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

по дисциплине

Технологии программирования

Вариант 8

РУКОВОДИТЕЛЬ:

_____ Капранов С.Н.

(подпись) (фамилия,и.,о.)

СТУДЕНТ:

_____ Кулагина К.А.

(подпись) (фамилия,и.,о.)

18-ИСТ-2

(шифр группы)

Работа защищена«____»_____

С оценкой _____

Нижний Новгород

2020

Задание:

Молекулярная структура веществ.

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.

Листинг (основной код, Form1):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.Drawing.Imaging;
using System.Runtime.Serialization.Formatters.Binary;
using System.Xml.Serialization;
using System.Globalization;
using System.Collections;

namespace StructureOfMolecules
{
   public partial class Form1 : Form
   {
      Form2 form2;

      Graphics g;
      // Список вершин
      Vertexes V = new Vertexes();
      // Список рёбер
      Edges E = new Edges();

      // Список для хранения вершин и рёбер
      ListEdgesAndVertexes LEV = new ListEdgesAndVertexes();


      // Радиус окружности вершины
      int R = 20;

      Point FirstCoordinate = default(Point);
      Point SecondCoordinate = default(Point);
```

```
        int color = 178034034; // DimGray
        int ColorForList = 178034034;

        ColorDialog colorDialog = new ColorDialog();

        public Form1()
        {
            InitializeComponent();
            g = field.CreateGraphics();
            form2 = new Form2();
        }

        // Добавление вершины
        public void AddVertex()
        {
            for (int i = 0; i < V.VertexesList.Count; i++)
            {
                Pen pen = new Pen(Color.FromArgb(V.VertexesList[i].ColorVertex));
                g.FillEllipse(new
SolidBrush(Color.FromArgb(V.VertexesList[i].ColorVertex)), V.VertexesList[i].X -
R, V.VertexesList[i].Y - R, 2 * R, 2 * R);
                g.DrawEllipse(pen, V.VertexesList[i].X - R, V.VertexesList[i].Y - R, 2 *
R, 2 * R);
            }

        }

        // Вызывает Form2 для того чтобы присвоить имя
        private void ChangeText()
        {
            for (int i = 0; i < V.VertexesList.Count; i++)
            {
                const float fontSize = 10f;
                g.DrawString(V.VertexesList[i].Name, new Font("Times New Roman",
fontSize, FontStyle.Bold), new SolidBrush(Color.Black), V.VertexesList[i].X - R +
50, V.VertexesList[i].Y - R);
            }
        }

        // Добавление ребра
        private void AddEdge()
        {
            for (int i = 0; i < E.EdgesList.Count; i++)
            {
                Pen pen = new Pen(Color.FromArgb(E.EdgesList[i].ColorEdge), 5);
                g.DrawLine(pen, E.EdgesList[i].v1, E.EdgesList[i].v2);
```

```
            }
        }

        // Поиск вершин для рёбер
        private Vertex ChooseVertexForEdge(int x, int y)
        {
            var distance = CheckDisnace(x, y);
            for (int i = 0; i < V.VertexesList.Count; i++)
            {
                if (Math.Sqrt(Math.Pow(x - V.VertexesList[i].X, 2) + Math.Pow(y -
V.VertexesList[i].Y, 2)) ==  distance)
                {
                    return V.VertexesList[i];
                }
            }
            return V.VertexesList[0];
        }

        // Поиск вершины
        public Vertex SearchVertex(int x, int y)
        {
            for (int i = 0; i < V.VertexesList.Count; i++)
            {
                if (Math.Sqrt(Math.Pow(x - V.VertexesList[i].X, 2) + Math.Pow(y -
V.VertexesList[i].Y, 2)) < R)
                {
                    return V.VertexesList[i];
                }
            }
            return V.VertexesList[0];
        }

        // Проверка, чтобы круги не рисовались друг на друге
        public double CheckDisnace(int x, int y)
        {
            var checkdistance = new List<double>();
            for (int i = 0; i < V.VertexesList.Count; i++)
            {
                checkdistance.Add(Math.Sqrt(Math.Pow(x - V.VertexesList[i].X, 2) +
Math.Pow(y - V.VertexesList[i].Y, 2)));
            }
            checkdistance.Sort();

            return checkdistance[0];
        }

        // Очистка поля
```

```
        private void ClearField()
        {
            g.Clear(Color.FromArgb(255, 255, 255));
        }

        // Кнопка добавления вершины
        private void AddVertexButton_Click_1(object sender, EventArgs e)
        {
            AddVertexButton.Enabled = false;
            AddEdgeButton.Enabled = true;
        }

        // Кнопка добавления ребра
        private void AddEdgeButton_Click_1(object sender, EventArgs e)
        {
            AddVertexButton.Enabled = true;
            AddEdgeButton.Enabled = false;
        }

        // Очистка поля, также очистка списков
        private void ClearAll_Click(object sender, EventArgs e)
        {
            const string message = "Are you sure?";
            const string header = "Delete";
            var Clear = MessageBox.Show(message, header,
MessageBoxButtons.YesNoCancel);
            if (Clear == DialogResult.Yes)
            {
                V.VertexesList.Clear();
                E.EdgesList.Clear();
                File.Delete("List.xml");
                ClearField();
            }
        }

        // Сериализация
        private void Serialize()
        {
                XmlSerializer xml = new XmlSerializer(typeof(ListEdgesAndVertexes));
                using (FileStream fileStream = new
FileStream(@"D:\TP\Algorithms_data_structures_IST2\Algorithms_data_structures
_IST2\laba 6\StructureOfMolecules\StructureOfMolecules\bin\Debug\List.xml",
FileMode.OpenOrCreate))
                {
                    xml.Serialize(fileStream, LEV);
                }
```

```
        }

        // Десериализация
        private ListEdgesAndVertexes Deserialize()
        {

            XmlSerializer xml = new XmlSerializer(typeof(ListEdgesAndVertexes));
                using (FileStream fileStream = File.OpenRead("List.xml"))
                {
                    return (ListEdgesAndVertexes)xml.Deserialize(fileStream);
                }
        }

        // Копка сохранения
        private async void SaveButton_Click(object sender, EventArgs e)
        {
            LEV = new ListEdgesAndVertexes(E, V);
            Serialize();
        }

        // Кнопка загрузки
        private void DownloadButton_Click(object sender, EventArgs e)
        {
            V = new Vertexes();
            E = new Edges();
            V = Deserialize().VertexesList;
            E = Deserialize().EdgesList;
            AddVertex();
            AddEdge();
            ChangeText();
        }

        // Смена цвета
        private void ResultingColor()
        {
            color = colorDialog.Color.ToArgb(); // Меняем цвет

            if (colorDialog.ShowDialog() == DialogResult.OK) //Если окно закрылось
с OK, то меняем цвет для Pen
            {
                color = colorDialog.Color.ToArgb();
            }
        }

        // Label с цветом, который вызывает функцию ResultingColor()
        private void ChangeColor_Click(object sender, EventArgs e)
        {
```

```
            ResultingColor();
        }

        // Работа с полем
        private void field_MouseClick(object sender, MouseEventArgs e)
        {
            // Добавление на поле вершины
            // Добавляет координаты вершины и присовенный цвет в список,
вызывает функцию добавления вершины
            if (AddVertexButton.Enabled == false)
            {
                if (e.Button == MouseButtons.Left)
                {
                    if (V.VertexesList.Count == 0 || CheckDisnace(e.X, e.Y) > 2 * R)
                    {
                        V.VertexesList.Add(new Vertex(e.X, e.Y, color, " "));
                        AddVertex();
                    }
                }
            }

            // Добавление текста
            if (e.Button == MouseButtons.Right && V.VertexesList.Count > 0)
            {
                if (CheckDisnace(e.X, e.Y) < R)
                {
                    for (int i = 0; i < V.VertexesList.Count; i++)
                    {
                        if (V.VertexesList[i] == SearchVertex(e.X, e.Y))
                        {
                            form2.ShowDialog();
                            V.VertexesList[i].Name = form2.nameText;
                            ChangeText();
                        }
                    }
                }
            }

            // Добавление на поле ребра
            if (AddEdgeButton.Enabled == false)
            {
                if (e.Button == MouseButtons.Left)
                {
                    if (FirstCoordinate.X == default(Point).X && FirstCoordinate.Y ==
default(Point).Y)
                    {
                        FirstCoordinate.X = ChooseVertexForEdge(e.X, e.Y).X;
```

```
                    FirstCoordinate.Y = ChooseVertexForEdge(e.X, e.Y).Y;
                    ColorForList = ChooseVertexForEdge(e.X, e.Y).ColorVertex;
                    return;
                }

                if (ColorForList == ChooseVertexForEdge(e.X, e.Y).ColorVertex)
                {
                    SecondCoordinate.X = ChooseVertexForEdge(e.X, e.Y).X;
                    SecondCoordinate.Y = ChooseVertexForEdge(e.X, e.Y).Y;
                    E.EdgesList.Add(new Edge(FirstCoordinate, SecondCoordinate,
ColorForList));
                    FirstCoordinate = default(Point);
                    SecondCoordinate = new Point();
                    AddEdge();
                }
                else
                {
                    FirstCoordinate = default(Point);
                    SecondCoordinate = new Point();
                    AddEdge();
                }
            }
        }
    }
}
```