Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Российский университет транспорта» (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа №2

по дисциплине

**«**Методы программирования**»**

Выполнили: ст. гр. ТКИ-341

Поваляева А.В.

Топталова С.Д.

Вариант №2

Проверил: доцент кафедры УиЗИ, к.т.н. Сафронов А.И.

Москва ­– 2024 г.

1. **Цель работы**

Закрепить навыки разработки визуального пользовательского интерфейса, освоить работу с текстовыми файлами и кодировкой в среде *Microsoft* *Visual Studio*, научиться реализовывать настройку множественных состояний объектов посредством управления компонентами со внутренней индексацией.

1. **Формулировка задачи**

В интегрированной среде разработки *Microsoft* *Visual Studio* разработать программу в режиме *Windows Forms Application* на языке *Visual C#*, представляющую собой экранную форму, содержащую главное меню, позволяющее:

1. Начать работу с приложением.
2. Прервать работу приложения.
3. Предоставить пользователю справочную информацию о работе с приложением.

Сама программа должна реализовывать вывод в графический элемент управления (например, *PictureBox*) главной экранной формы плоскостную геометрическую фигуру, выбираемую пользователем из списка (вид списка \* задаётся вариантом индивидуального задания). Список должен обязательно содержать следующие пункты:

1. «Квадрат»,
2. «Прямоугольный треугольник»,
3. «Эллипс»,
4. «Равнобедренный треугольник»,
5. «Круг»,
6. «Равносторонний треугольник»,
7. «Окружность»,
8. «Ромб»,
9. «Трапеция»,
10. «Параллелограмм»,
11. «Прямоугольник».

Требования к оформлению экранной формы:

1. Заголовок экранной формы должен содержать надпись вида: «Задание №2 выполнил: [Фамилия И.О. авторов]; Номер варианта: [Номер]; Дата выполнения: [дд/мм/гггг]».

2. Дата выполнения проставляется в момент, когда программа считается законченной и по ней можно готовить итоговый отчёт о выполнении работы.

3. Чётные варианты заменяют стандартный курсор экранной формы со «стрелки» на «руку».

4. Справочная информация должна быть вызвана в дочерней экранной форме и считана в статическое по размеру текстовое поле (*TextBox*) с выставленным запретом на редактирование текстовой информации (чётные варианты).

5. Чётные варианты заменяют цвет подложки в соответствии с представленной ниже таблицей:

Таблица 1 – Таблица системного цвета подложки

|  |  |
| --- | --- |
| 2. | *ActiveCaption* |

Подобрать читаемый цвет текста к предложенной по варианту схеме (руководствоваться принципами разработки эргономичного графического пользовательского интерфейса).

6. Исходное состояние всех элементов, расположенных на главной экранной форме, должно быть настроено через перечень параметров этих элементов.

7. В качестве исходного состояния принимается заранее известная и заполненная элементами структура списков, все элементы экранной формы за исключением главного меню находятся либо в недоступном состоянии (*.Enabled = false*), либо в невидимом состоянии (*.Visible = false*).

8. Пункт меню «Начало работы с приложением» должен реализовывать активацию доступа пользователя к элементам или отображение элементов на экранной форме для пользователя.

Размещение основных элементов:

Таблица 2 – Таблица индексации фигур

|  |  |
| --- | --- |
| **№ вар.** | **Пункт порядок индексов фигур из списка** |
| **2** | 10, 9, 5, 1, 3, 11, 2, 6, 7, 8, 4. |

Индивидуальное изображение:

Таблица 3 – Сложное, комбинированное изображение

|  |  |  |
| --- | --- | --- |
| **№ вар.** | **Пункт списка** | **№ позиции** |
| **2** | Крупная, центрированная надпись «Текст» | 10 |

Индивидуальные элементы управления:

Таблица 4 – Элемент управления, необходимые для выбора изображаемых в *PictureBox* фигур

|  |  |
| --- | --- |
| **№ вар.** | **Элемент управления** |
| **2** | Список |

1. **Диаграмма классов, входящих в состав решения**

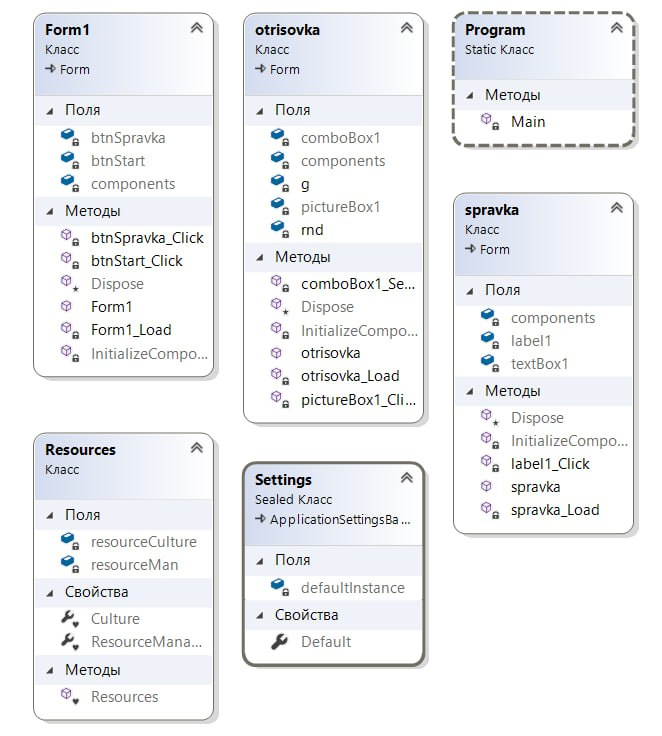


Рисунок 1 – Диаграмма классов, входящих в состав решения

1. **Сеть Петри запрограммированного процесса**

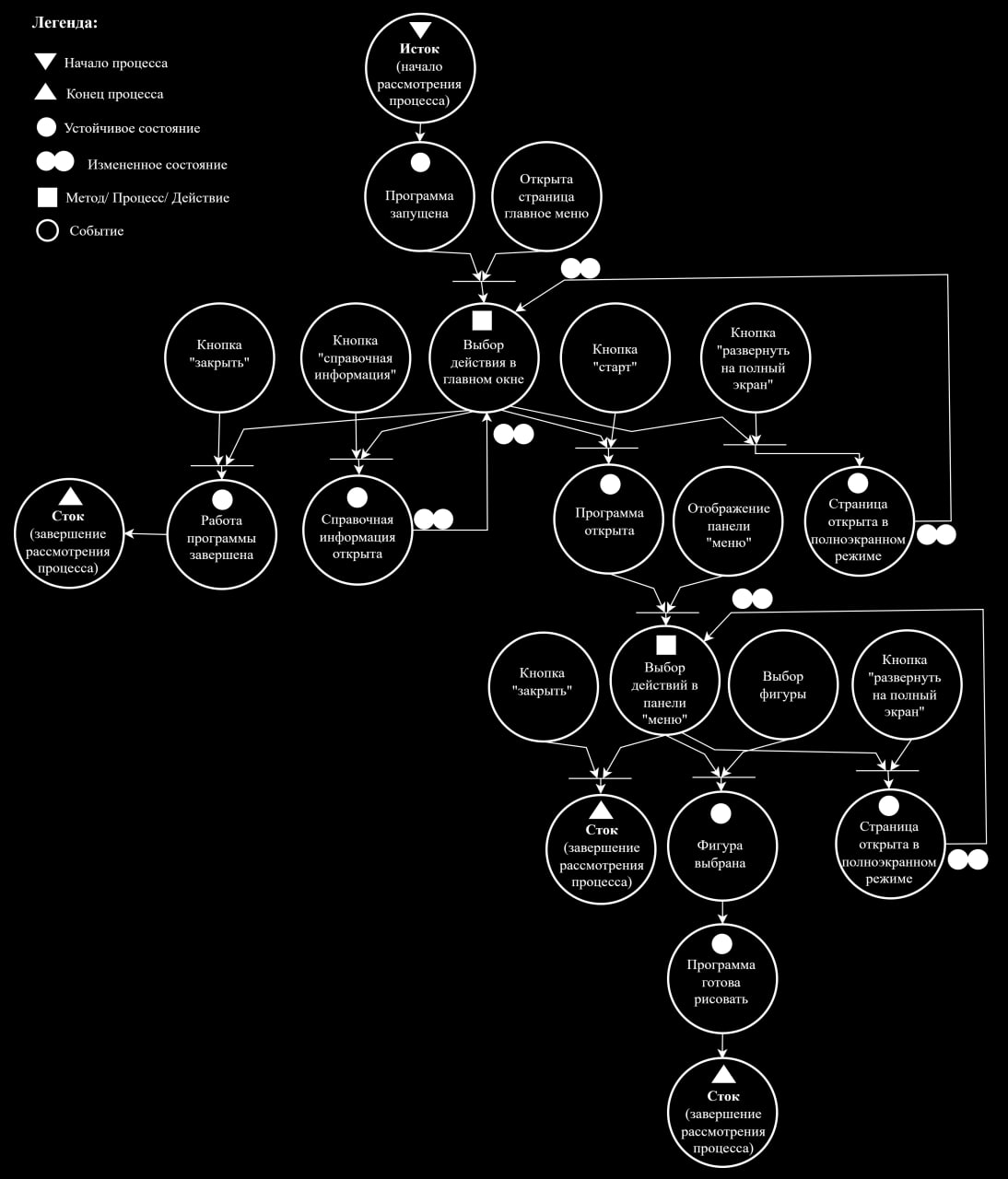


Рисунок 2 – Сеть Петри запрограммированного процесса

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

* **e1**



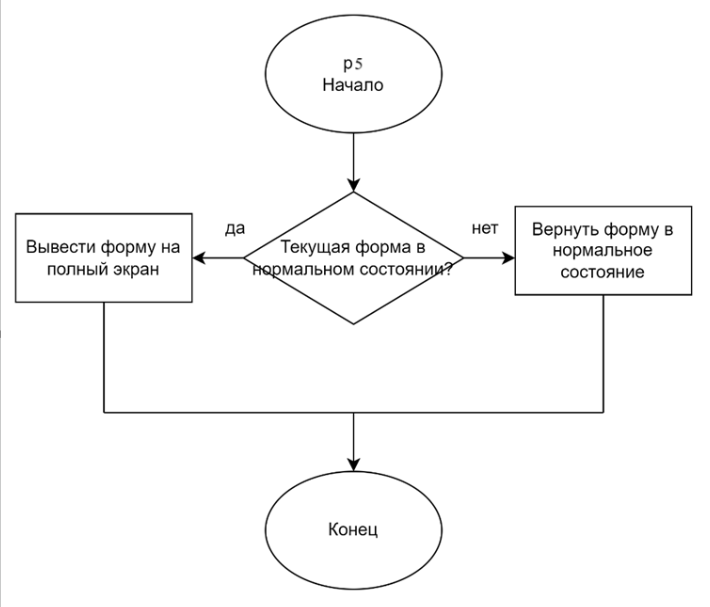
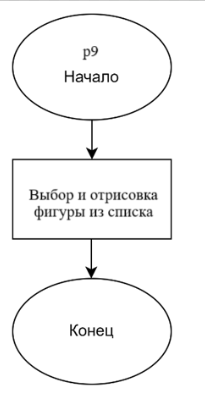
****

Рисунок 3 – Схемы алгоритмов методов в составе решения (метка e1 )

* **e2**

****

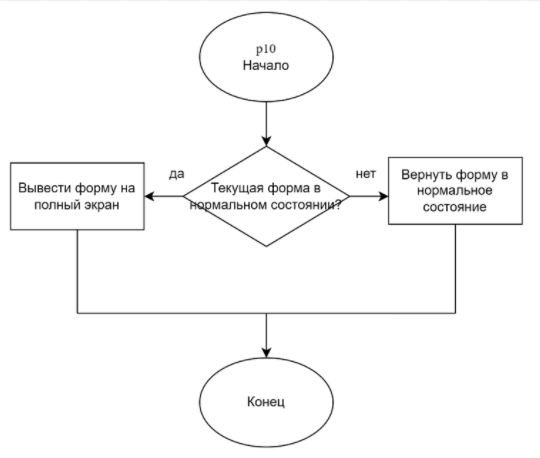
****

Рисунок 4 – Схемы алгоритмов методов в составе решения (метка e2)

1. **Тестовые примеры**

* При нажатии в главном меню кнопки «Старт» происходит запуск программы;
* При нажатии кнопки «Справочная информация» происходит получение пользователем информации о программе.

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

* **frmMain**

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;

namespace frmMain

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

}

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

{

spravka form2 = new spravka();

form2.Show();

}

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

{

otrisovka form3 = new otrisovka();

form3.Show();

}

}

}

* **frmSpravka**

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;

namespace frmMain

{

public partial class spravka : Form

{

public spravka()

{

InitializeComponent();

}

private void label1\_Click(object sender, EventArgs e)

{

}

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

{

}

}

}

* **otrisovka**

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;

namespace frmMain

{

public partial class otrisovka : Form

{

Graphics g;

Random rnd = new Random();

public otrisovka()

{

InitializeComponent();

}

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

{

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

int a = int.Parse(comboBox1.SelectedIndex.ToString());

MessageBox.Show(comboBox1.SelectedIndex.ToString());

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

g = this.pictureBox1.CreateGraphics();

int x = rnd.Next(0, this.ClientSize.Width);

int y = rnd.Next(0, this.ClientSize.Height);

switch (Convert.ToInt32(comboBox1.SelectedIndex))

{

case 0: // Параллелограмм

Point[] parallelogramPoints = { new Point(x, y), new Point(x + 30, y), new Point(x + 50, y + 30), new Point(x + 20, y + 30) };

g.DrawPolygon(Pens.Black, parallelogramPoints);

break;

case 1: // Трапеция

Point[] trapezoidPoints = { new Point(x + 10, y), new Point(x + 40, y), new Point(x + 30, y + 30), new Point(x + 20, y + 30) };

g.DrawPolygon(Pens.Black, trapezoidPoints);

break;

case 2: // Закрашенный круг

g.FillEllipse(Brushes.Black, x, y, 30, 30);

break;

case 3: // Квадрат

g.DrawRectangle(Pens.Black, x, y, 30, 30);

break;

case 4: // Эллипс

g.DrawEllipse(Pens.Black, x, y, 50, 30);

break;

case 5: // Прямоугольник

g.DrawRectangle(Pens.Black, x, y, 50, 30);

break;

case 6: // Прямоугольный треугольник

Point[] rightTrianglePoints = { new Point(x, y), new Point(x, y + 30), new Point(x + 30, y + 30) };

g.DrawPolygon(Pens.Black, rightTrianglePoints);

break;

case 7: // Равносторонний треугольник

Point[] equilateralTrianglePoints = { new Point(x, y + 30), new Point(x + 30, y + 30), new Point(x + 15, y) };

g.DrawPolygon(Pens.Black, equilateralTrianglePoints);

break;

case 8: // Окружность

g.DrawEllipse(Pens.Black, x, y, 30, 30);

break;

case 9: // Ромб

Point[] points = { new Point(x, y + 15), new Point(x + 25, y), new Point(x + 50, y + 15), new Point(x + 25, y + 30) };

g.DrawPolygon(Pens.Black, points);

break;

case 10: // Равнобедренный треугольник

Point[] isoscelesTrianglePoints = { new Point(x, y), new Point(x + 30, y), new Point(x + 15, y + 30) };

g.DrawPolygon(Pens.Black, isoscelesTrianglePoints);

break;

case 11: // Надпись "Текст"

Rectangle textRect = new Rectangle(x, y, 100, 50);

StringFormat sf = new StringFormat

{

LineAlignment = StringAlignment.Center,

Alignment = StringAlignment.Center

};

g.DrawString("Текст", this.Font, Brushes.Black, textRect, sf);

break;

}

}

}

}

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

* **ГПИ стартовой страницы**

На рисунке 5 показано, что на стартовой странице находятся две кнопки, одна из которых запускает работу приложения (кнопка «Старт» перебрасывает пользователя на страницу с отрисовкой фигур), а при нажатии второй кнопки «Справочная информация» пользователь получает информацию о программе.

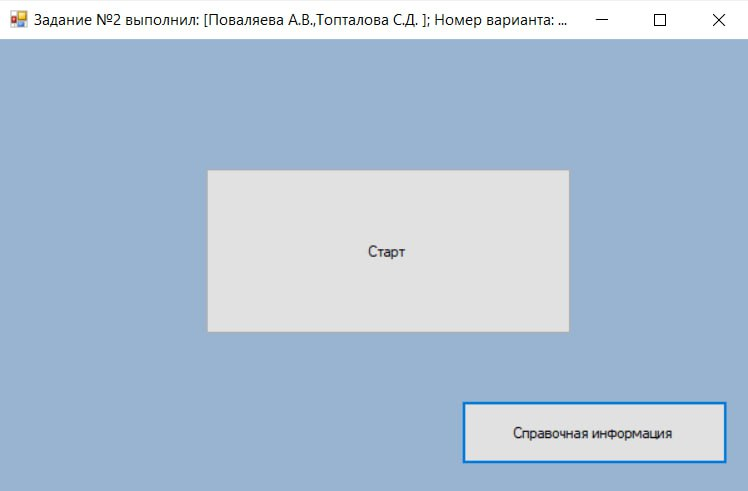


Рисунок 5 – ГПИ стартовой страницы

* **ГПИ программы**

На рисунке 6 показано, что в ГПИ приложения находится список фигур.

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание**

Рисунок 6 – ГПИ приложения

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 7 – Результат вывода в PictureBox сложного комбинированного изображения

Изображение выглядит как снимок экрана, текст, программное обеспечение, Прямоугольник

Автоматически созданное описание

Рисунок 8 – Результат вывода в PictureBox параллелограмма

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 9 – Результат вывода в PictureBox прямоугольного треугольника

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рисунок 10 – Результат вывода в PictureBox разных фигур

* **ГПИ справочной информации**

На рисунке 11 показана краткая справочная информация о программе. Справочная информация вызвана в дочерней экранной форме и считана в статическое по размеру текстовое поле (TextBox) с выставленным запретом на редактирование текстовой информации.

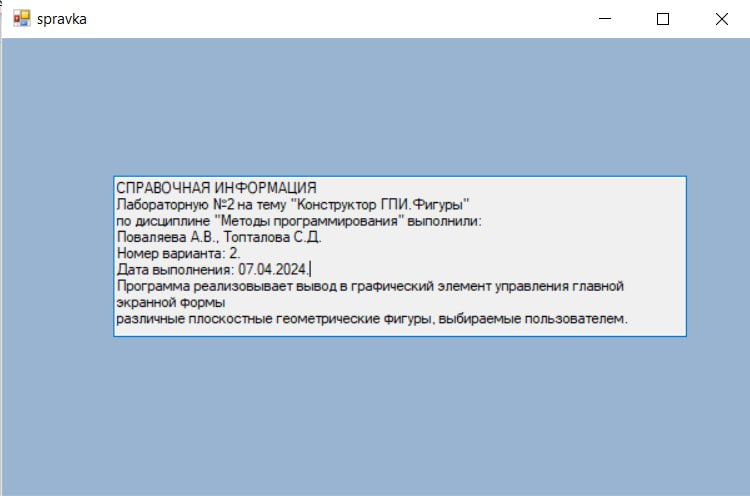


Рисунок 11 – ГПИ справочной информации

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

* Заголовок экранной формы должен содержать надпись вида: «Задание №2 выполнил: [Фамилия И.О. авторов]; Номер варианта: [Номер]; Дата выполнения: [дд/мм/гггг]»;

****

Рисунок 12 – Заголовок экранной формы

* Чётные варианты заменяют стандартный курсор экранной формы со «стрелки» на «руку»;



Рисунок 13 – Курсор «рука»

* Справочная информация должна быть вызвана в дочерней экранной форме и считана в статическое по размеру текстовое поле (*TextBox*) с выставленным запретом на редактирование текстовой информации (чётные варианты);

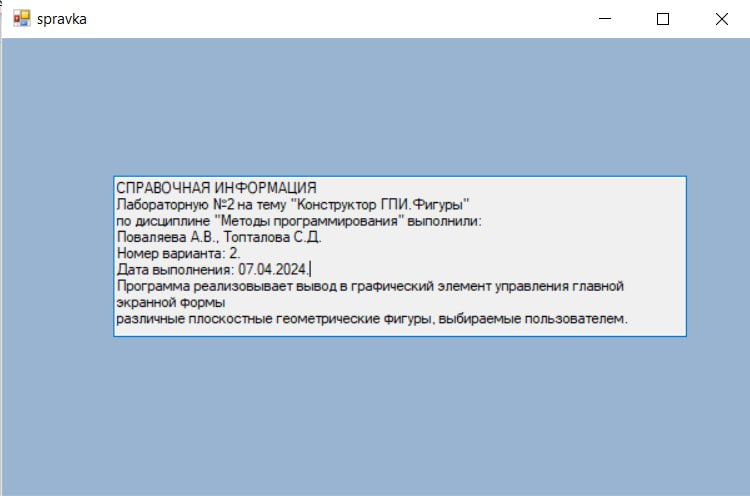


Рисунок 14 – Справочная информация вызвана в дочерней экранной форме, считана в статическое по размеру текстовое поле (*TextBox*)

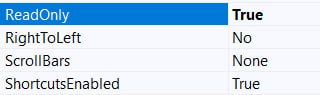


Рисунок 15 – Запрет на редактирование текстовой информации

* Заменен цвет подложки в соответствии с представленной выше таблицей 1 и подобран читаемый цвет текста к предложенной по варианту схеме;

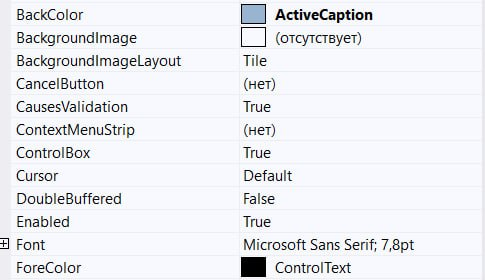


Рисунок 16 – Замена цвета подложки и подбор читаемого цвета текста

* Исходное состояние всех элементов, расположенных на главной экранной форме, настроено через перечень параметров этих элементов

(Рисунки 14, 16, 17, 19 подтверждают это);

* Все элементы программы должны носить значащие имена переменных, в которых отражено существо этих элементов;



Рисунок 17 – Значащие имена переменных

* В качестве исходного состояния принимается заранее известная и заполненная элементами структура списков, все элементы экранной формы за исключением главного меню находятся в недоступном состоянии (*.Enabled = false*);

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 18 – Элементы экранной формы находятся в недоступном состоянии

* Пункт меню «Старт» реализовывает активацию доступа пользователя к элементам или отображение элементов на экранной форме для пользователя.

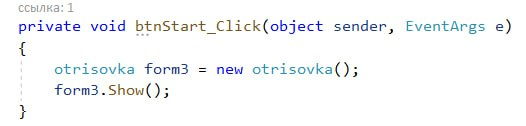


Рисунок 19 – Пункт меню «Старт» реализовывает активацию доступа пользователя к элементам на экранной форме для пользователя

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

* При нажатии в главном меню кнопки «Старт» происходит запуск программы;

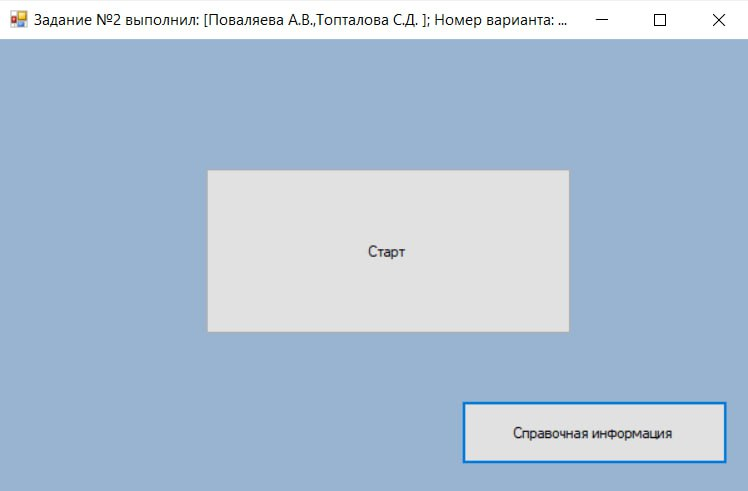


Рисунок 20 – ГПИ главного меню

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 21 – Результат нажатия кнопки «Старт» (запустилась программа)

* При нажатии кнопки «Справочная информация» происходит получение пользователем информации о программе.

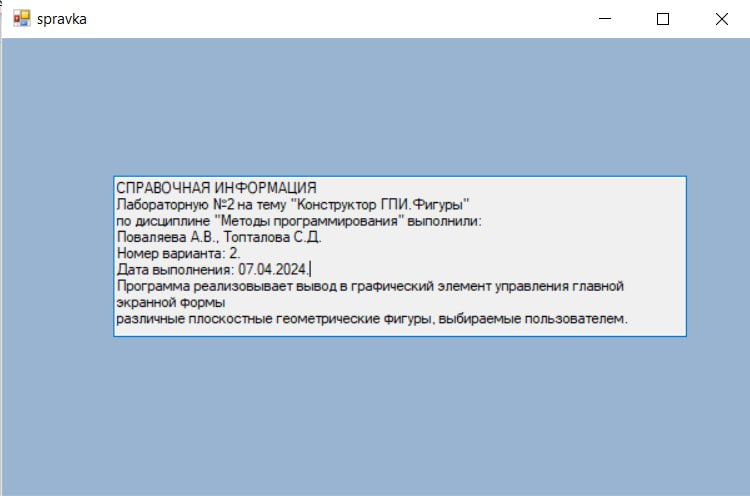


Рисунок 22 – Результат нажатия кнопки «Справочная информация» (пользователь получил информацию о программе)

1. **Вывод**

В ходе выполнения лабораторной работы нами была разработана программа на языке Visual C# с использованием интегрированной среды разработки Microsoft Visual Studio. Программа представляет собой экранную форму с главным меню, позволяющим пользователю начать или прервать работу с приложением, а также получить справочную информацию о его использовании.

Основной функционал программы состоит в выводе на главную экранную форму плоскостной геометрической фигуры, выбираемой пользователем из списка. Для реализации данной функции был использован графический элемент управления - PictureBox.

Таким образом, мы закрепили навыки разработки визуального пользовательского интерфейса, освоили работу с текстовыми файлами и кодировкой в среде *Microsoft* *Visual Studio*, научились реализовывать настройку множественных состояний объектов посредством управления компонентами со внутренней индексацией.