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**

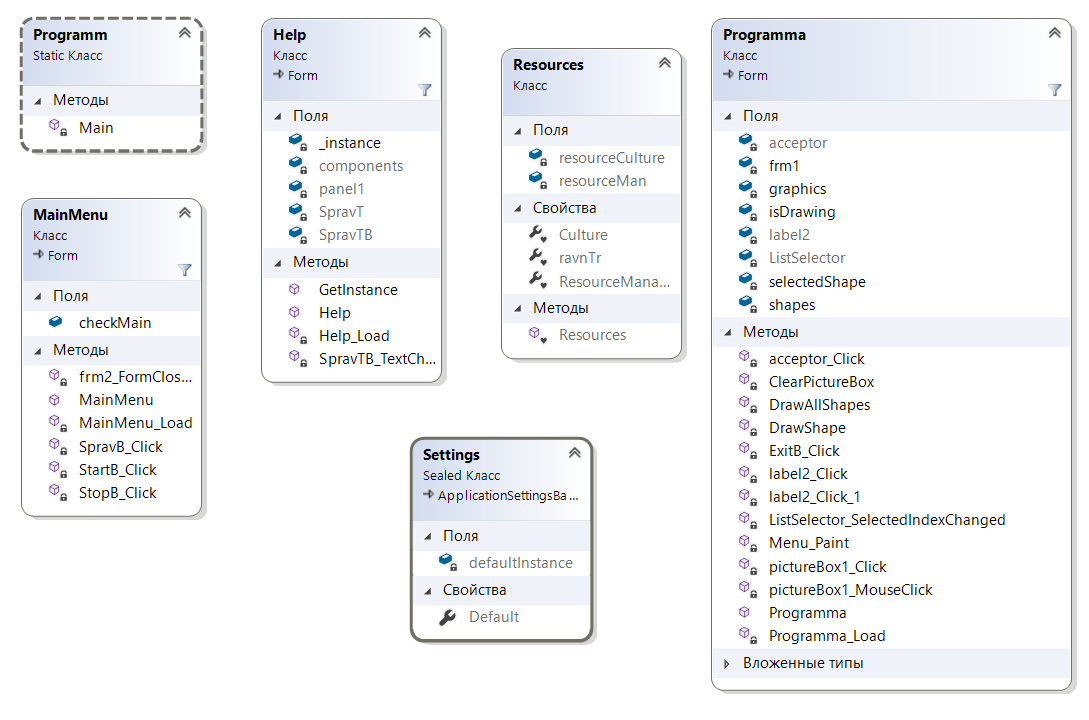
****

Рисунок 1 – Диаграмма классов, входящих в состав решения

1. **Сеть Петри запрограммированного процесса**

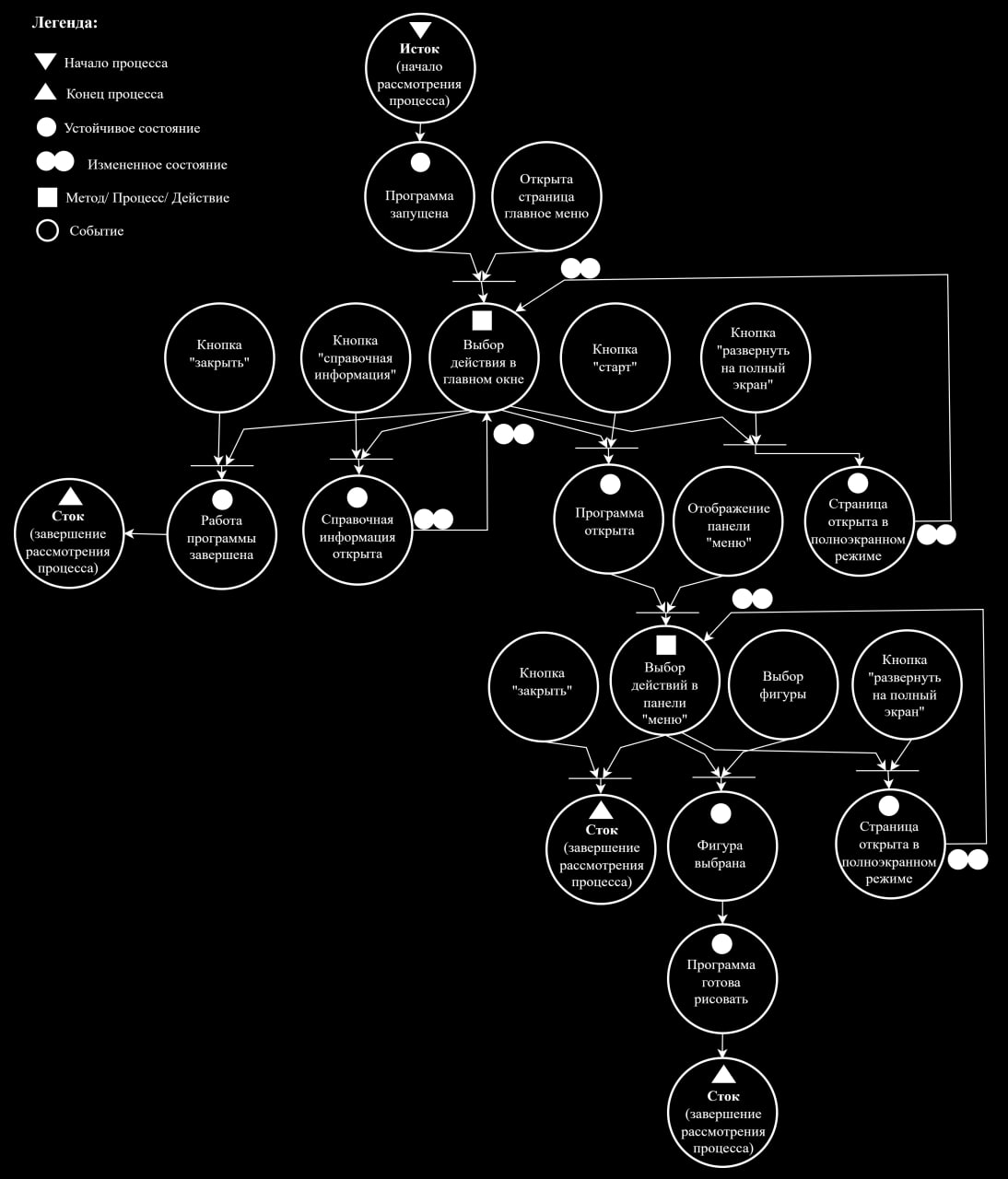


Рисунок 2 – Сеть Петри запрограммированного процесса

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

* **e1**



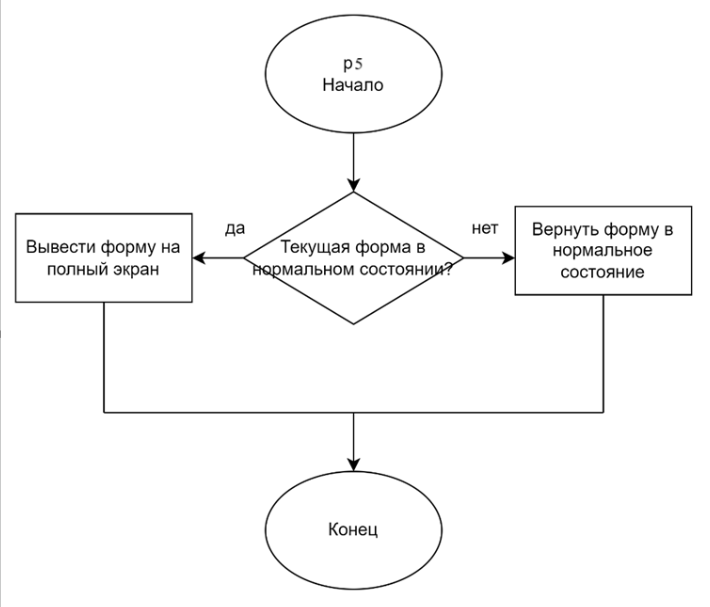
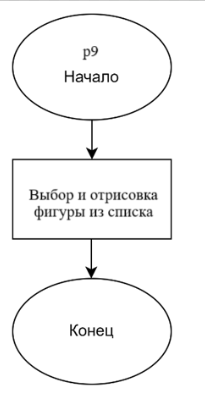
****

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

* **e2**

****

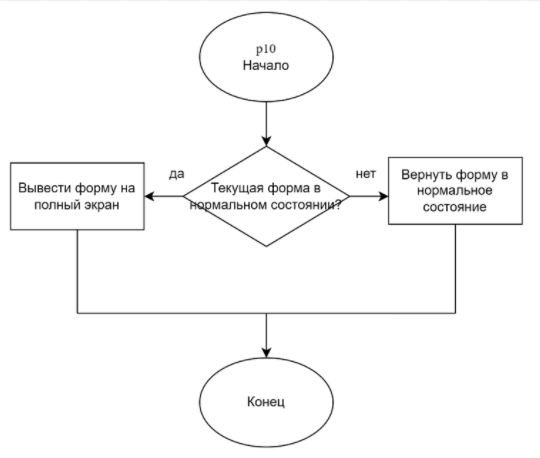
****

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**

* **Programma**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace Lab2

{

public partial class Programma :Form

{

MainMenu frm1 = new MainMenu();

private Graphics graphics;

private ShapeType selectedShape;

private bool isDrawing;

private List<Shape> shapes = new List<Shape>();

public Programma()

{

InitializeComponent();

graphics = pictureBox1.CreateGraphics();

selectedShape = ShapeType.None;

isDrawing = false;

}

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

{

ListSelector.SelectionMode = SelectionMode.One;

ListSelector.Items.Insert(0,"Параллелограмм");

ListSelector.Items.Insert(1,"Трапеция");

ListSelector.Items.Insert(2,"Круг");

ListSelector.Items.Insert(3,"Квадрат");

ListSelector.Items.Insert(4,"Эллипс");

ListSelector.Items.Insert(5,"Прямоугольник");

ListSelector.Items.Insert(6,"Прямоугольный треугольник");

ListSelector.Items.Insert(7, "Равносторонний треугольник");

ListSelector.Items.Insert(8,"Окружность");

ListSelector.Items.Insert(9,"Ромб");

ListSelector.Items.Insert(10,"Равнобедренный треугольник");

ListSelector.Items.Insert(11,"Надпись \"Текст\"");

}

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

{

this.Close();

//MainMenu frm1 = new MainMenu();

Application.Exit();

frm1.Show();

frm1.Close();

}

public class Shape

{

public ShapeType Type { get; set; }

public Point Location { get; set; }

public Shape(ShapeType type, Point location)

{

Type = type;

Location = location;

}

}

private void DrawAllShapes()

{

ClearPictureBox();

foreach (var shape in shapes)

{

DrawShape(shape.Type, shape.Location);

}

}

public enum ShapeType

{

None,

Circle,

Rectangle,

Square,

Ellipse,

RightTriangle,

IsoscelesTriangle,

EquilateralTriangle,

Rhombus,

Trapezoid,

Parallelogram,

FilledCircle,

Text

}

private void DrawShape(ShapeType shapeType, Point location)

{

//ClearPictureBox();

switch (shapeType)

{

case ShapeType.Circle:

graphics.DrawEllipse(Pens.White, location.X - 50, location.Y - 50, 100, 100);

break;

// Окружность

case ShapeType.Rectangle:

graphics.DrawRectangle(Pens.White, location.X - 50, location.Y - 50, 200, 100);

// Прямоугольник

break;

case ShapeType.Square:

graphics.DrawRectangle(Pens.White, location.X - 50, location.Y - 50, 100, 100);

// Квадрат

break;

case ShapeType.Ellipse:

graphics.DrawEllipse(Pens.White, location.X - 50, location.Y - 30, 100, 60);

// Эллипс

break;

case ShapeType.RightTriangle:

Point[] trianglePoints = { new Point(location.X - 50, location.Y + 50), new Point(location.X + 50, location.Y + 50), new Point(location.X - 50, location.Y - 50) };

graphics.DrawPolygon(Pens.White, trianglePoints);

// Прямоугольный треугольник

break;

case ShapeType.IsoscelesTriangle:

Point[] isoscelesTrianglePoints = { new Point(location.X - 50, location.Y + 50), new Point(location.X + 50, location.Y + 50), new Point(location.X, location.Y - 50) };

graphics.DrawPolygon(Pens.White, isoscelesTrianglePoints);

// Равнобедренный треугольник

break;

case ShapeType.EquilateralTriangle:

Point[] equilateralTrianglePoints = { new Point(location.X, location.Y - 50), new Point(location.X + 43, location.Y + 25), new Point(location.X - 43, location.Y + 25) };

graphics.DrawPolygon(Pens.White, equilateralTrianglePoints);

// Равносторонний треугольник

break;

case ShapeType.Rhombus:

Point[] rhombusPoints = { new Point(location.X, location.Y - 50), new Point(location.X + 50, location.Y), new Point(location.X, location.Y + 50), new Point(location.X - 50, location.Y) };

graphics.DrawPolygon(Pens.White, rhombusPoints);

// Ромб

break;

case ShapeType.Trapezoid:

Point[] invertedTrapezoidPoints = { new Point(location.X - 60, location.Y + 40), new Point(location.X + 60, location.Y + 40), new Point(location.X + 40, location.Y - 40), new Point(location.X - 40, location.Y - 40) };

graphics.DrawPolygon(Pens.White, invertedTrapezoidPoints);

// Трапеция

break;

case ShapeType.Parallelogram:

Point[] parallelogramPoints = { new Point(location.X - 40, location.Y + 40), new Point(location.X + 20, location.Y + 40), new Point(location.X + 40, location.Y - 40), new Point(location.X - 20, location.Y - 40) };

graphics.DrawPolygon(Pens.White, parallelogramPoints);

// Параллелограмм

break;

case ShapeType.FilledCircle: // круг

graphics.FillEllipse(Brushes.White, location.X - 50, location.Y - 50, 100, 100);

break;

case ShapeType.Text:

graphics.DrawString("Текст", this.Font, Brushes.White, location);

break;

}

}

private void ClearPictureBox()

{

graphics.Clear(SystemColors.Desktop);

}

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

{

Dictionary<int, ShapeType> SelectedObj = new Dictionary<int, ShapeType>();

SelectedObj.Add(0, ShapeType.Parallelogram);

SelectedObj.Add(1, ShapeType.Trapezoid);

SelectedObj.Add(2, ShapeType.FilledCircle);

SelectedObj.Add(3, ShapeType.Square);

SelectedObj.Add(4, ShapeType.Ellipse);

SelectedObj.Add(5, ShapeType.Rectangle);

SelectedObj.Add(6, ShapeType.RightTriangle);

SelectedObj.Add(7, ShapeType.EquilateralTriangle);

SelectedObj.Add(8, ShapeType.Circle);

SelectedObj.Add(9, ShapeType.Rhombus);

SelectedObj.Add(10, ShapeType.IsoscelesTriangle);

SelectedObj.Add(11, ShapeType.Text);

if (ListSelector.SelectedIndex != -1)

{

ClearPictureBox();

shapes.Clear();

//selectedShape = ShapeType.Parallelogram;

selectedShape = SelectedObj[ListSelector.SelectedIndex];

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void pictureBox1\_MouseClick(object sender, MouseEventArgs e)

{

if (isDrawing == true)

{

shapes.Add(new Shape(selectedShape, e.Location));

DrawAllShapes();

}

}

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

{

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void Menu\_Paint(object sender, PaintEventArgs e)

{

}

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

{

if (selectedShape != ShapeType.None)

{

isDrawing = true;

}

}

private void label2\_Click\_1(object sender, EventArgs e)

{

}

}

}

* **Help**

using System;

using System.Windows.Forms;

namespace Lab2

{

public partial class Help : Form

{

public Help()

{

InitializeComponent();

}

private void SpravTB\_TextChanged(object sender, EventArgs e)

{

}

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

{

}

private static Help \_instance;

public static Help GetInstance()

{

if (\_instance == null) \_instance = new Help();

return \_instance;

}

}

}

* **MainMenu**

using System;

using System.Windows.Forms;

namespace Lab2

{

public partial class MainMenu : Form

{

public MainMenu()

{

InitializeComponent();

}

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

{

}

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

{

Application.Exit();

}

private void frm2\_FormClosed(object sender, FormClosedEventArgs e)

{

this.Show();

}

//

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

{

Programma frm2 = new Programma();

//frm2.TopMost = true;

frm2.WindowState = FormWindowState.Maximized;

frm2.FormClosed += new FormClosedEventHandler(frm2\_FormClosed);

this.Hide();

frm2.ShowDialog();

}

public bool checkMain = false;

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

{

Help frm3 = new Help();

//Help frm3 = Help.GetInstance();

if (!checkMain)

{

checkMain = true;

frm3.Show();

//frm3.ShowDialog(this);

}

else if (checkMain)

{

checkMain = false;

frm3.Focus();

}

//this.Hide();

}

}

}

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.

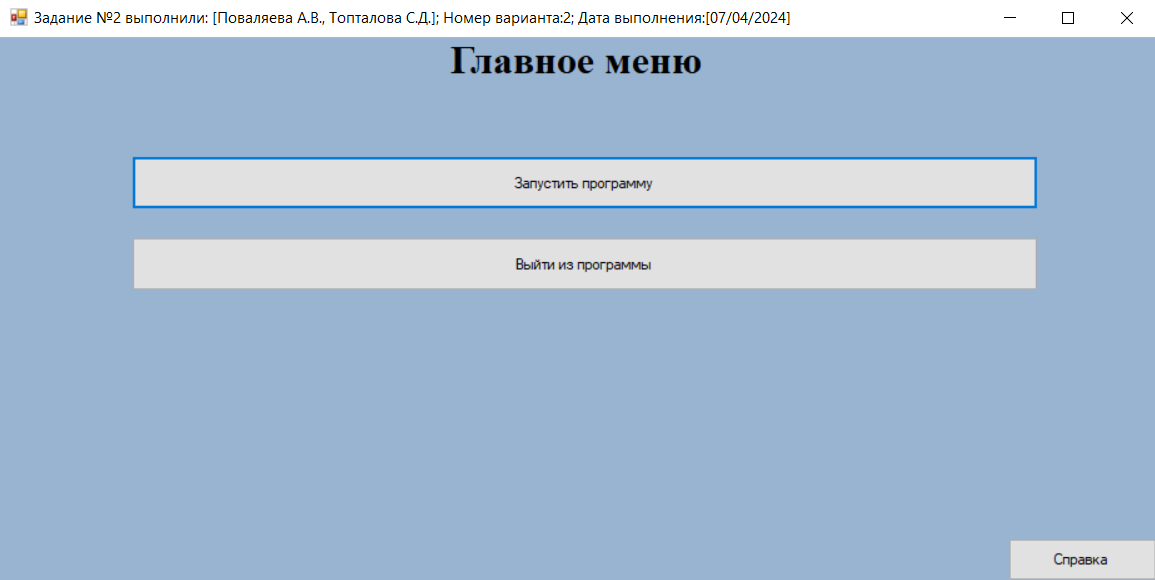
****

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, Мультимедийное программное обеспечение

Автоматически созданное описание**

Рисунок 6 – ГПИ приложения

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

Автоматически созданное описание

Рисунок 7 – Результат вывода в PictureBox сложного комбинированного изображения

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

Автоматически созданное описание

Рисунок 8 – Результат вывода в PictureBox параллелограмма

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

Автоматически созданное описание

Рисунок 9 – Результат вывода в PictureBox прямоугольного треугольника

Изображение выглядит как текст, снимок экрана, дизайн, диаграмма

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.

**Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание**

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»;

Изображение выглядит как текст, программное обеспечение, веб-страница, Значок на компьютере

Автоматически созданное описание

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);

**Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание**

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

**Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание**

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;

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

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*);



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.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

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;

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

Автоматически созданное описание**

Рисунок 20 – ГПИ главного меню

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

Автоматически созданное описание

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

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

**Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание**

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.