МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра Информатики и систем управления

Лабораторная работа №6 «Проектирование пользовательского интерфейса.»

ОТЧЕТ по лабораторной работе № 6

по дисциплине

Технологии программирования

РУКОВОДИТЕЛЬ:

_____ Капранов С.Н.
(подпись) (фамилия, и.,о.)

СТУДЕНТ:

_____ Куликова Е.А.
(подпись) (фамилия, и.,о.)

18-ИСТ-4
(шифр группы)

Работа защищена «___» _____

С оценкой _____

Нижний Новгород

2020

Содержание

Введение

Сетевой график – модель производственного процесса, отражающая последовательность выполнения комплекса работ, связывающая их свершение во времени с учётом затрат ресурсов и стоимости работ с выделением при этом критических мест. Основные элементы сетевого графика – работа и событие.

Работа отражает трудовой процесс, в котором участвуют люди, машины, механизмы, материальные ресурсы (проектирование сооружения и систем, поставки оборудования, кладка стен, решение задач на ЭВМ и т. п.) либо процесс ожидания (твердение бетона, сушка штукатурки и т. п.). Работа как трудовой процесс требует затрат времени и ресурсов, как ожидание – только времени. Для наглядного отображения порядка предшествования работ при построении сети используют изображаемые штриховыми линиями дополнительные дуги, называемые фиктивными работами. Они не требуют времени и ресурсов, они указывают, что начало одной работы зависит от окончания другой.

Событие выражает факт окончания одной или нескольких предшествующих работ, необходимых для начала следующих работ. Событие, стоящее в начале работы, называется начальным, в конце – конечным. Начальное событие сетевого графика называется исходным, конечное – завершающим. Событие, не являющееся ни исходным, ни завершающим, называется промежуточным. В отличие от работ, события совершаются мгновенно без потребления ресурсов.

В шестой лабораторной работе (Вариант 10 – следствие порядкового номера в списке группы) необходимо выполнить следующее задание: Разработать программное обеспечение для ввода, сохранения, загрузки и отображения данных, представленных в виде графа. Пользователь должен иметь возможность ввести ориентированный и неориентированный граф, деревья, различные варианты сохранения**.** Тема работы – сетевой график производства работ.

1. Цель работы

Создать программу, соответствующую указаниям шестой лабораторной работы, то есть программу с вводом, сохранением, загрузкой и отображением данных, представленных в виде графа. Пользователь должен иметь возможность ввести ориентированный и неориентированный граф, деревья, также должны присутствовать различные варианты сохранения.

2. Задачи

Поставленные задачи:

1. Написать код, реализующий задание;
2. Разработать интерфейс;
3. Протестировать, чтобы убедиться в правильности решения.

3. Код программы

Node.cs

```
namespace ProjectNetwork
{
    class Node
    {// Класс Событие
        string state; // Состояние
        int x; // Координата x
        int y; // Координата y
        bool input; // Есть ли входящие ребра

        public string State => state;
        public int X => x;
        public int Y => y;
        public bool Input { get => input; set => input = value; }

        public Node(string s = "")
        {
            state = s;
            x = -1;
            y = -1;
            input = false;
        }
        // Функция заполняющая координаты узла
        public void Coordinates(int _x, int _y)
        {
            x = _x;
            y = _y;
        }
        // Функция проверяющая выбран ли данный узел
        public bool Selected(int _x, int _y)
        {
```

```
            if (_x > x - 15 && _x < x + 15)
                if (_y > y - 15 && _y < y + 15)
                    return true;
            return false;
        }
    }
}
```

Листинг 1 – файл Node.cs

Arrow.cs

```
using System;

namespace ProjectNetwork
{
    class Arrow
    {// Класс Работа
        string action; // Действие
        uint time; // Время
        string resource; // Ресурсы
        Node nodeIn = null; // Узел начала
        Node nodeOut = null; // Узел конца

        public string Action => action;
        public uint Time => time;
        public string Resource => resource;
        public Node NodeIn => nodeIn;
        public Node NodeOut => nodeOut;

        public Arrow(string a, uint t, string r)
        {
            action = a;
            time = t;
            resource = r;
        }
        // Функция заполняющая узлы ребра
        public void InOut(Node n1, Node n2)
        {
            nodeIn = n1;
            nodeOut = n2;
        }
        // Функция проверяющая выбрано ли данное ребро
        public bool Selected(int _x, int _y)
        {
            int p1x = nodeIn.X, p1y = nodeIn.Y;
            int p2x = nodeOut.X, p2y = nodeOut.Y;
            if (nodeIn == nodeOut)
            {// Если петля
                if (_x > p1x - 30 && _x < p1x)
                    if (_y > p1y - 30 && _y < p1y)
                        return true;
            }
            if (Length(p1x, p1y, _x, _y) + Length(p2x, p2y, _x, _y) -
            Length(p1x, p1y, p2x, p2y) < 1)
                return true;
            return false;
        }
        // Расстояние между двумя точками
        double Length(int x1, int y1, int x2, int y2) =>
            Math.Sqrt(Math.Pow(x1 - x2, 2) + Math.Pow(y1 - y2, 2));
    }
}
```

Листинг 2 – файл Arrow.cs

Form1.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace ProjectNetwork
{
    public partial class Form1 : Form
    {
        SqlConnection connection;
        Dictionary<uint, Node> nodes; // Узлы
        List<Arrow> arrows; // Ребра
        uint current = 1; // Текущий номер узла
        bool flagAddNode = false;
        bool flagAddArrow = false;
        uint r = 15; // Радиус узла
        uint selectedNode = 0; // Узел из которого начинается ребро

        public Form1()
        {
            InitializeComponent();
            nodes = new Dictionary<uint, Node>();
            arrows = new List<Arrow>();
            comboBox1.SelectedIndex = 0; // По умолчанию выбран ориентированный граф
            button2.Enabled = false;
            string connectionString = @"Data Source=.\SQLEXPRESS;Initial
            Catalog=PW;Integrated Security=True";
            connection = new SqlConnection(connectionString);
            connection.Open();
        }

        private void drawGraph()
        {// Отрисовка структуры данных
            Graphics gr = pictureBox1.CreateGraphics();
            foreach (var a in arrows)
            {// Отрисовка всех ребер
                if(a.NodeIn != null && a.NodeOut != null)
                {
                    Pen pen = new Pen(Brushes.Black, 5);
                    int beginX = a.NodeIn.X, beginY = a.NodeIn.Y;
                    // Для деревьев и ориентированного графа ребра оиентированы
                    if (comboBox1.SelectedIndex == 0 || comboBox1.SelectedIndex == 2)
                        pen.EndCap = System.Drawing.Drawing2D.LineCap.ArrowAnchor;
                    // Если затрачиваемые ресурсы отсутствуют, то ребро пунктирное
                    if (a.Resource == "")
                        pen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;
                    if (a.NodeIn == a.NodeOut) // Если петля
                        gr.DrawArc(pen, beginX - 2 * r, beginY - 2 * r,
                        2 * r, 2 * r, 90, 270);
                    else
                    {
                        int end = a.NodeOut.X - (int)r;
                        if (a.NodeOut.X < beginX)
                            end = a.NodeOut.X + (int)r;
                        gr.DrawLine(pen, beginX, beginY, end, a.NodeOut.Y);
                    }
                }
```

```
        }
        foreach (var n in nodes)
        {// Отрисовка всех узлов
            int x = n.Value.X, y = n.Value.Y;
            gr.FillEllipse(Brushes.White, (x - r), (y - r), 2 * r, 2 * r);
            gr.DrawEllipse(Pens.Black, (x - r), (y - r), 2 * r, 2 * r);
            gr.DrawString(n.Key.ToString(), new Font("Arial", 15),
            Brushes.Black, new Point(x - 9, y - 11));
        }
    }

    private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
    {// В поле "Время" можно вводить только числа и использовать Backspace
        if (!Char.IsDigit(e.KeyChar) && e.KeyChar != 8)
            e.Handled = true;
    }

    private void button1_Click(object sender, EventArgs e)
    {// Добавить событие
        comboBox1.Enabled = false; // Больше нельзя редактировать тип графа
        button4.Enabled = false; // Больше нельзя загрузить структуру данных
        button2.Enabled = false;
        if (flagAddNode)
        {
            label5.Text = "Добавьте узел на поле";
            return;
        }
        nodes.Add(current++, new Node(textBox1.Text));
        flagAddNode = true;
    }

    private void button2_Click(object sender, EventArgs e)
    {// Добавить работу
        comboBox1.Enabled = false; // Больше нельзя редактировать тип графа
        button1.Enabled = false;
        if (flagAddArrow)
        {
            label5.Text = "Добавьте ребро на поле";
            return;
        }
        uint time = 0;
        if (textBox3.Text != "")
            time = Convert.ToUInt32(textBox3.Text);
        arrows.Add(new Arrow(textBox2.Text, time, textBox4.Text));
        flagAddArrow = true;
    }

    private void pictureBox1_MouseClick(object sender, MouseEventArgs e)
    {// Клик по полю
        label5.Text = "Ошибки";
        listBox1.Items.Clear();
        if (!flagAddNode && !flagAddArrow)
        {// Информация о событии или работе
            foreach (var n in nodes)
            {
                if (n.Value.Selected(e.X, e.Y))
                {
                    listBox1.Items.Add("Событие №" + n.Key.ToString());
                    listBox1.Items.Add(n.Value.State);
                    return;
                }
            }
            foreach (var a in arrows)
            {
```

```
                if (a.Selected(e.X, e.Y))
                {
                    uint input = nodes.FirstOrDefault(x => x.Value == a.NodeIn).Key;
                    uint output = nodes.FirstOrDefault(x =>
                    x.Value == a.NodeOut).Key;
                    listBox1.Items.Add("Работа из " + input.ToString() +
                    " в " + output.ToString());
                    listBox1.Items.Add("Действие " + a.Action);
                    listBox1.Items.Add("Время(мин) " + a.Time);
                    listBox1.Items.Add("Ресурсы " + a.Resource);
                    return;
                }
            }
        }
        if (flagAddNode)
        {// Отрисовка события
            nodes.Values.Last().Coordinates(e.X, e.Y);
            flagAddNode = false;
            if (current > 2) // Нельзя добавлять ребра пока узлов меньше 2
                button2.Enabled = true;
        }
        if (flagAddArrow)
        {// Отрисовка работы
            if (selectedNode == 0)
            {// Выбор выходного узла ребра
                foreach (var n in nodes)
                {
                    if (n.Value.Selected(e.X, e.Y))
                    {
                        selectedNode = n.Key;
                        break;
                    }
                }
                return; // Первая вершина выбрана
            }
            foreach (var n in nodes)
            {// Выбор входящего узла ребра
                if (n.Value.Selected(e.X, e.Y))
                {
                    if (comboBox1.SelectedIndex == 2)
                    {// Если структура данных дерево
                        if (n.Key == 1)
                        {
                            label5.Text = "В корень дерева ребра не входят";
                            return;
                        }
                        if (n.Value.Input || n.Key == selectedNode)
                        {
                            label5.Text = "В дереве не может быть циклов или петель";
                            return;
                        }
                        n.Value.Input = true;
                    }
                    arrows.Last().InOut(nodes[selectedNode], n.Value);
                    selectedNode = 0;
                    flagAddArrow = false;
                    button1.Enabled = true;
                    break;
                }
            }
        }
        drawGraph();
    }
```

```
        private void pictureBox1_MouseDoubleClick(object sender, MouseEventArgs e)
        {// При двойном клике выполняется тоже, что и для одного
            pictureBox1_MouseClick(sender, e);
        }

        private void button5_Click(object sender, EventArgs e)
        {// Кнопка отмены добавления
            listBox1.Items.Clear();
            if (flagAddNode)
            {// Отмена узла
                button2.Enabled = true;
                nodes.Remove(--current);
                flagAddNode = false;
            }
            if (flagAddArrow)
            {// Отмена ребра
                button1.Enabled = true;
                arrows.Remove(arrows.Last());
                flagAddArrow = false;
                selectedNode = 0;
            }
        }

    private void button3_Click(object sender, EventArgs e)
        {// Запись
            if (radioButton1.Checked)
            {// В файл
                using (StreamWriter sw = new StreamWriter("PN.txt",
                false, System.Text.Encoding.Default))
                {
                    sw.WriteLine(comboBox1.SelectedIndex);
                    sw.WriteLine(current - 1);
                    foreach (var n in nodes)
                    {// Запись узлов
                        sw.WriteLine(n.Key);
                        sw.WriteLine(n.Value.State);
                        sw.WriteLine(n.Value.X);
                        sw.WriteLine(n.Value.Y);
                        sw.WriteLine(n.Value.Input);
                    }
                    sw.WriteLine(arrows.Count);
                    foreach (var a in arrows)
                    {// Запись ребер
                        sw.WriteLine(a.Action);
                        sw.WriteLine(a.Time);
                        sw.WriteLine(a.Resource);
                        sw.WriteLine(nodes.FirstOrDefault(x => x.Value == a.NodeIn).Key);
                        sw.WriteLine(nodes.FirstOrDefault(x =>
                        x.Value == a.NodeOut).Key);
                    }
                }
            }
            else
            {// В базу данных
                // Очищение таблиц данных
                string sql = "TRUNCATE TABLE Arrow";
                SqlCommand command = new SqlCommand(sql, connection);
                command.ExecuteNonQuery();
                sql = "TRUNCATE TABLE Node";
                command = new SqlCommand(sql, connection);
                command.ExecuteNonQuery();

                foreach (var n in nodes)
                {// Запись узлов
```

```
                sql = "INSERT Node VALUES ('" + n.Value.State + "', '" + n.Value.X +
                "', '" + n.Value.Y + "', '" + n.Value.Input + "', '" +
                comboBox1.SelectedIndex + "')";
                command = new SqlCommand(sql, connection);
                command.ExecuteNonQuery();
            }
            for (int i = 0; i < arrows.Count; i++)
            {// Запись ребер
                sql = "INSERT Arrow VALUES ('" + arrows[i].Action + "', '" +
                arrows[i].Time + "', '" + arrows[i].Resource + "', '" +
                nodes.FirstOrDefault(x => x.Value == arrows[i].NodeIn).Key + "', '"
                + nodes.FirstOrDefault(x => x.Value == arrows[i].NodeOut).Key + "')";
                command = new SqlCommand(sql, connection);
                command.ExecuteNonQuery();
            }
        }
    }

    private void button4_Click(object sender, EventArgs e)
    {// Загрузка
        if (radioButton1.Checked)
        {// Из файла
            using (StreamReader sr = new StreamReader("PN.txt",
            System.Text.Encoding.Default))
            {
                comboBox1.SelectedIndex = Convert.ToInt32(sr.ReadLine());
                current = Convert.ToUInt32(sr.ReadLine());
                for (uint i = 0; i < current; i++)
                {// Чтение узлов
                    uint key = Convert.ToUInt32(sr.ReadLine());
                    string state = sr.ReadLine();
                    int x = Convert.ToInt32(sr.ReadLine());
                    int y = Convert.ToInt32(sr.ReadLine());
                    bool input = Convert.ToBoolean(sr.ReadLine());
                    nodes.Add(key, new Node(state));
                    nodes.Values.Last().Coordinates(x, y);
                    nodes.Values.Last().Input = input;
                }
                current++;
                uint arrowsSize = Convert.ToUInt32(sr.ReadLine());
                for (uint i = 0; i < arrowsSize; i++)
                {// Чтение ребер
                    string action = sr.ReadLine();
                    uint time = Convert.ToUInt32(sr.ReadLine());
                    string resource = sr.ReadLine();
                    uint nodeIn = Convert.ToUInt32(sr.ReadLine());
                    uint nodeOut = Convert.ToUInt32(sr.ReadLine());
                    arrows.Add(new Arrow(action, time, resource));
                    arrows.Last().InOut(nodes[nodeIn], nodes[nodeOut]);
                }
            }
        }
        else
        {// Из базы данных
            string sql = "SELECT * FROM Node";
            SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);
            DataSet ds = new DataSet();
            adapter.Fill(ds);
            DataTable dt = ds.Tables[0];

            comboBox1.SelectedIndex = Convert.ToInt32(dt.Rows[0][5]);
            current = Convert.ToUInt32(dt.Rows.Count);
            for (int i = 0; i < current; i++)
            {// Чтение узлов
```

```
                uint key = Convert.ToUInt32(dt.Rows[i][0]);
                string state = dt.Rows[i][1].ToString();
                int x = Convert.ToInt32(dt.Rows[i][2]);
                int y = Convert.ToInt32(dt.Rows[i][3]);
                bool input = Convert.ToBoolean(dt.Rows[i][4]);
                nodes.Add(key, new Node(state));
                nodes.Values.Last().Coordinates(x, y);
                nodes.Values.Last().Input = input;
            }
            current++;

            sql = "SELECT * FROM Arrow";
            adapter = new SqlDataAdapter(sql, connection);
            ds = new DataSet();
            adapter.Fill(ds);
            dt = ds.Tables[0];

            int arrowsSize = Convert.ToInt32(dt.Rows.Count);
            for (int i = 0; i < arrowsSize; i++)
            {// Чтение ребер
                string action = dt.Rows[i][1].ToString();
                uint time = Convert.ToUInt32(dt.Rows[i][2]);
                string resource = dt.Rows[i][3].ToString();
                resource = resource.Trim();
                uint nodeIn = Convert.ToUInt32(dt.Rows[i][4]);
                uint nodeOut = Convert.ToUInt32(dt.Rows[i][5]);
                arrows.Add(new Arrow(action, time, resource));
                arrows.Last().InOut(nodes[nodeIn], nodes[nodeOut]);
            }
        }
        comboBox1.Enabled = false;
        if (current > 2)
            button2.Enabled = true;
        button4.Enabled = false;
        drawGraph();
    }
}
   }
```

Листинг 3 – файл Form1.cs

4. Реализация программы

Интерфейс программы выглядит следующим образом. В нём имеется форма под описание создаваемого события (узла, вершины), кнопка “Добавить”, которая даёт возможность разместить на рабочем поле свою вершину по щелчку мыши (в окне построения графа), форма под описание работы, которая имеет описание действия, затраченное время на это действие (в минутах), а также ресурсы, которые имеются не всегда. Если пользователь не заполнит поле с ресурсами, то работа графически будет представлена пунктирной прямой. Соответственно имеется кнопка “Добавить” для работы

(стрелки). Кнопка “Отменить” отменяет действие добавления события и работы.

Имеется радиобаттон для сохранения файла разными способами, соответственно в виде txt файла и в виде базы данных. И две кнопки сохранить и загрузить, которые записывают новые данные и загружают.

Боковое меню заканчивается информационным окном, которое выводит информацию о ранее созданных событиях и происходящими между ними действиями. А ещё чуть ниже поле для вывода ошибок.

В левой же части экрана имеется выпадающий список, где пользователь сможет выбрать тип построения графа (ориентированный, неориентированный граф или дерево).

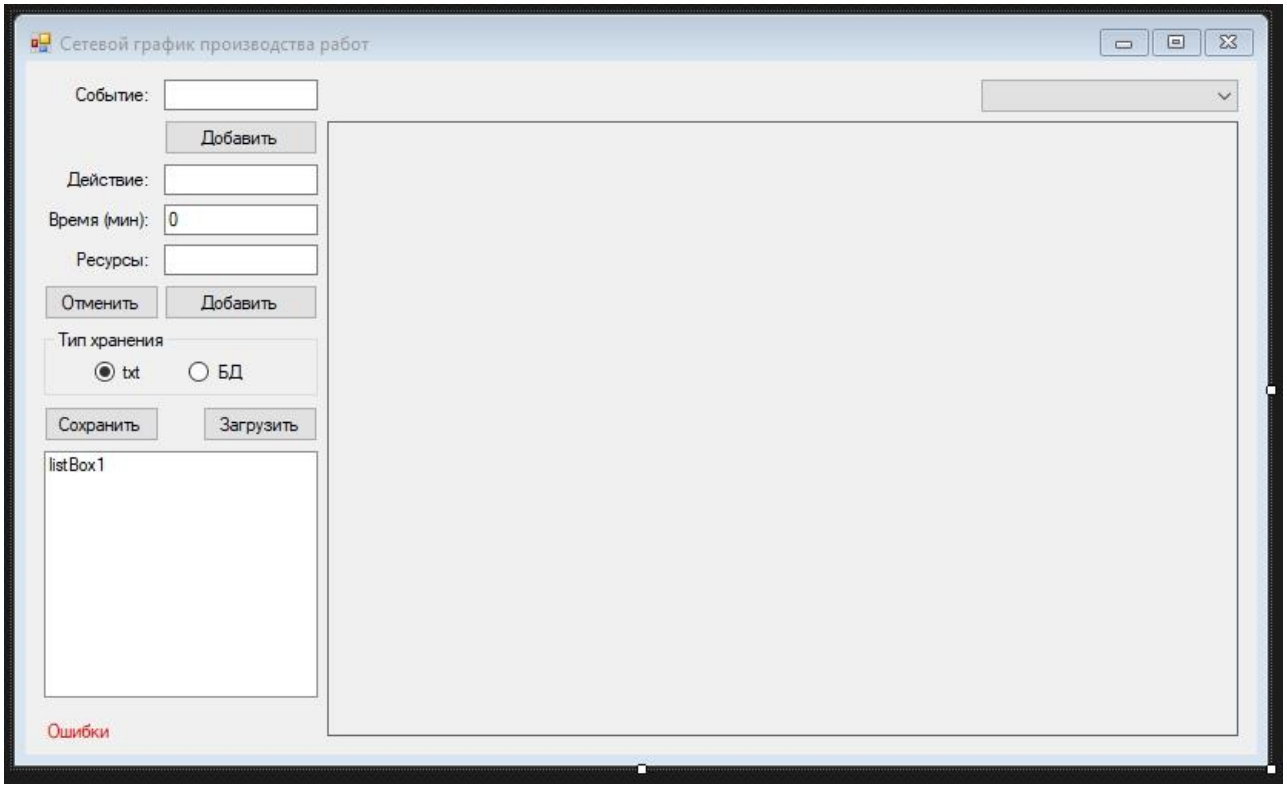

Рисунок 1 – Созданный интерфейс

При включении приложения тип построения графов является ориентированным. И изначально кнопка добавления работы (действие, выраженное виде стрелки или прямой) недоступна, пока на поле не появится минимум два события (узла, вершины).

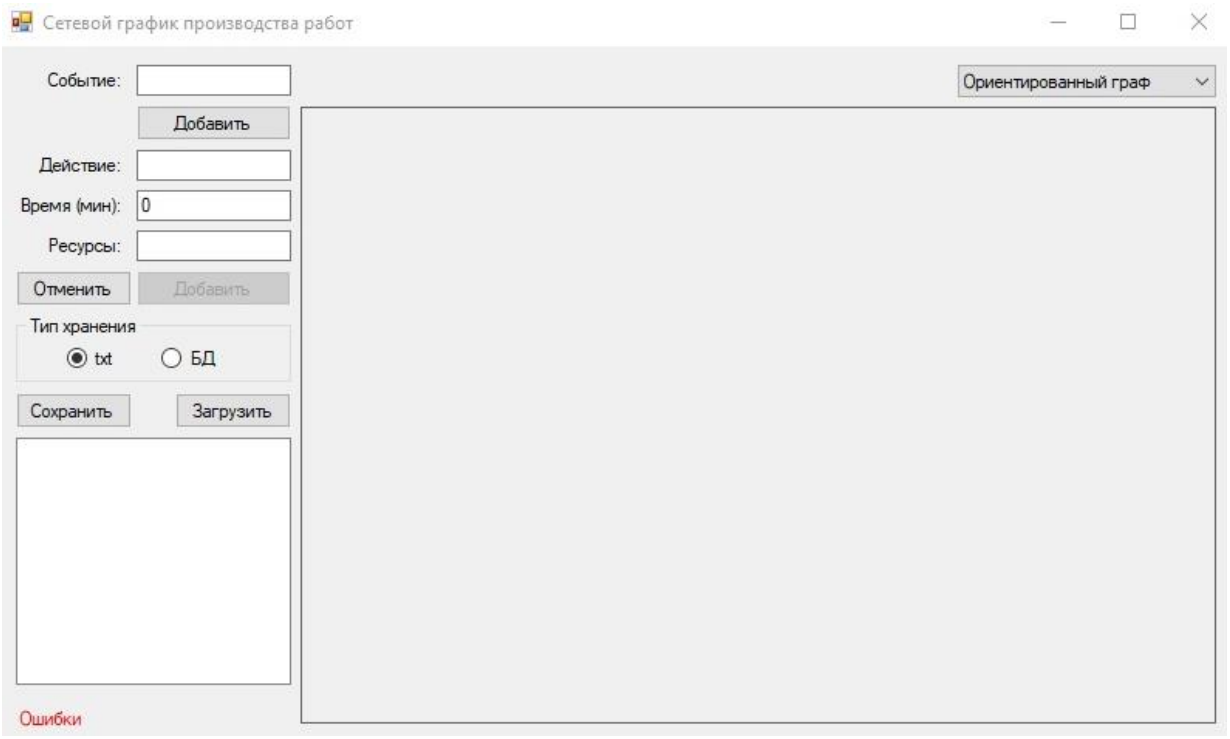

Рисунок 2 – Интерфейс при включении

При добавлении первой вершины смена типа графа перестаёт быть доступна.

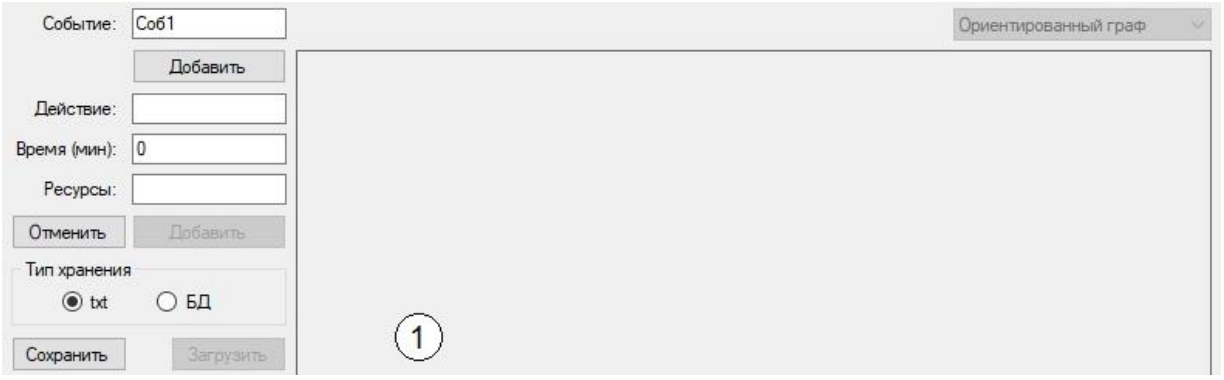

Рисунок 3 – Построение первой вершины

При создании ребра если не заполнять пункт с ресурсами, то размещённая полоса будет отображаться пунктиром. Также при добавлении ребра соответственно блокируется кнопка добавления узла, а при добавлении узла блокируется добавление ребра.

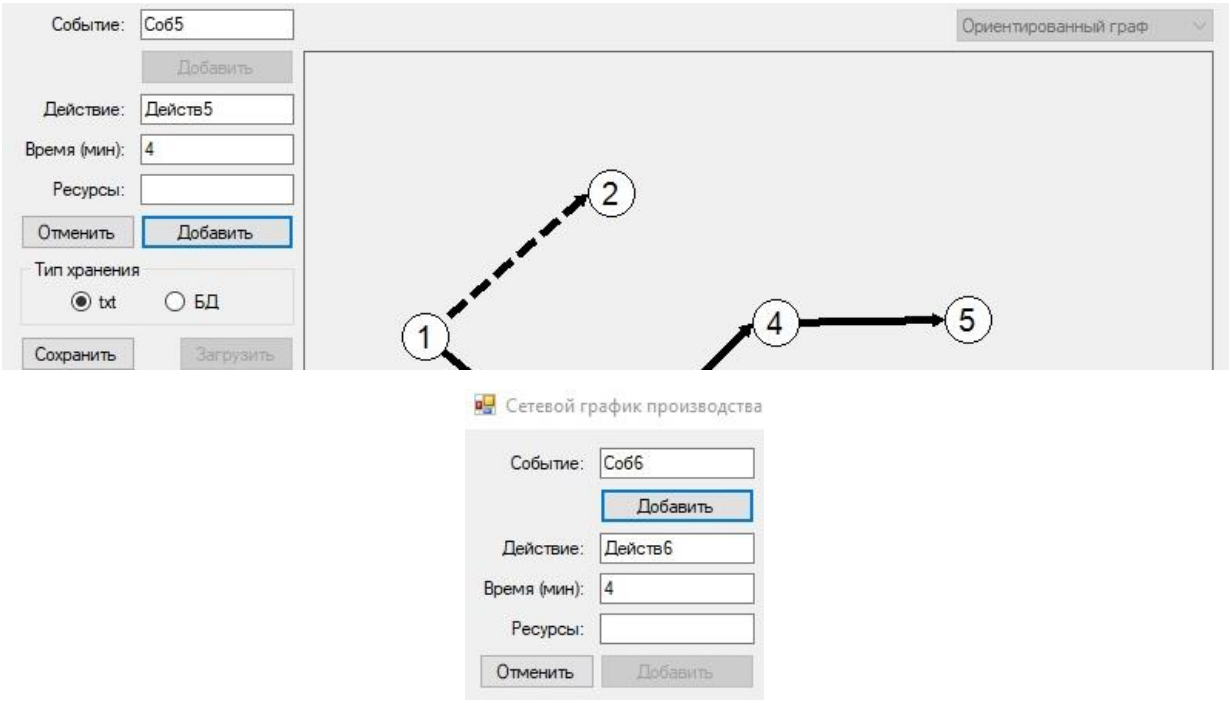

Рисунок 4 – Построение рёбер и блокировка кнопок добавления

При добавлении ребра нельзя добавить следующее пока не будет размещено предыдущее иначе будет выдана ошибка, подобная ошибка имеется и при добавлении вершины.

Рисунок 5 – Ошибки при попытке повторного добавления

После каждого ново-поставленного события или работы можно щёлкнуть по созданному элементу и увидеть о нём информацию. Так информация об узле, сплошном ребре, пунктирном ребре, и о пунктирной петле соответственно выглядит следующим образом (Соб4 – это описание, вводимое при создании узла 4, а Событие №4 – автоматически выводимое значение данной вершины, так и для рёбер Работа из … в … - это автоматически выводимые значения, а остальные данные после соответствующих слов являются вводимыми при создании).

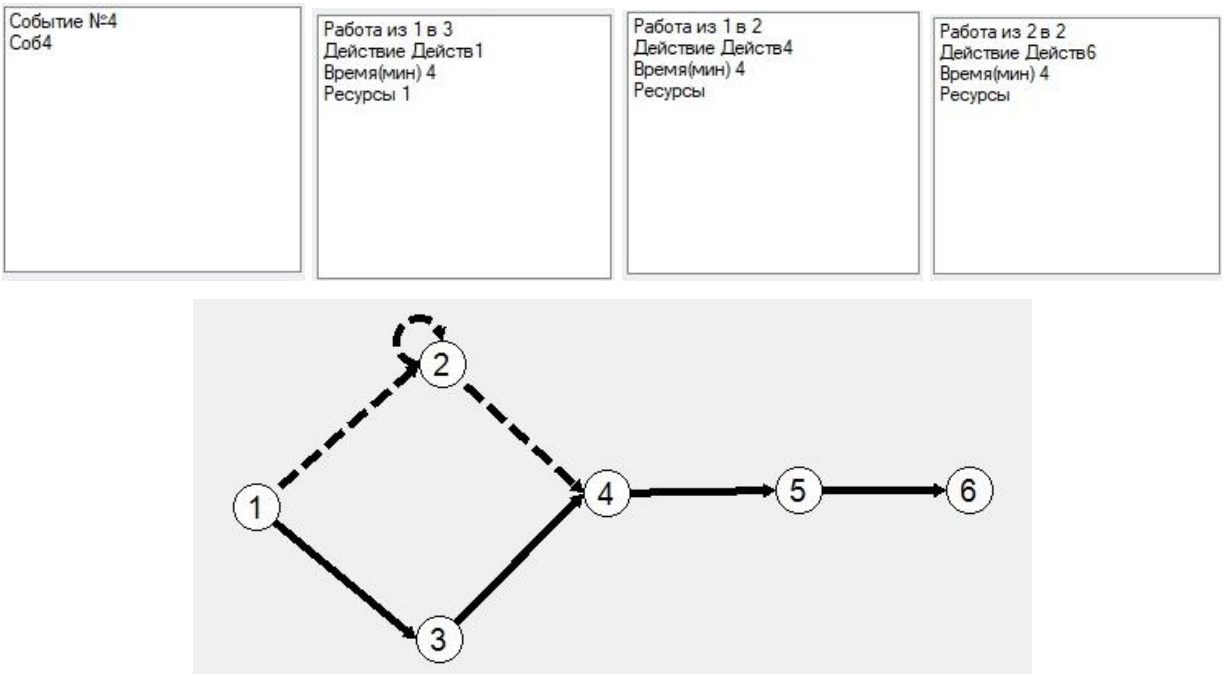

Рисунок 6 – Информационные поля и граф, к которому они относятся

Чтобы сохранить построенный граф в виде txt, необходимо выбрать соответствующее значение в радиобаттоне и нажать на кнопку сохранить. Таким образом часть сохранённого файла в txt имеет следующий вид.

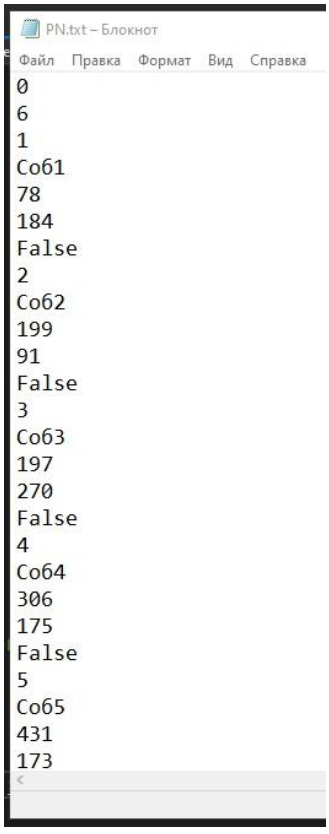

Рисунок 7 – Часть сохранённого txt файла

Чтобы сохранить построенный граф в виде БД, необходимо выбрать соответствующее значение в радиобаттоне и нажать на кнопку сохранить. Таким образом сохранённый файл в базе данных имеет следующий вид для вершин и рёбер соответственно.

	ID_Node	State	X	Y	Input	Type
▶	1	Со61 ...	78	184	False	0
	2	Со62 ...	199	91	False	0
	3	Со63 ...	197	270	False	0
	4	Со64 ...	306	175	False	0
	5	Со65 ...	431	173	False	0
	6	Со66 ...	542	173	False	0
*	NULL	NULL	NULL	NULL	NULL	NULL

	ID_Arrow	Action	Time	Resource	ID_NodeIn	ID_NodeOut
▶	1	Действ1 ...	4	1 ...	1	3
	2	Действ2 ...	4	2 ...	3	4
	3	Действ3 ...	4	3 ...	4	5
	4	Действ4 ...	4	...	1	2
	5	Действ5 ...	4	...	2	4
	6	Действ6 ...	4	...	2	2
	7	Действ7 ...	10	2 ...	5	6
*	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 8 – Таблица Node и Arrow в базе данных

Выбираемый тип графа влияет на наличие стрелки и на проверку наличия циклов, как в случае с построениями деревьев.

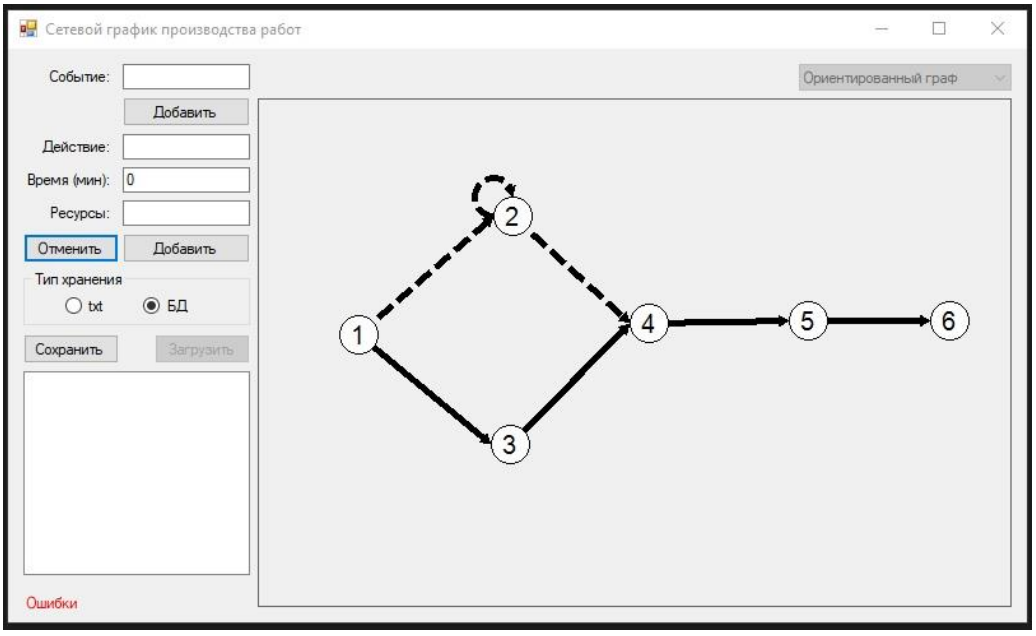

Рисунок 9 – Ориентированный граф

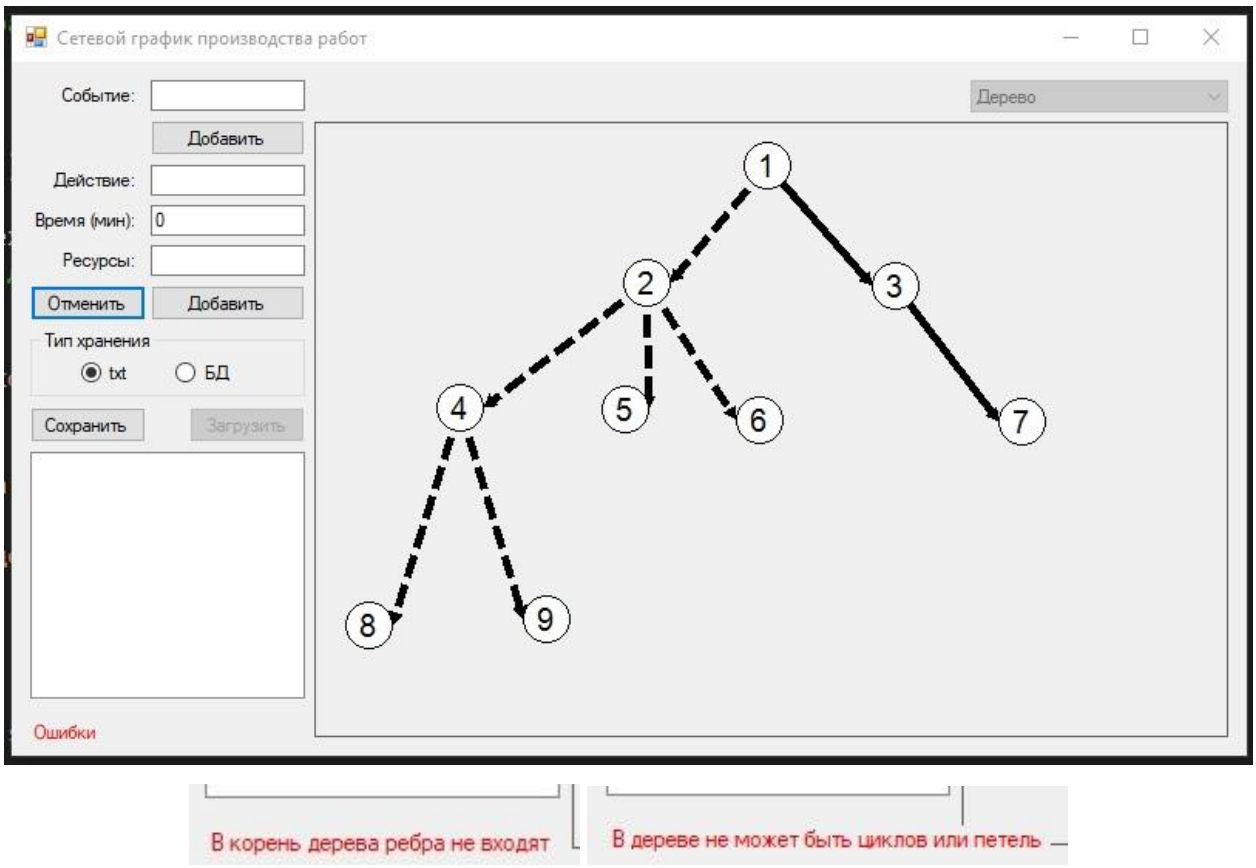

Рисунок 10 – Дерево и возможные ошибки при построении

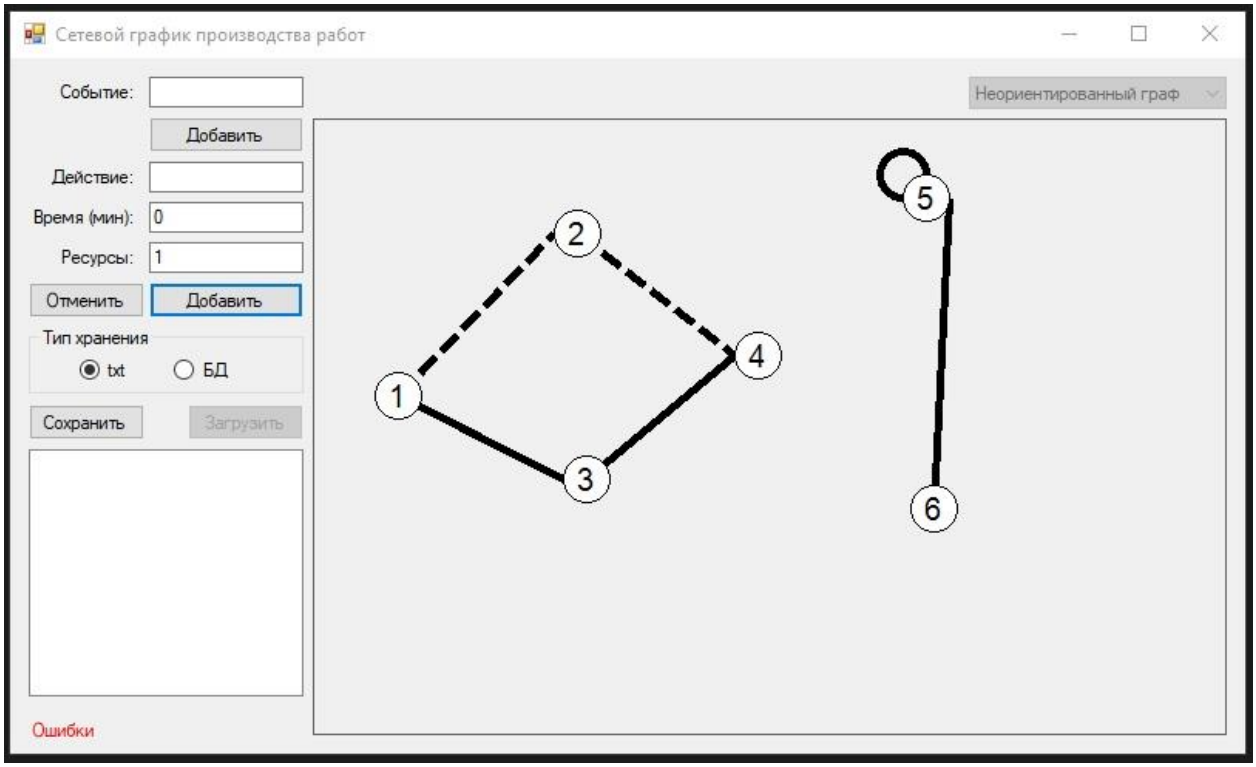

Рисунок 11 – Неориентированный граф

Заключение

В ходе шестой лабораторной работы была создана программа, с вводом, сохранением, загрузкой и отображением данных, представленных в виде графа. Пользователь имеет возможность ввести ориентированный и неориентированный граф, деревья, также присутствуют различные варианты сохранения. Программа была протестирована для проверки на корректность.

Используемая литература

1. Сетевой график – https://ru.wikipedia.org/wiki/Сетевой_график
2. Граф (математика) – https://ru.wikipedia.org/wiki/Граф_(математика)
3. Работа с графами онлайн – https://graphonline.ru
4. Программа для построения графов C# – https://vscode.ru/prog-lessons/programma-dlya-postroeniya-grafov.html