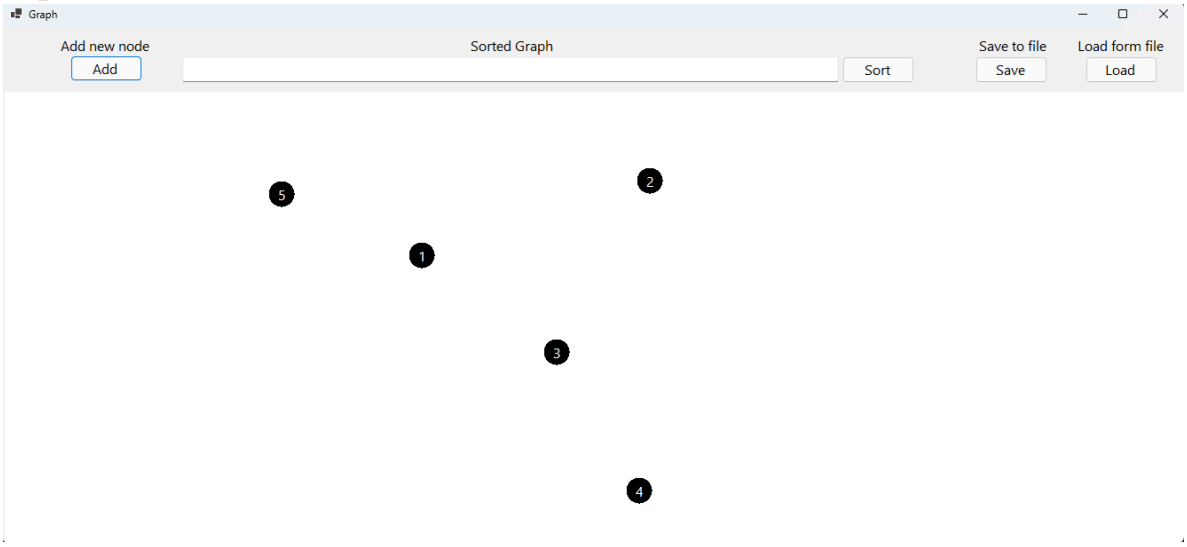


Результат программы:

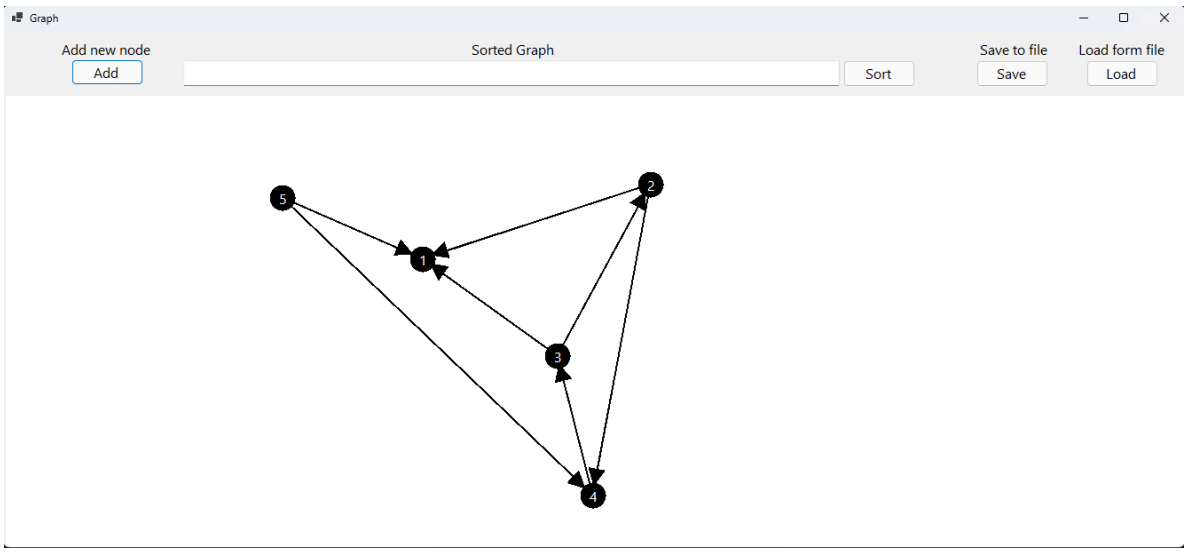
Добавлю 5 вершин.




Наведу курсор на вершину и, зажав левую кнопку мыши, перемещу эти вершины.

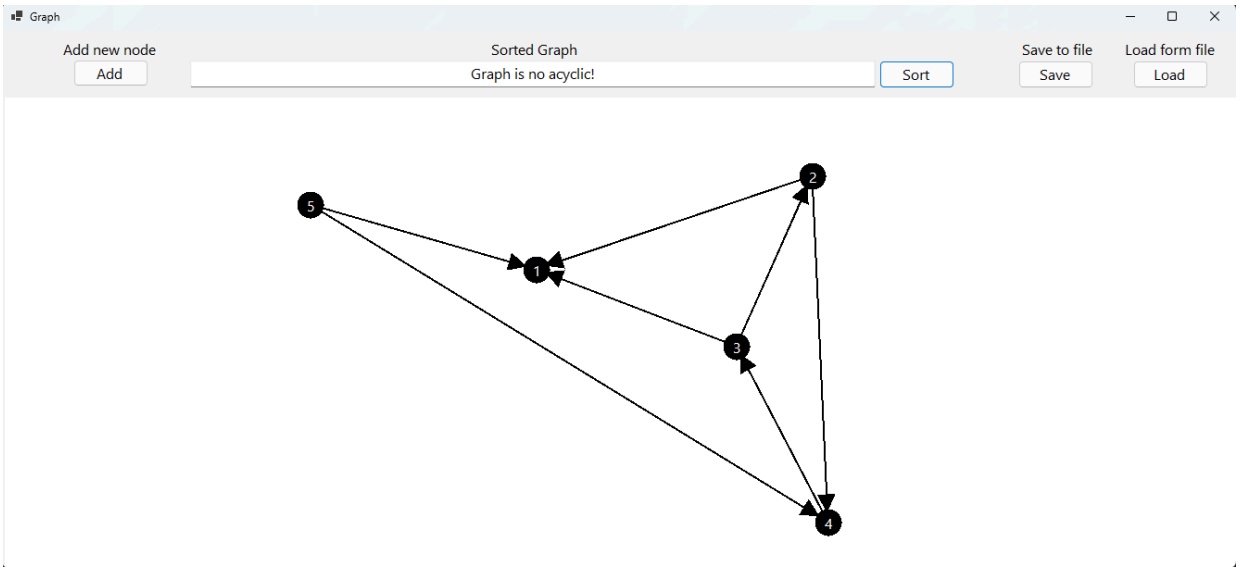



Теперь наведу курсор на вершину и зажму правую кнопку мыши, появится линия, переведу курсор с зажатой правой кнопкой мыши на другую вершину и отожму ее - образуется ребро. Повторю операцию несколько раз.

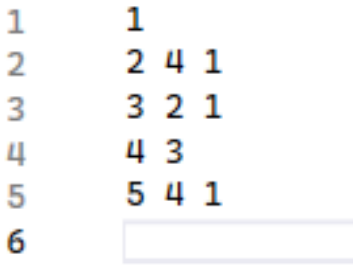


Применю алгоритм сортировки к построенному графу.

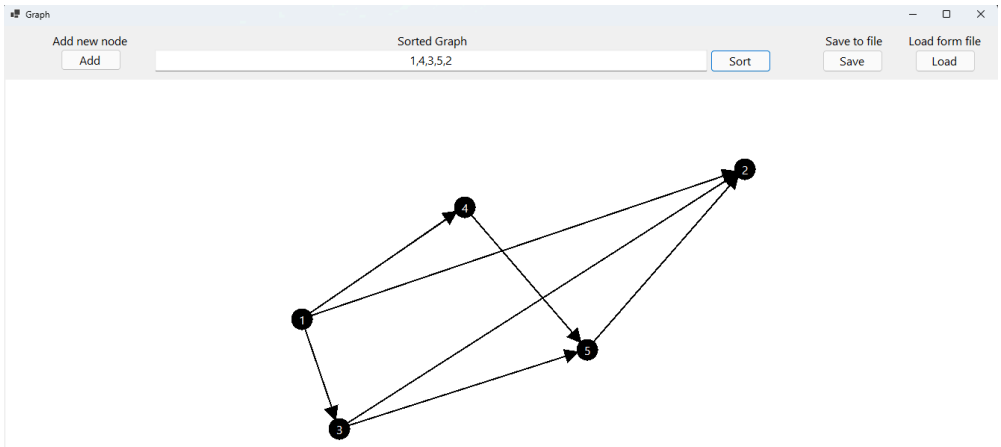
Как видим сортировка не возможна поскольку граф содержит цикл.

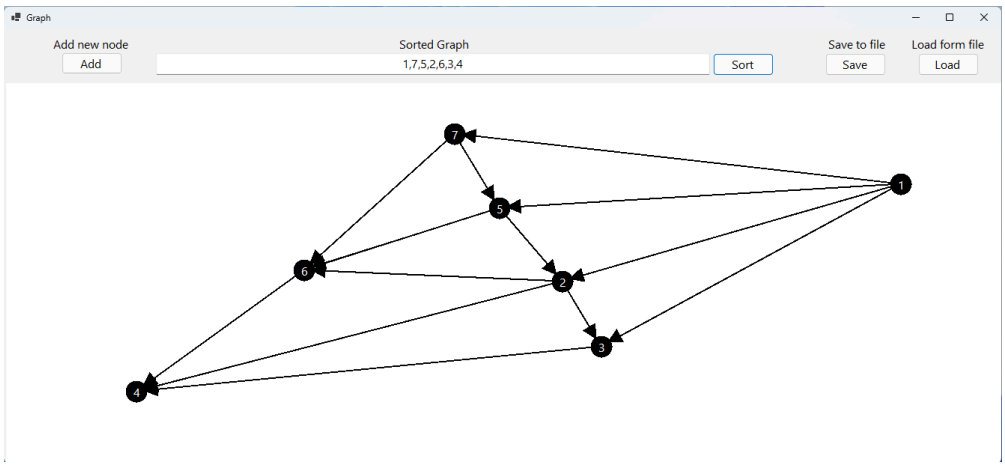



Теперь сохраню данный граф в файл.



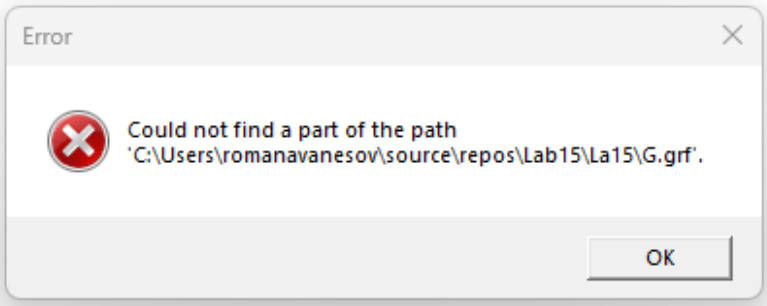

Еще примеры работы программы:



Обработка исключительных ситуаций:

При попытке считать граф из несуществующего файла вызывается MessageBox.




**Вывод:**

В ходе лабораторной работы изучил новые возможности создания программ с графическим интерфейсом, использующим Windows Forms. В ходе работы реализовал программу для работы с ориентированным графом, выполняющую топологическую сортировку созданного или считанного из файла графа. Также познакомился с двойной буферизацией. Были созданы модульные тесты для тестирования программы.