

**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

Автоношкин А.М.

Смецкая А.С.

**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:

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

1. Индивидуальное изображение:

Дом (квадрат с маленьким квадратом с перекрестием внутри, треугольник равносторонний с маленьким кругом с перекрестием внутри)

1. Индивидуальные элементы управления:

CheckBox и кнопка подтверждения

## Диаграммы классов, входящих в состав решения.

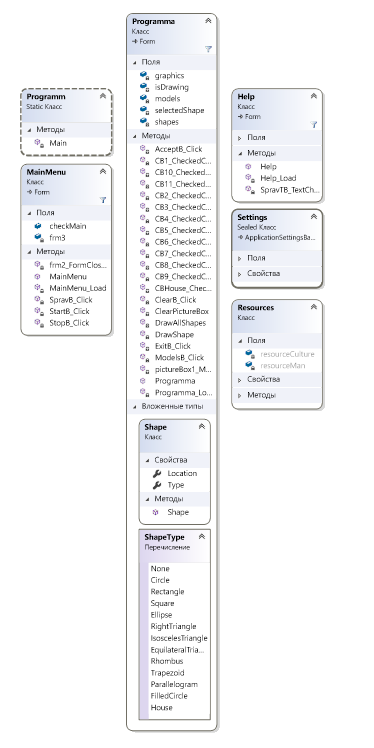


Рисунок 1 – Диаграмма классов

## Сеть Петри

**Легенда:**

****

Рисунок 2 – Значение обозначений

## Полная

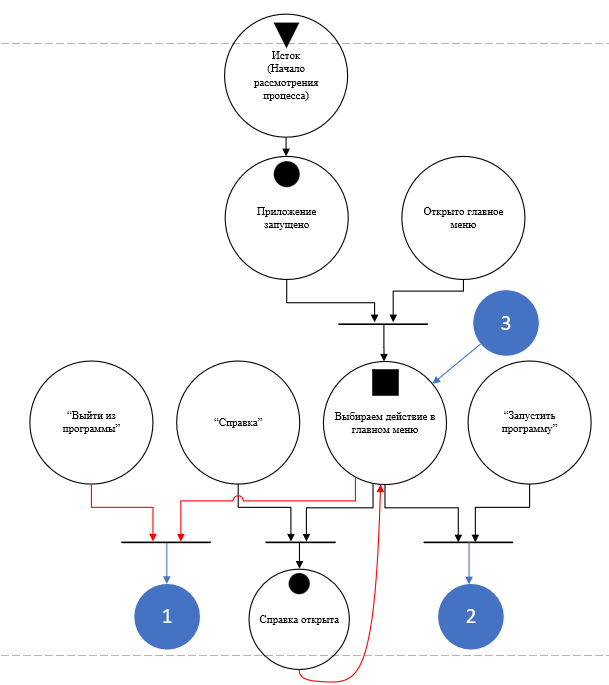


Рисунок 3 – Первая часть сети Петри (Выход из 3-ей ссылки)

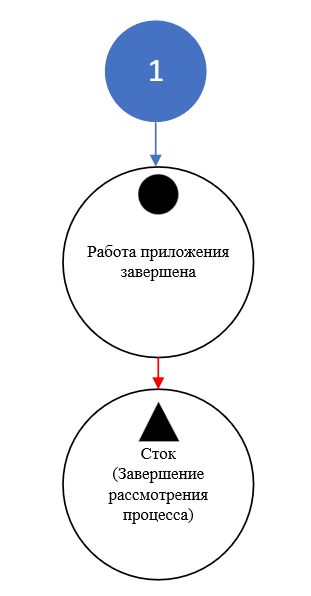


Рисунок 4 – Вторая часть сети Петри (Выход из 1-ой ссылки)

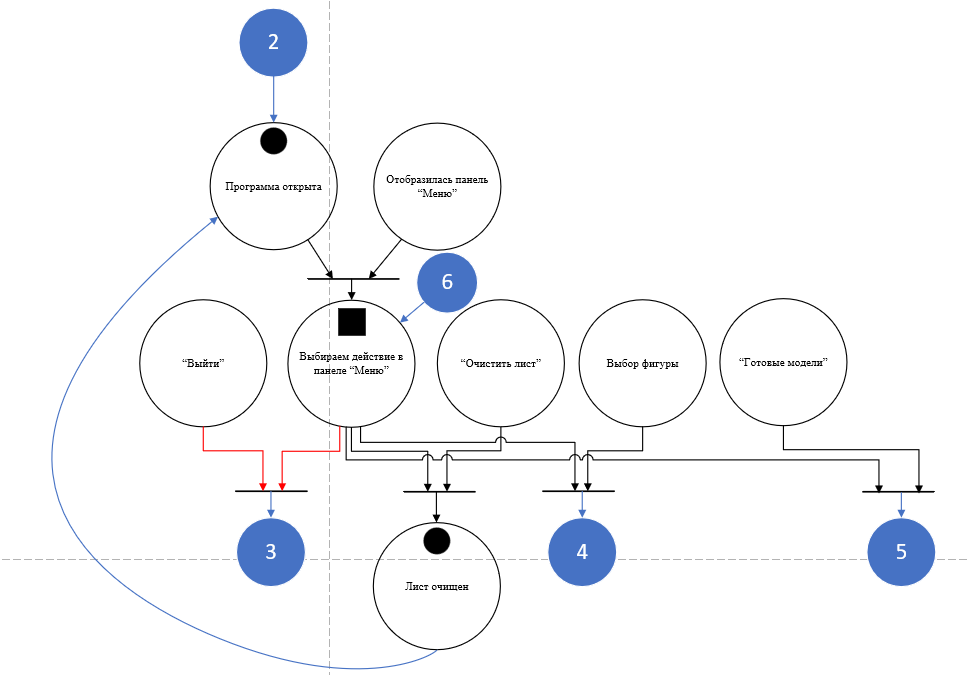


Рисунок 5 – Третья часть сети Петри (Выход из 2-ой и 6-ой ссылки)

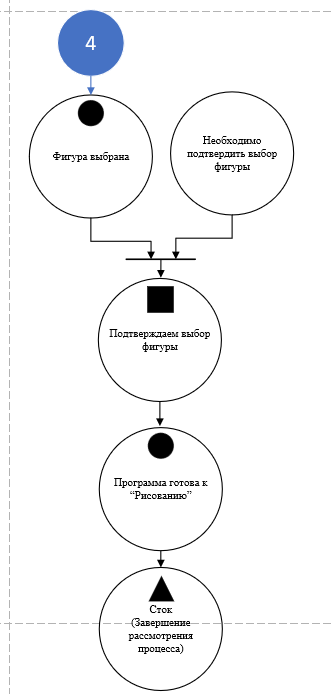


Рисунок 6 – Четвертая часть сети Петри (Выход из 4-ой ссылки)

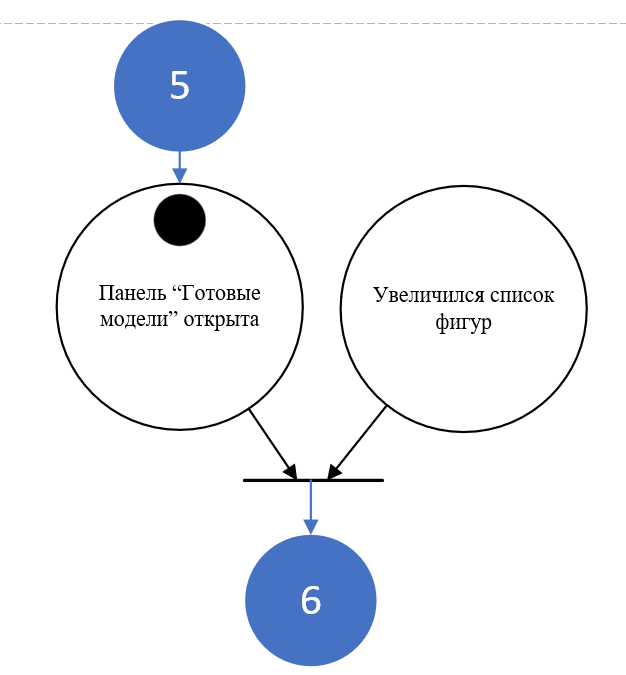
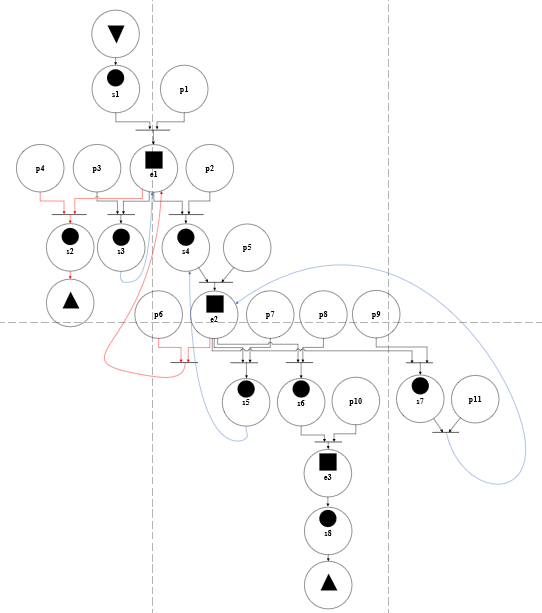


Рисунок 7 – Пятая часть сети Петри (Выход из 5-ой ссылки)

## Краткая



## Описание:

*–*состояния (states):

s1 *–* Приложение запущено

s2 *–* Работа приложения завершена

s3 *–* Справка открыта

s4 *–* Программа открыта

s5 *–* Лист очищен

s6 *–* Фигура выбрана

s7 *–* Панель “Готовые модели” открыта

s8 *–* Программа готова к “Рисованию”

*–* действия (effects):

e1 *–* Выбираем действие в главном меню

e2 *–* Выбираем действие в панели “Меню”

e3 *–* Подтверждаем выбор фигуры

*–* события (prompts):

p1 *–* Открыто главное меню

p2 *–* “Запустить программу”

p3 *–* “Справка”

p4 *–* “Выйти из программы”

p5 *–* Отобразилась панель “Меню”

p6 *–* “Выйти”

p7 *–* “Очистить лист”

p8 *–* Выбор фигуры

p9 *–* “Готовые модели”

p10 *–* Необходимо подтвердить выбор фигуры

p11 *–* Увеличился список фигур

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

**e1 - Выбираем действие в главном меню**

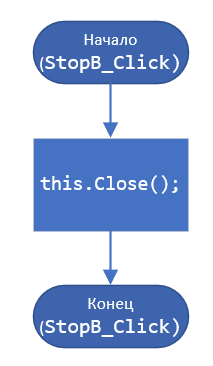


Рисунок 8 – “Выйти из программы”

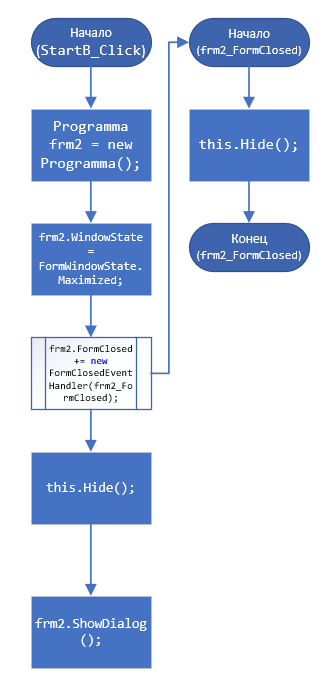


Рисунок 9 – “Запустить программу”

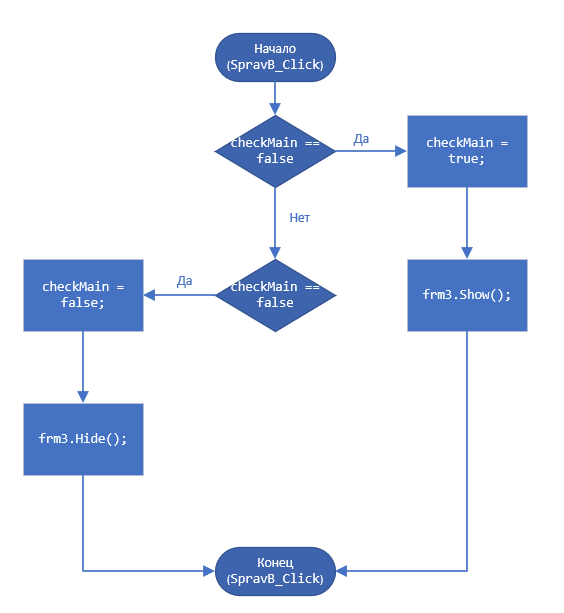


Рисунок 10 – “Справка”

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

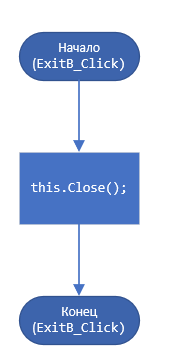


Рисунок 11 – “Выйти”

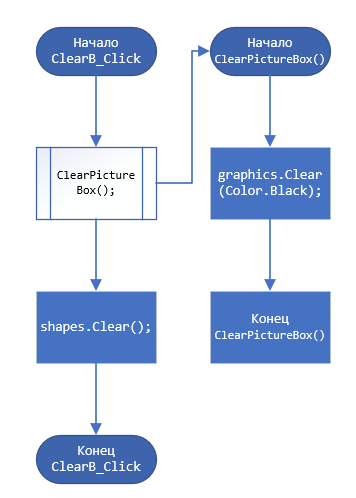


Рисунок 12 – “Очистить лист”

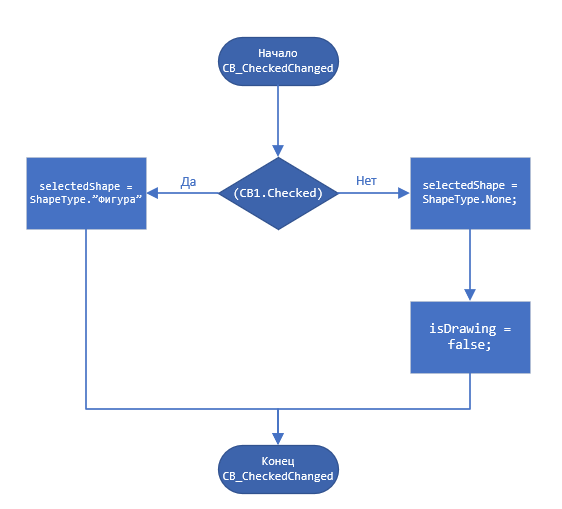


Рисунок 13 – Выбор фигуры

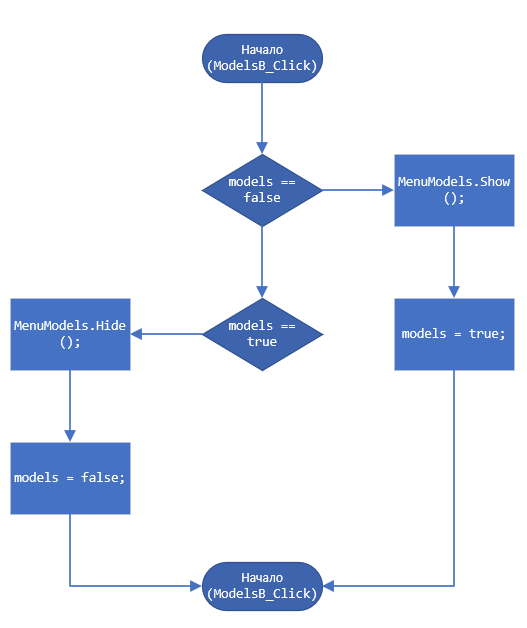


Рисунок 14 – “Готовые модели”

**e3 - Подтверждаем выбор фигуры**

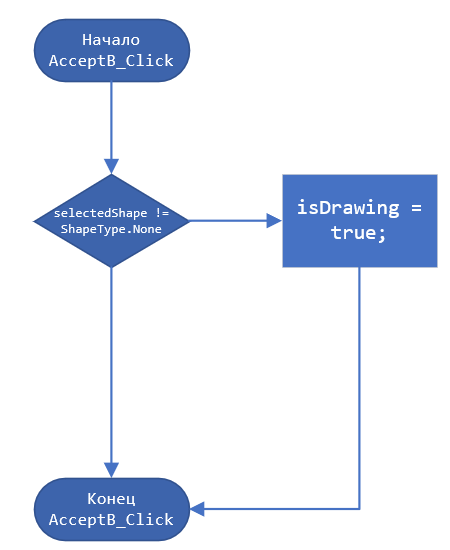


Рисунок 15 – Подтверждаем выбор фигуры

## Тестовые примеры

## Пользователь выбирает фигуры и изображает их на листе:

Ожидаемый результат:

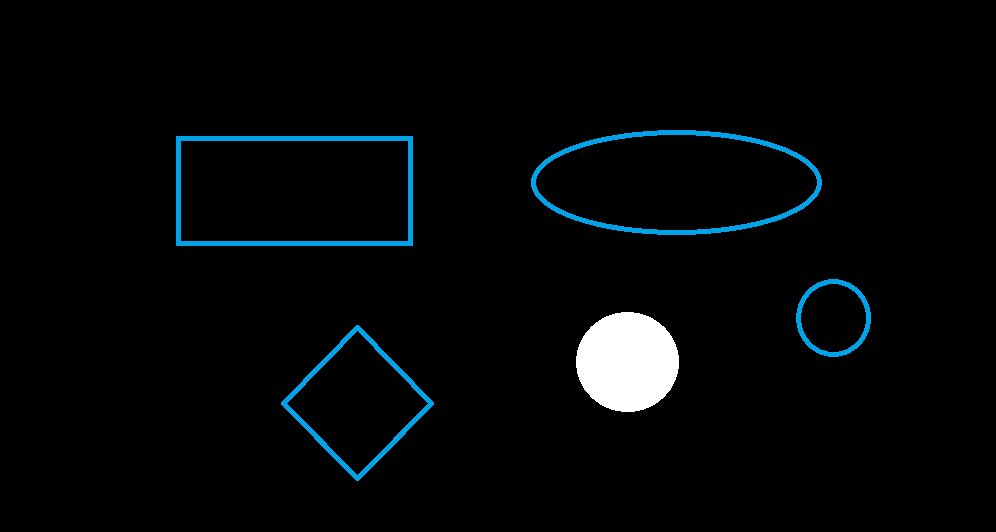


Рисунок 16 – Ожидаемый результат первого тестового примера

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

Ожидаемый результат:



Рисунок 17 – Ожидаемый результат второго тестового примера

## Листинг (код) составленного программного обеспечения

## 7.1 Листинг 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();

}

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;

Help frm3 = new Help();

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

{

if (checkMain == false)

{

checkMain = true;

frm3.Show();

}

else if (checkMain == true)

{

checkMain = false;

frm3.Hide();

}

}

}

}

## 7.2 Листинг 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

public Programma()

{

InitializeComponent();

graphics = pictureBox1.CreateGraphics();

selectedShape = ShapeType.None;

isDrawing = false; // Для мышки

}

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,

House

// 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.House:

// Рисуем равносторонний треугольник

Point[] Roof = { 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, Roof);

// Рисуем квадрат

graphics.DrawRectangle(Pens.White, location.X - 43, location.Y + 25, 86, 86);

// Рисуем второй маленький квадрат внутри основного квадрата

graphics.DrawRectangle(Pens.White, location.X - 30, location.Y + 25, 60, 60);

// Рисуем перекрестие внутри второго маленького квадрата

graphics.DrawLine(Pens.White, location.X - 30, location.Y + 55, location.X + 30, location.Y + 55);

graphics.DrawLine(Pens.White, location.X, location.Y + 25, location.X, location.Y + 85);

// Рисуем маленький круг в центре крыши

graphics.FillEllipse(Brushes.White, location.X - 15, location.Y - 15, 30, 30);

// Рисуем перекрестие внутри круга

graphics.DrawLine(Pens.Black, location.X - 15, location.Y, location.X + 15, location.Y);

graphics.DrawLine(Pens.Black, location.X, location.Y - 15, location.X, location.Y + 15);

break;

}

}

private void ClearPictureBox()

{

graphics.Clear(Color.Black);

}

private void CB1\_CheckedChanged\_1(object sender, EventArgs e)

{

if (CB1.Checked)

{

selectedShape = ShapeType.IsoscelesTriangle;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void CB2\_CheckedChanged(object sender, EventArgs e)

{

if (CB2.Checked)

{

selectedShape = ShapeType.Ellipse;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void CB3\_CheckedChanged(object sender, EventArgs e)

{

if (CB3.Checked)

{

selectedShape = ShapeType.Circle;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void CB4\_CheckedChanged(object sender, EventArgs e)

{

if (CB4.Checked)

{

selectedShape = ShapeType.FilledCircle;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void CB5\_CheckedChanged(object sender, EventArgs e)

{

if (CB5.Checked)

{

selectedShape = ShapeType.EquilateralTriangle;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void CB6\_CheckedChanged(object sender, EventArgs e)

{

if (CB6.Checked)

{

selectedShape = ShapeType.RightTriangle;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void CB7\_CheckedChanged(object sender, EventArgs e)

{

if (CB7.Checked)

{

selectedShape = ShapeType.Rhombus;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void CB8\_CheckedChanged(object sender, EventArgs e)

{

if (CB8.Checked)

{

selectedShape = ShapeType.Trapezoid;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void CB9\_CheckedChanged(object sender, EventArgs e)

{

if (CB9.Checked)

{

selectedShape = ShapeType.Rectangle;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void CB10\_CheckedChanged(object sender, EventArgs e)

{

if (CB10.Checked)

{

selectedShape = ShapeType.Parallelogram;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void CB11\_CheckedChanged(object sender, EventArgs e)

{

if (CB11.Checked)

{

selectedShape = ShapeType.Square;

}

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 ModelsB\_Click(object sender, EventArgs e)

{

if (models == false)

{

MenuModels.Show();

models = true;

}

else if(models == true)

{

MenuModels.Hide();

models = false;

}

}

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

{

if (CBHouse.Checked)

{

selectedShape = ShapeType.House;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

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

{

}

}

}

## 7.3 Листинг 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)

{

}

}

}

## 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;

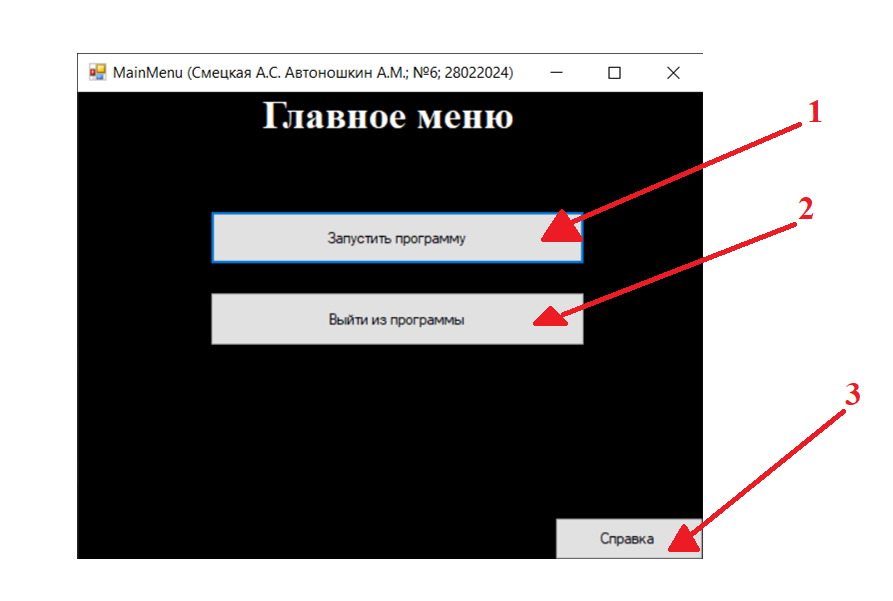


Рисунок 18 – Окно главного меню

## 9.2 Пользовательский интерфейс программы:

На рисунке 19 изображено окно программы с пятью выделенными пунктами:

1 – Выбор фигуры;

2 – Кнопка подтверждения выбора;

3 – Кнопка дополнительной панели с готовыми моделями (Рисунок 20);

4 – Кнопка очищения листа;

5 – Кнопка выхода из программы



Рисунок 19 – Окно программы

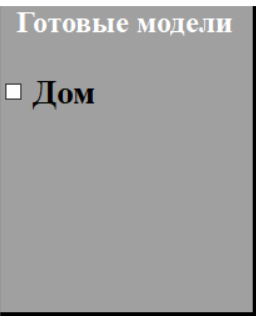


Рисунок 20 – Панель готовых моделей

## 9.3 Пользовательский интерфейс справки:

## 

Рисунок 21 – 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]

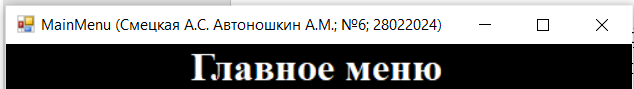


Рисунок 22 – Заголовок главного меню

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

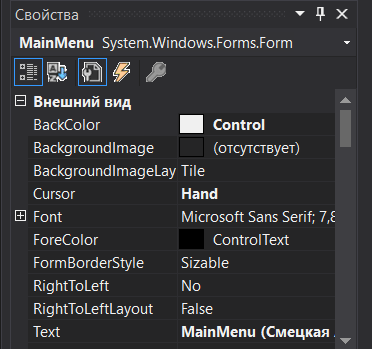


Рисунок 23 – Курсор экранной формы

## 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 24 – Справка находится в дочерней экранной форме

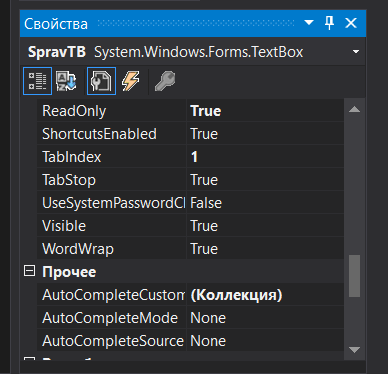


Рисунок 25 – Стоит запрет на редактирование текстовой информации

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

## 

Рисунок 26 – Цвет подложки заменен на ControlText

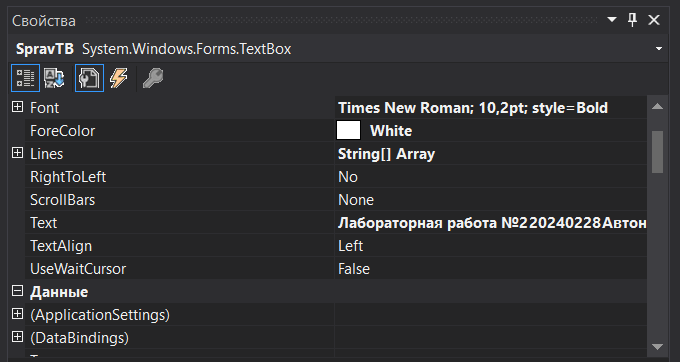


Рисунок 27 – Подобран читаемый текст для выбранной подложки

## Индивидуальные элементы управления: CheckBox и кнопка подтверждения

## 

Рисунок 28 – Индивидуальные элементы управления

## Расчёт тестовых примеров с использованием составленного программного обеспечения.

## 

Рисунок 29 – Результат первого тестового примера

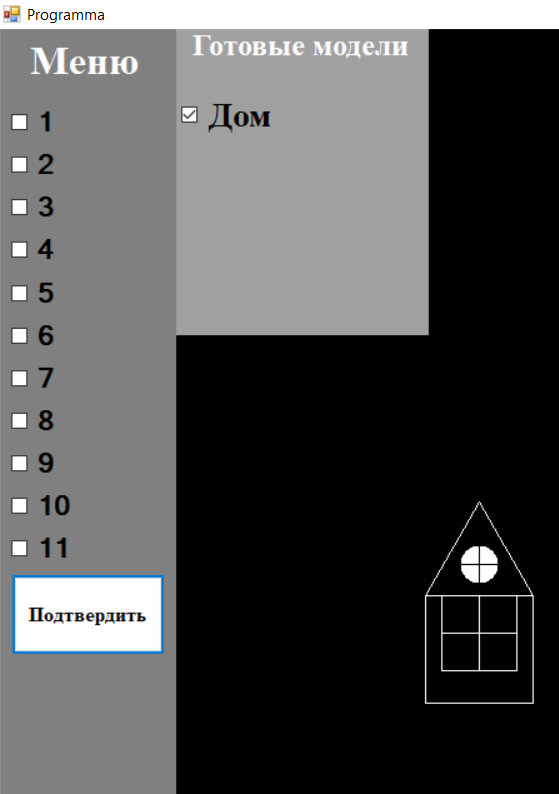


Рисунок 30 – Результат второго тестового примера

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.