Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Российский университет транспорта»

(ФГАОУ ВО РУТ(МИИТ), РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа № 16

по дисциплине: «Программирование и основы алгоритмизации»

на тему: «Освоение навыков программирования инкапсулированных классов»

Выполнил: ст. гр. ТУУ-111

Кругликов Е.А.

Вариант №11

Проверил: к.т.н., доц. Сафронов А.И.

1. **Формулировка цели работы**

Разработать приложение с инкапсулированным классом.

Индивидуальные задачи на инкапсуляцию необходимо решать в режиме разработки «Windows Forms Application».

Решение задач должно сводиться к максимально инкапсулированной реализации указанного класса.

При решении задач инкапсуляции необходимо сосредоточиться на составлении свойств и методов для классов и для экземпляров классов, которые предоставляют защищённый доступ к полям классов.

1. **Описание задачи**

**Вариант 11.**

Создайте класс «точка в полярных координатах».

Все поля класса инкапсулированы в классе. При создании экземпляра класса задаются координаты точки на плоскости в Декартовой системе координат. Предоставить возможность пользователю для изменения координат в каждом созданном экземпляре (get; set).

Каждая точка должна быть в состоянии отвечать соответствующими значениями модуля вектора (get) и угла поворота вектора (get) в полярной системе координат.

Иметь возможность хранения экземпляров класса в расширяемом (по мере возникновения необходимости) массиве. Предоставить возможность удаления указанных в экранном списке экземпляров класса «точка в полярных координатах», а также возможность очистки содержимого массива по нажатии на соответствующую экранную кнопку.

1. **Составление диаграммы классов, входящих в состав решения**

- int x

- int y

\_Point

+ string ToString()

+ double VectorModule

+ string VectorAngle

Рисунок 1 – Диаграмма классов, входящих в состав решения

1. **Сети Петри запрограммированного технологического процесса**

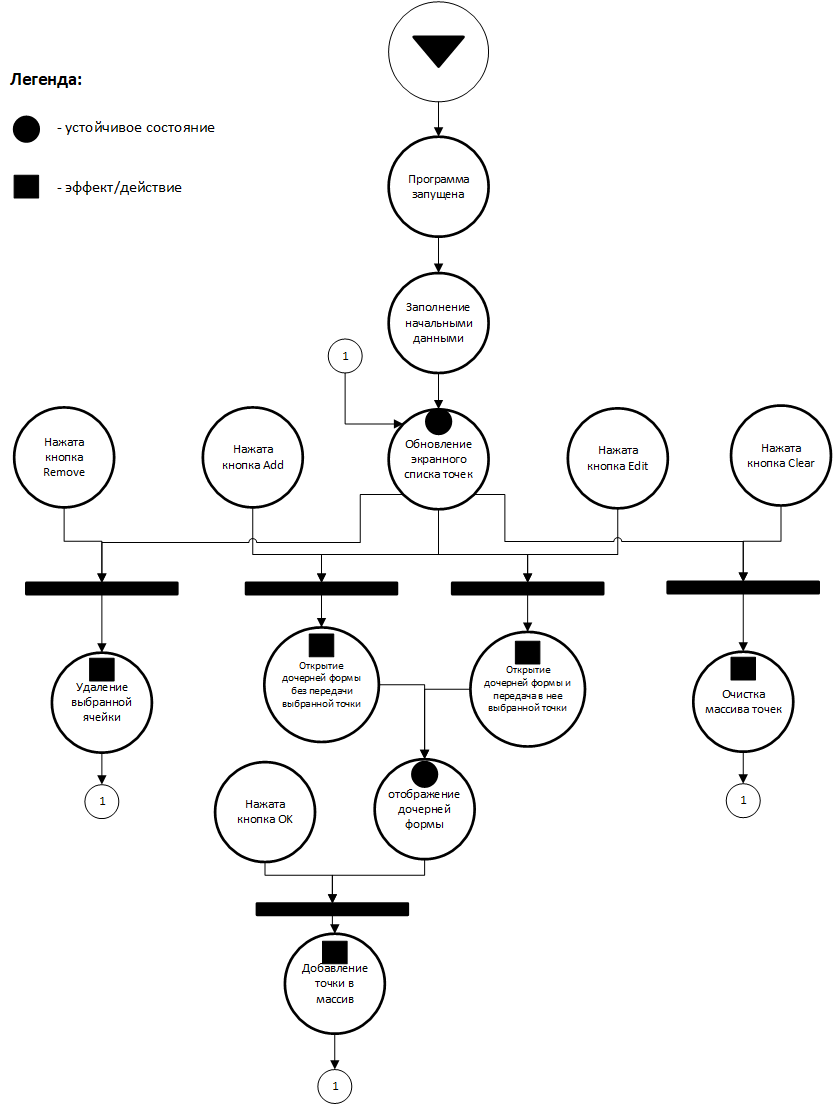


Рисунок 2 – Сеть Петри запрограммированного технологического процесса

1. **Схемы алгоритмов методов в составе решения, отмеченных на сети Петри в качестве «эффектов» (метка )**

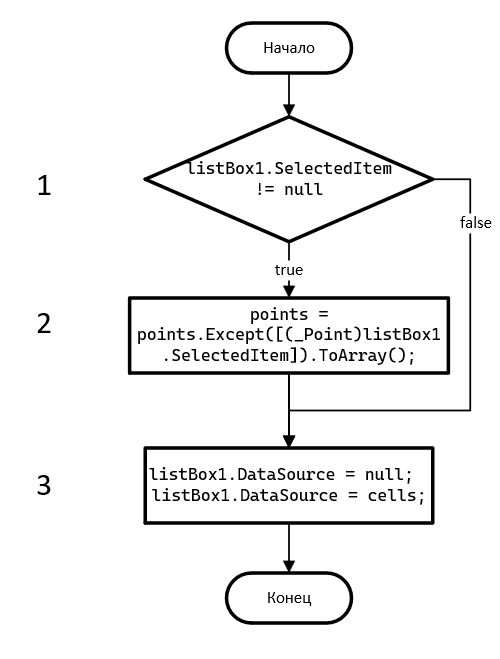


Рисунок 3 – Удаление выбранной точки

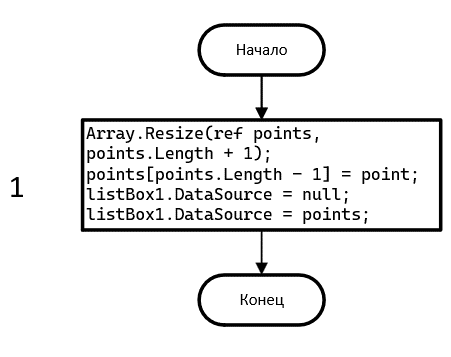


Рисунок 4 – Открытие дочерней формы без параметра

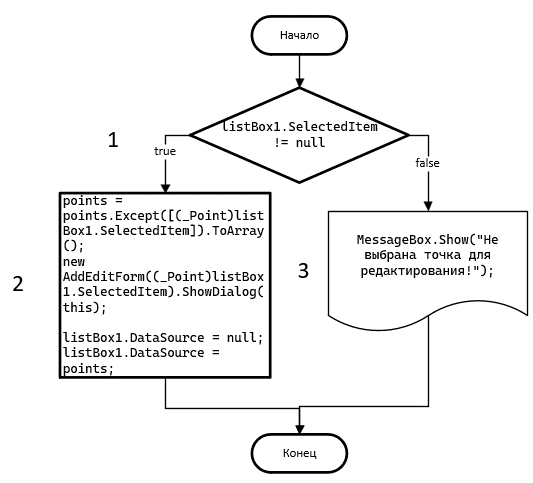


Рисунок 5 – Открытие дочерней формы с параметром выбранной ячейки

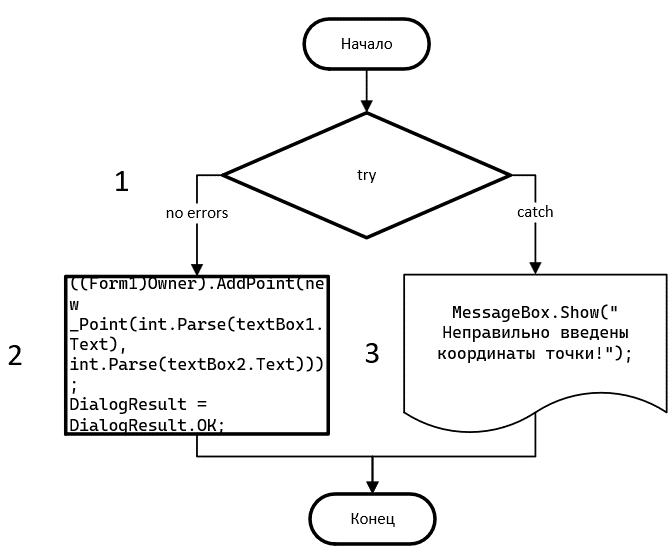


Рисунок 6 – Добавление ячейки в список

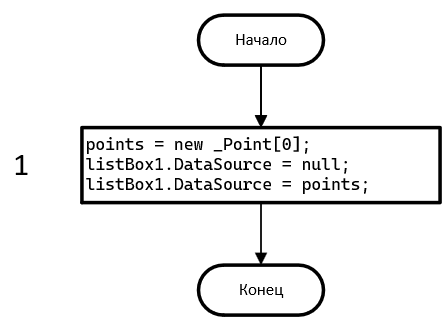


Рисунок 7 – Добавление ячейки в список

1. **Подбор тестовых примеров.**

В качестве тестовых данных выбраны четыре точки:

1. X – 0;

Y – 1;

1. X – 1;

Y – 0;

1. X – 1;

Y – 1;

1. X – 2;

Y – 3;

1. **Листинг (код) составленного программного обеспечения.**

**7.1 Файл Form1.cs**

namespace Points

{

public partial class Form1 : Form

{

\_Point[] points = new \_Point[0];

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

AddPoint(new \_Point(0, 1));

AddPoint(new \_Point(1, 0));

AddPoint(new \_Point(1, 1));

AddPoint(new \_Point(2, 3));

listBox1.DataSource = points;

}

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

{

new AddEditForm().ShowDialog(this);

}

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

{

if (listBox1.SelectedItem != null)

{

points = points.Except([(\_Point)listBox1.SelectedItem]).ToArray();

new AddEditForm((\_Point)listBox1.SelectedItem).ShowDialog(this);

listBox1.DataSource = null;

listBox1.DataSource = points;

}

else

MessageBox.Show("Не выбрана точка для редактирования!");

}

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

{

if (listBox1.SelectedItem != null)

{

points = points.Except([(\_Point)listBox1.SelectedItem]).ToArray();

}

listBox1.DataSource = null;

listBox1.DataSource = points;

}

public void AddPoint(\_Point point)

{

Array.Resize(ref points, points.Length + 1);

points[points.Length - 1] = point;

listBox1.DataSource = null;

listBox1.DataSource = points;

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyData == Keys.F1)

{

new About().ShowDialog(this);

}

}

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

{

points = new \_Point[0];

listBox1.DataSource = null;

listBox1.DataSource = points;

}

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

{

if (listBox1.SelectedValue != null)

{

\_Point p = (\_Point)listBox1.SelectedValue;

MessageBox.Show($"{p.VectorModule}", "Vector module");

}

else

MessageBox.Show("Не выбрана точка!");

}

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

{

if (listBox1.SelectedValue != null)

{

\_Point p = (\_Point)listBox1.SelectedValue;

MessageBox.Show($"{p.VectorAngle}", "Vector angle");

}

else

MessageBox.Show("Не выбрана точка!");

}

}

public class \_Point

{

public int x { get; private set; }

public int y { get; private set; }

public double VectorModule { get => Math.Sqrt(x \* x + y \* y); }

public double VectorAngle { get => Math.Round(Math.Acos(x / VectorModule) \* (180 / Math.PI), 2); }

public \_Point(int x, int y)

{

this.x = x;

this.y = y;

}

public override string ToString()

{

return $"({x}; {y})";

}

}

}

**7.2 Файл AddEditForm.cs**

namespace Points

{

public partial class AddEditForm : Form

{

public AddEditForm()

{

InitializeComponent();

}

public AddEditForm(\_Point point)

{

InitializeComponent();

textBox1.Text = point.x.ToString();

textBox2.Text = point.y.ToString();

}

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

{

try

{

((Form1)Owner).AddPoint(new \_Point(int.Parse(textBox1.Text), int.Parse(textBox2.Text)));

DialogResult = DialogResult.OK;

}

catch

{

MessageBox.Show("Входная строка имела неверный формат!");

}

}

}

}

**7.3 Файл About.cs**

namespace Points

{

public partial class About : Form

{

public About()

{

InitializeComponent();

label1.Text = File.ReadAllText("info.txt");

}

}

}

1. **Графический пользовательский интерфейс программного обеспечения и его описание**

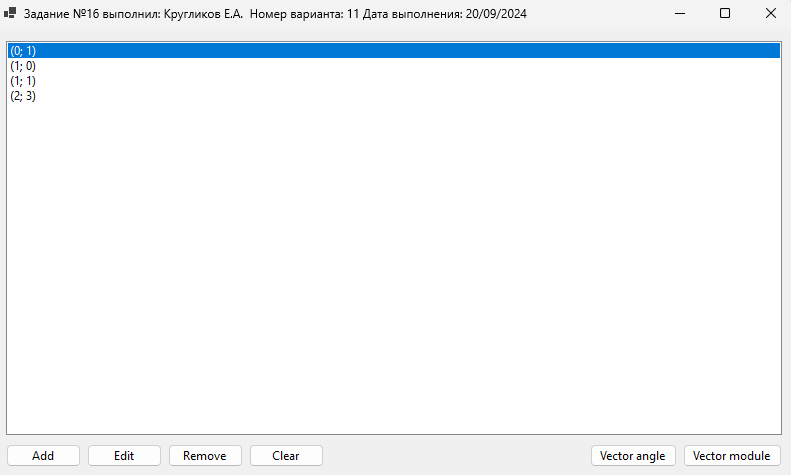


Рисунок 8 – Интерфейс главной формы

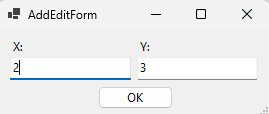


Рисунок 9 – Интерфейс окна добавления/изменения ячейки

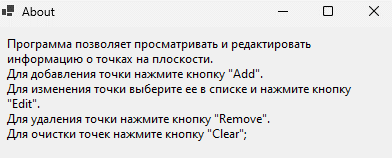


Рисунок 10 – Интерфейс окна справки

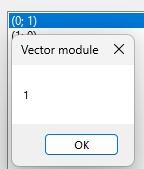


Рисунок 11 – Интерфейс окна с подробной информацией о длине модуля

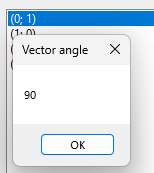


Рисунок 12 - Интерфейс окна с подробной информацией об угле модуля

1. **Подтверждение соответствия графического пользовательского интерфейса требованиям к оформлению**

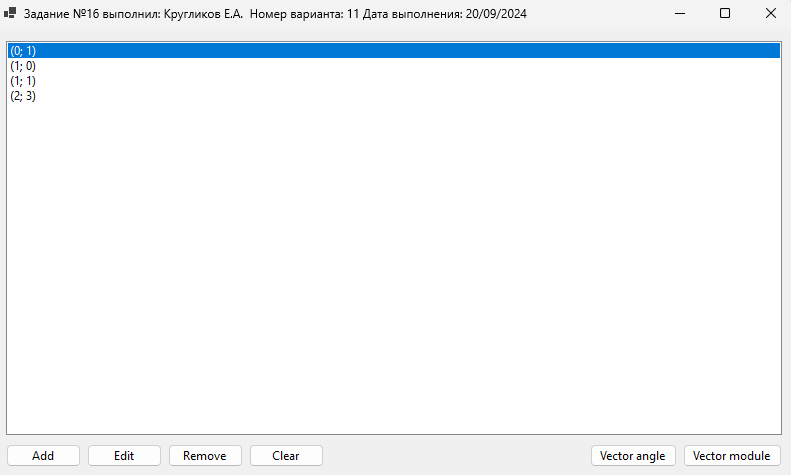


Рисунок 13 – Подтверждение соответствия графического пользовательского интерфейса требованиям к оформлению.

1. **Расчёт тестовых примеров с использованием составленного программного обеспечения**

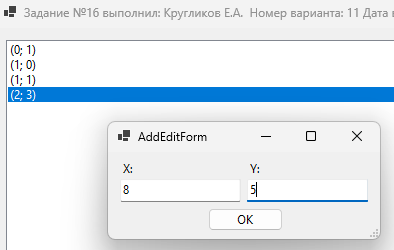


Рисунок 14 – Изменения данных, введенные с помощью составленного ПО.

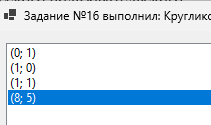


Рисунок 15 – Изменения, введенные на Рисунке 12 сохранились.

1. **Формулировка вывода о проделанной работе**

Освоил навыки программирования инкапсулированных классов. Улучшил навыки работы в режиме *Windows Forms Application.*