

**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ (МИИТ)**

**ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ УПРАВЛЕНИЯ**

**Лабораторная работа №2**

**по дисциплине** «**Методы программирования**»

«Конструктор ГПИ»

**Выполнил:** ст. гр. ТКИ-341

Прокопович Д.Е

**Проверил:** 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

# 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».

## 2.1 Индивидуальная задача

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. Заменить 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
5. Заменить цвет подложки на ControlText.

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

1. Исходное состояние всех элементов, расположенных на главной экранной форме, должно быть настроено через перечень параметров этих элементов.
2. 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).
3. Пункт меню «Начало работы с приложением» должен реализовывать активацию доступа пользователя к элементам или отображение элементов на экранной форме для пользователя.
4. Все элементы программы должны носить значащие имена переменных, в которых отражено существо этих элементов, например, главная экранная форма – frmMain, ярлык – lblHelp, комбинированный список – cmbFigures и т.д.
5. Размещение основных элементов:

7, 3, 8, 4, 6, 2, 11, 9, 10, 5, 1.

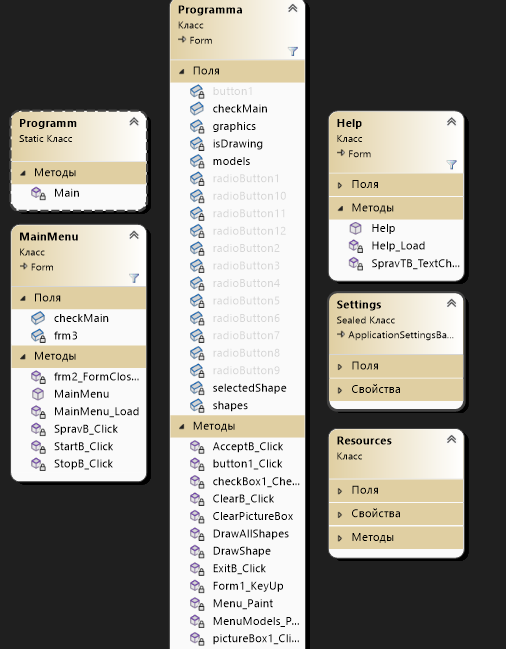
1. Индивидуальное изображение:

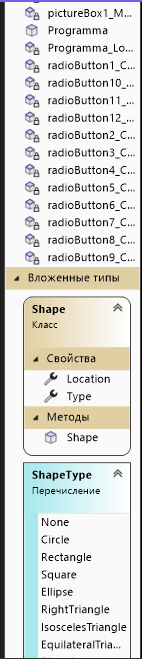
Трапеция с обводкой

1. Индивидуальные элементы управления:

Горячие клавиши

## Диаграммы классов, входящих в состав решения.





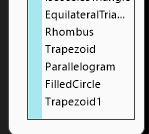


Рисунок 1 – Диаграмма классов

## Сеть Петри

**Легенда:**



Рисунок 2 – Значение обозначений

## Полная

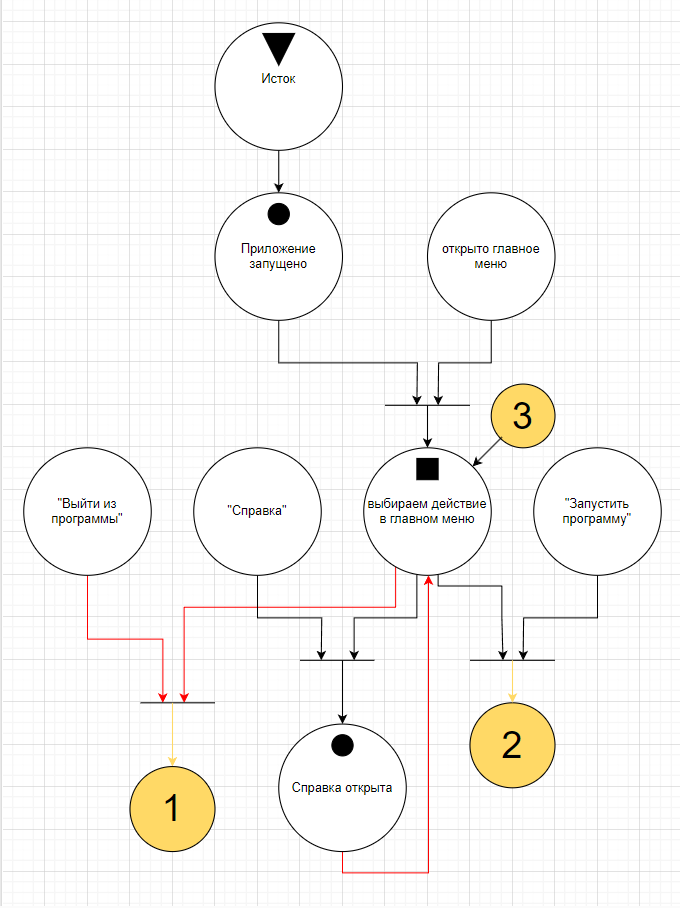


Рисунок 3 – Первая часть сети Петри (Выход из 3-ей ссылки)

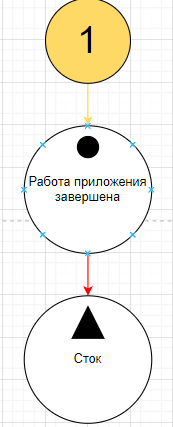


Рисунок 4 – Вторая часть сети Петри (Выход из 1-ой ссылки)

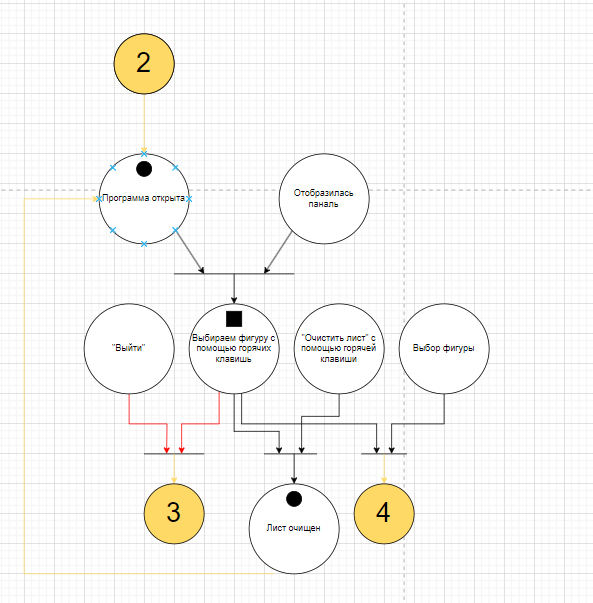


Рисунок 5 – Третья часть сети Петри (Выход из 2-ой )

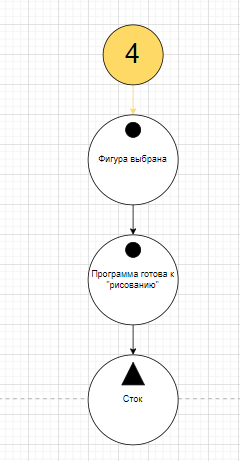


Рисунок 6 – Четвертая часть сети Петри (Выход из 4-ой ссылки)

## Краткая

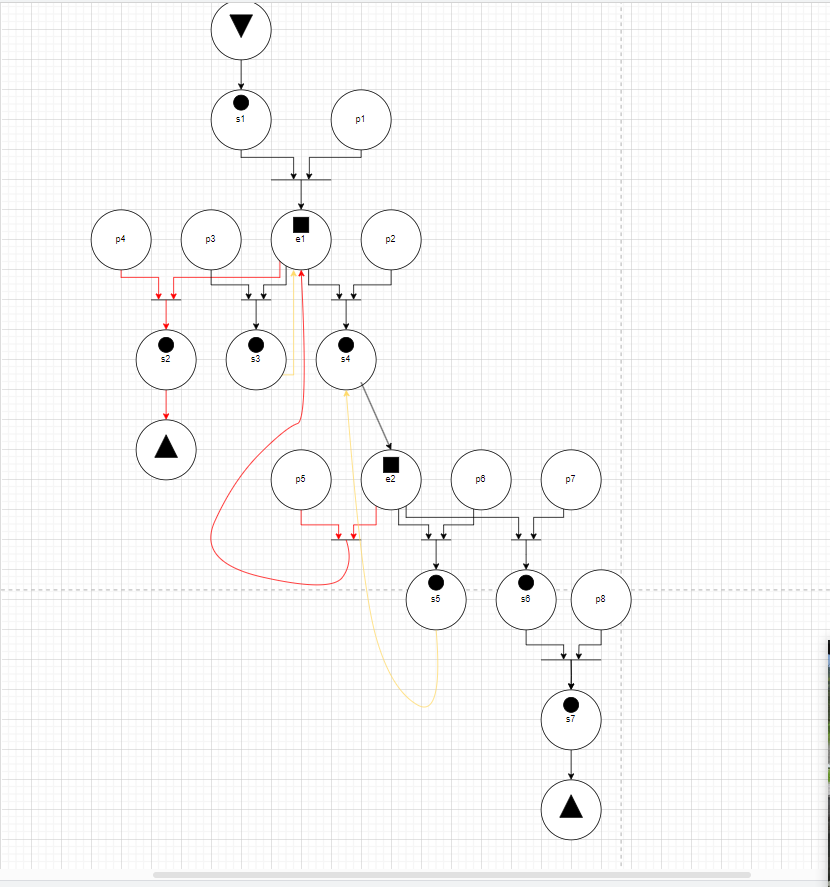


Рисунок 7 – Краткая сеть Петри

## Описание:

*–*состояния (states):

s1 *–* Приложение запущено

s2 *–* Работа приложения завершена

s3 *–* Справка открыта

s4 *–* Программа открыта

s5 *–* Лист очищен

s6 *–* Фигура выбрана

s7 *–* Программа готова к “Рисованию”

*–* действия (effects):

e1 *–* Выбираем действие в главном меню

e2 *–* Выбираем действие в панели “Меню”

*–* события (prompts):

p1 *–* Открыто главное меню

p2 *–* “Запустить программу”

p3 *–* “Справка”

p4 *–* “Выйти из программы”

p5 *–* “Выйти”

p6 *–* “Очистить лист”

p7 *–* Выбор фигуры

## Составление схем алгоритмов методов в составе решения, отмеченных на сети Петри в качестве «эффектов» (метка )

**e1 - Выбираем действие в главном меню**

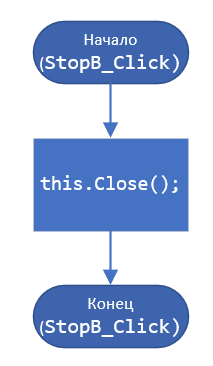
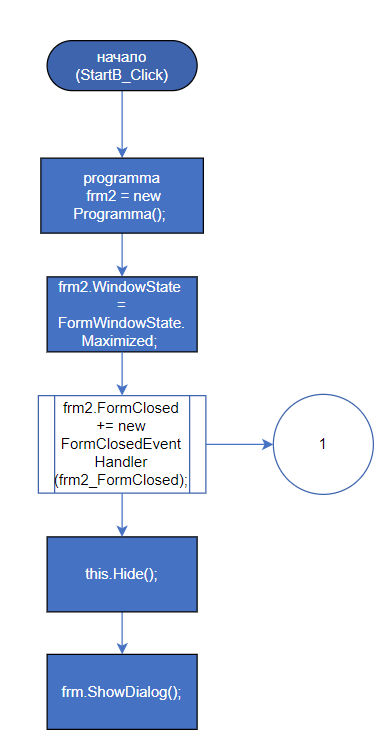


Рисунок 8 – “Выйти из программы”



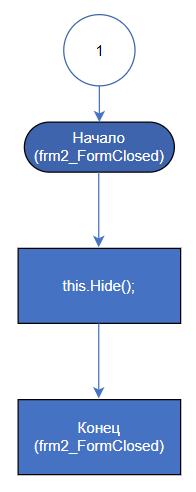


Рисунок 9 – “Запустить программу”

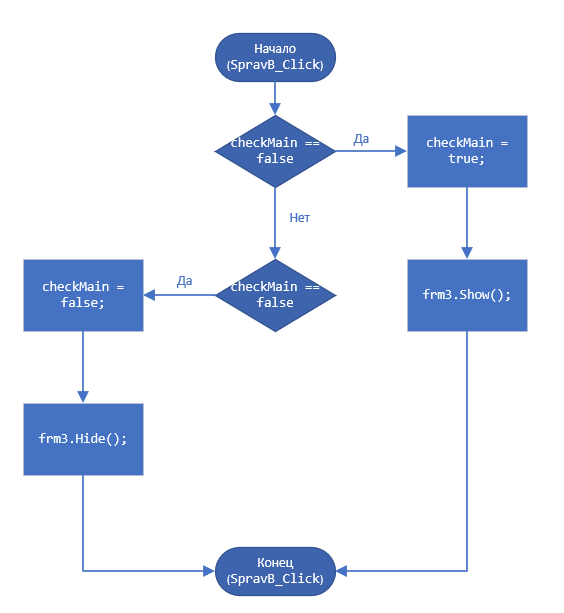


Рисунок 10 – “Справка”

**e2 - Выбираем действие в панели “Меню”**

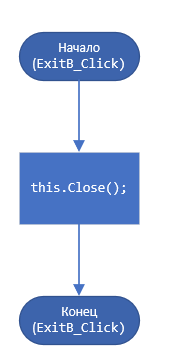
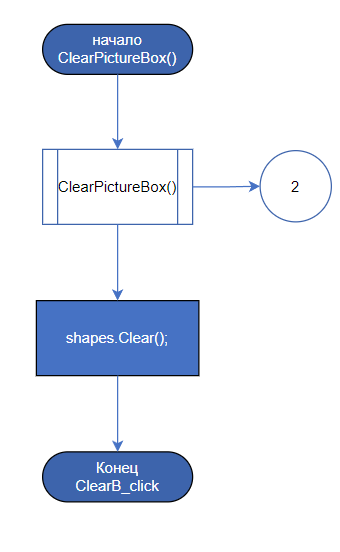


Рисунок 11 – “Выйти”



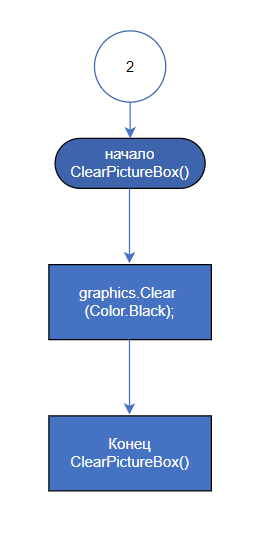


Рисунок 12 – “Очистить лист”

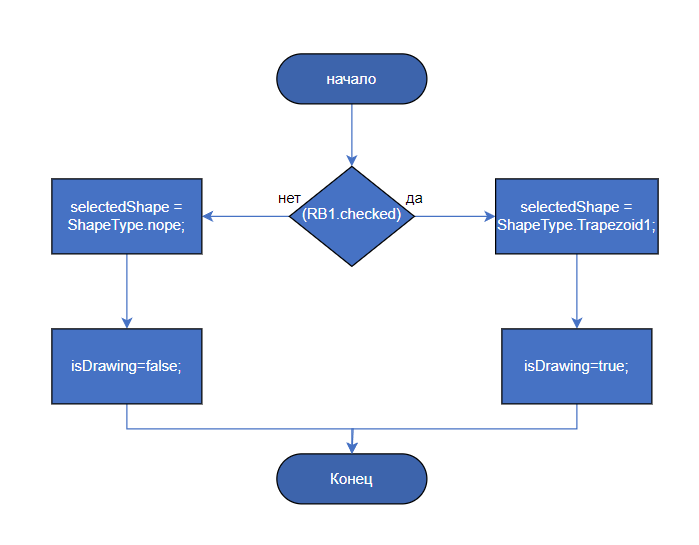


Рисунок 13 – Выбор фигуры трап. с обвод.

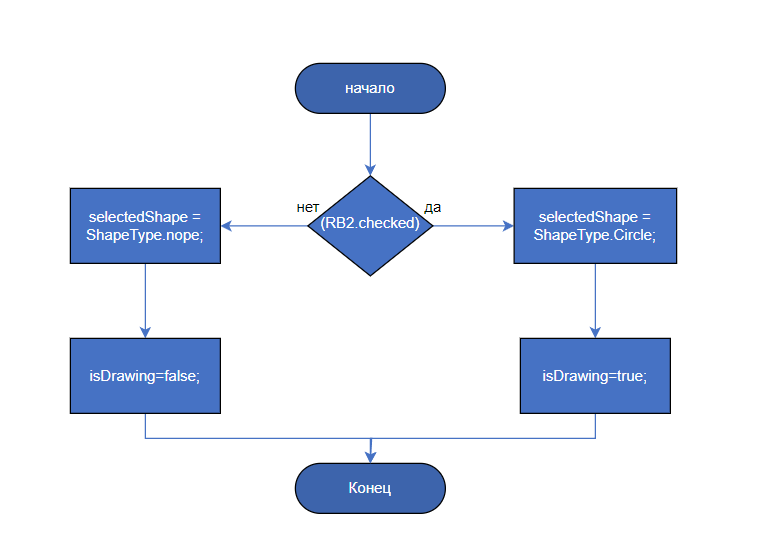


Рисунок 14 – Выбор фигуры трап. с обвод.

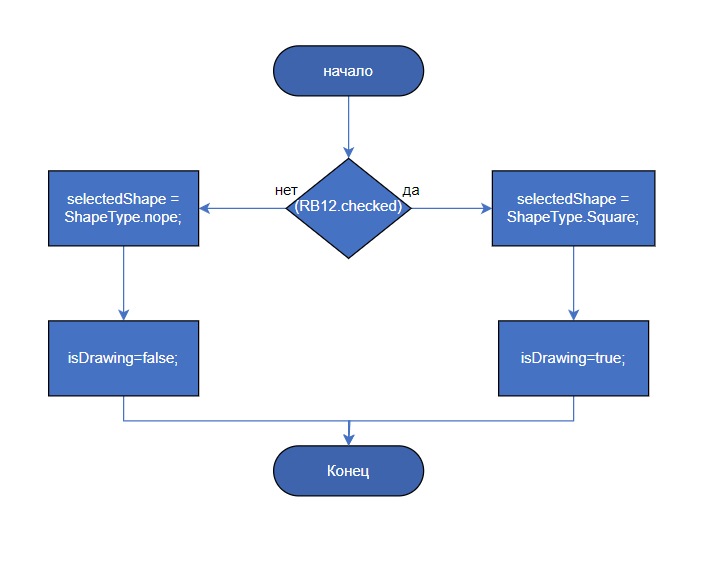
**

Рисунок 15 – Выбор фигуры трап. с обвод.

## Тестовые примеры

## Пользователь выбирает фигуры и изображает их на листе:

Ожидаемый результат:

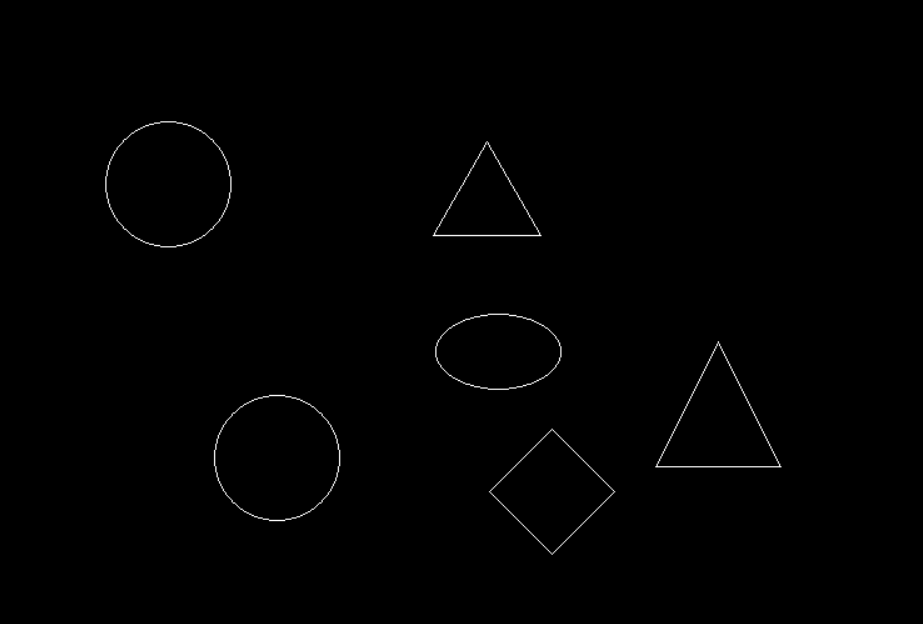


Рисунок 14 – Ожидаемый результат первого тестового примера

## Пользователь открывает панель с готовыми моделями и изображает трапецию с обводкой на листе:

Ожидаемый результат:

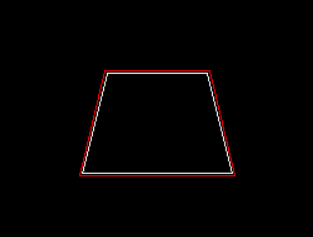


Рисунок 15 – Ожидаемый результат второго тестового примера

## Код составленного программного обеспечения

## 7.1 Код первой формы “MainMenu”

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 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)

{

this.Close();

}

//Пробный варинт

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();

//Programma frm2 = new Programma();

//frm2.Show();

//frm2.TopMost = true;

//frm2.WindowState = FormWindowState.Maximized;

//this.Hide();

////frm2.ShowDialog();

////this.Show();

}

public bool checkMain = false;

Help frm3 = new Help();

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

{

if (checkMain == false)

{

checkMain = true;

frm3.Show();

//frm3.ShowDialog();

}

else if (checkMain == true)

{

checkMain = false;

frm3.Hide();

}

//this.Hide();

}

}

## }7.2 Код второй формы “Programma”

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 Lab2

{

public partial class Programma : Form

{

private Graphics graphics;

private ShapeType selectedShape; //переменная "selectedShape" хранит выбранный тип фигуры

private bool isDrawing; //Для нажатия мышки

private List<Shape> shapes = new List<Shape>(); // для хранения инф. о фигурах, которые использовал пользователь

//Dictionary<ShapeType, Size> initialSizes = new Dictionary<ShapeType, Size>(); // Для начального размера фигур

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

{

if (e.KeyCode == Keys.Alt)

{

MessageBox.Show("Alt key released");

}

}

public Programma()

{

InitializeComponent();

graphics = pictureBox1.CreateGraphics();

selectedShape = ShapeType.None;

isDrawing = false; // Для мышки

// Подписываемся на событие KeyUp

this.KeyUp += Form1\_KeyUp;

}

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

{

this.Close();

//MainMenu frm1 = new MainMenu();

//frm1.Show();

}

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,

Trapezoid1

// Добавьте другие типы фигур

}

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:

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

// круг

break;

case ShapeType.Trapezoid1:

//трапеция

Point[] invertedTrapezoid1Points = { 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, invertedTrapezoid1Points);

//обводка трапеции

graphics.DrawLine(Pens.Red, location.X - 62, location.Y + 42, location.X + 62, location.Y + 42);

graphics.DrawLine(Pens.Red, location.X + 62, location.Y + 42, location.X + 42, location.Y - 42);

graphics.DrawLine(Pens.Red, location.X + 42, location.Y - 42, location.X - 42, location.Y - 42);

graphics.DrawLine(Pens.Red, location.X - 42, location.Y - 42, location.X - 62, location.Y + 42);

;

break;

}

}

private void ClearPictureBox()

{

graphics.Clear(Color.Black);

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton2.Checked)

{

selectedShape = ShapeType.Circle;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

{

if (radioButton3.Checked)

{

selectedShape = ShapeType.Ellipse;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

}

private void radioButton4\_CheckedChanged(object sender, EventArgs e)

{

{

if (radioButton4.Checked)

{

selectedShape = ShapeType.Rhombus;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

}

private void radioButton5\_CheckedChanged(object sender, EventArgs e)

{

{

if (radioButton5.Checked)

{

selectedShape = ShapeType.IsoscelesTriangle;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

}

private void radioButton6\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton6.Checked)

{

selectedShape = ShapeType.EquilateralTriangle;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void radioButton7\_CheckedChanged(object sender, EventArgs e)

{

{

if (radioButton7.Checked)

{

selectedShape = ShapeType.RightTriangle;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

}

private void radioButton8\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton8.Checked)

{

selectedShape = ShapeType.Rectangle;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void radioButton9\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton9.Checked)

{

selectedShape = ShapeType.Trapezoid;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void radioButton10\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton10.Checked)

{

selectedShape = ShapeType.Parallelogram;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void radioButton11\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton11.Checked)

{

selectedShape = ShapeType.FilledCircle;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void radioButton12\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton12.Checked)

{

selectedShape = ShapeType.Square;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton1.Checked)

{

selectedShape = ShapeType.Trapezoid1;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

/////

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

{

if (selectedShape != ShapeType.None)

{

isDrawing = true;

}

}

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

{

ClearPictureBox();

shapes.Clear();

}

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

{

if (isDrawing == true)

{

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

DrawAllShapes();

}

}

bool models = false;

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

{

}

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

{

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

}

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

{

}

public bool checkMain = false;

private void button1\_Click(object sender, EventArgs e)

{

Help frm3 = new Help();

if (checkMain == false)

{

checkMain = true;

frm3.Show();

//frm3.ShowDialog();

}

else if (checkMain == true)

{

checkMain = false;

frm3.Hide();

}

}

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

{

pictureBox1.BackgroundImage = null;

}

}

}

## 7.3 Код третьей формы “Help”

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 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 void Help\_FormClosing(object sender, FormClosingEventArgs e)

//{

// MainMenu frm1 = new MainMenu();

// this.Hide();

// frm1.checkMain = false;

//}

}

}

## Графический пользовательский интерфейс программного обеспечения и его описание.

## 9.1 Пользовательский интерфейс главного меню:

На рисунке 18 изображено главное меню приложения с тремя выделенными пунктами:

1 – запуск окна программы;

2 – выход из приложения;

3 – запуск окна справки;

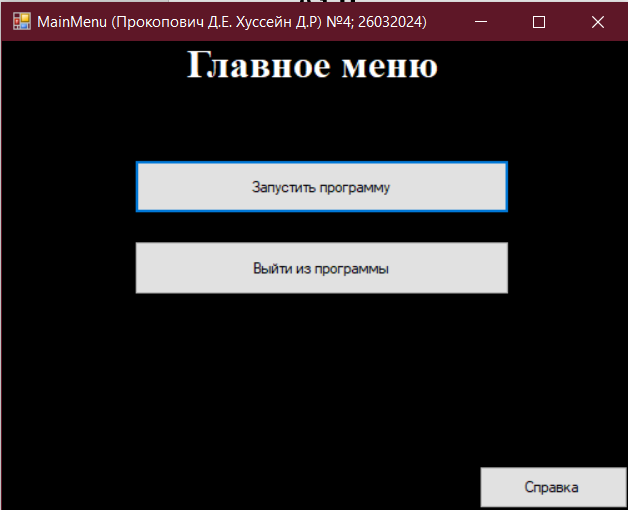


Рисунок 16 – Окно главного меню

## 9.2 Пользовательский интерфейс программы:

На рисунке 19 изображено окно программы с областью для рисования

Управление программой происходит с помощью горячих клавиш

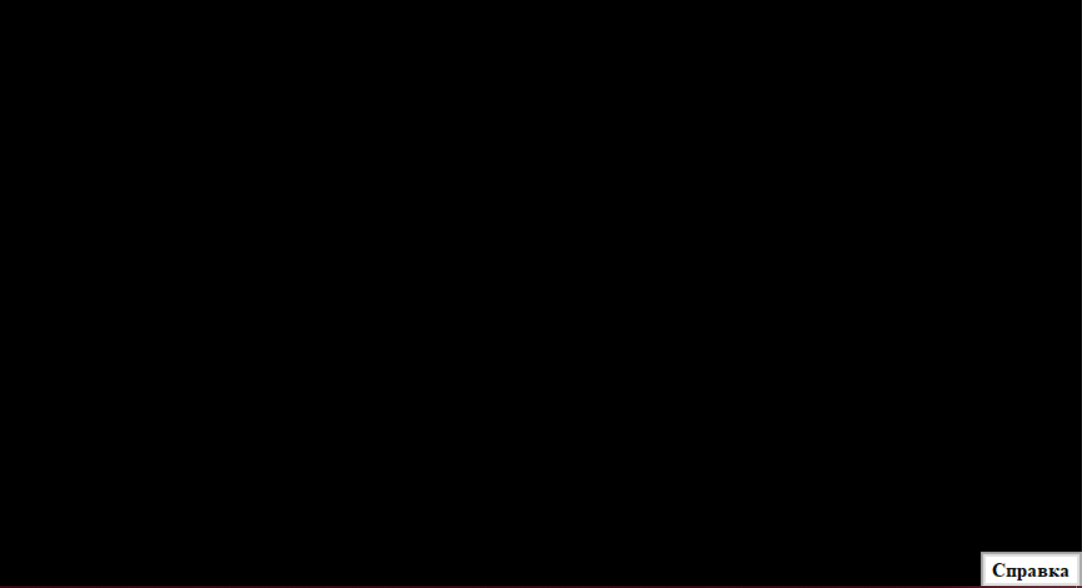


Рисунок 17 – Окно программы

## 9.3 Пользовательский интерфейс справки:

## 

Рисунок 18 – Окно краткой справки

## Подтверждение соответствия графического пользовательского интерфейса требованиям к оформлению.

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



Рисунок 19 – Заголовок главного меню

* 1. Заменить стандартный курсор экранной формы со «стрелки» на «руку».

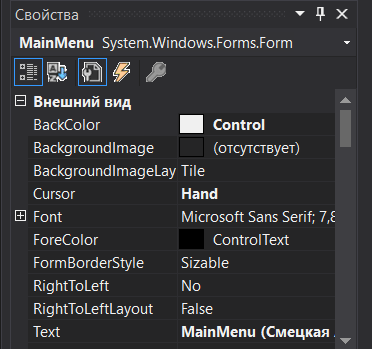


Рисунок 20 – Курсор экранной формы

## 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

## 

Рисунок 21 – Справка находится в дочерней экранной форме

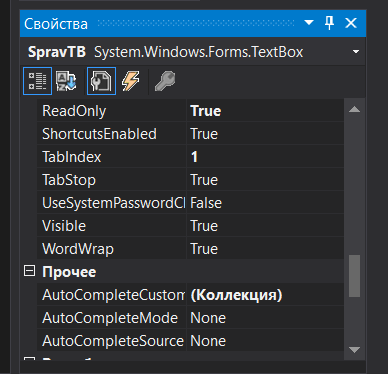


Рисунок 22 – Стоит запрет на редактирование текстовой информации

## Заменить цвет подложки на ControlText. Подобрать читаемый цвет текста к схеме (руководствоваться принципами разработки эргономичного графического пользовательского интерфейса).

## 

Рисунок 23 – Цвет подложки заменен на ControlText

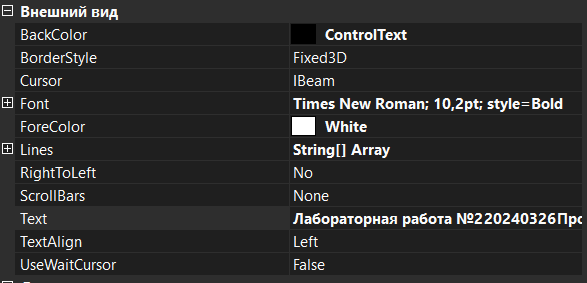


Рисунок 24 – Подобран читаемый текст для выбранной подложки

## Расчёт тестовых примеров с использованием составленного программного обеспечения.

## 

Рисунок 25 – Результат первого тестового примера

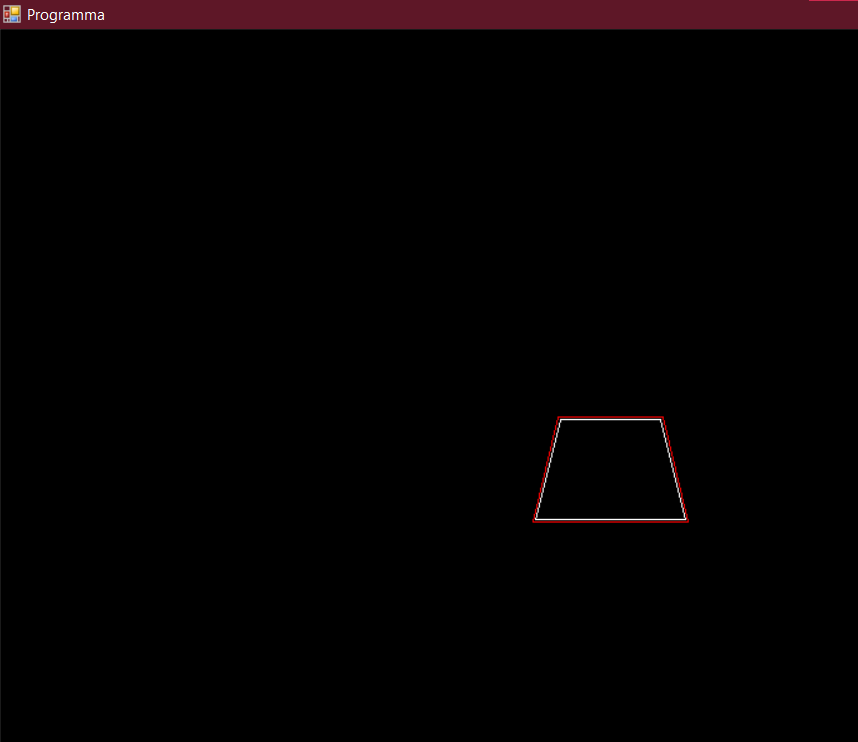


Рисунок 26 – Результат второго тестового примера

1. **Вывод по работе**

В процессе выполнения данной лабораторной работы были получены навыки разработки визуального пользовательского интерфейса, была освоена работа с текстовыми файлами и кодировкой в среде *Microsoft* *Visual Studio*, научились реализовывать настройку множественных состояний объектов посредством управления компонентами со внутренней индексацией, а также научились указывать “горячие” клавиши для управления меню.