ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ

Кафедра ВМиК

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

по предмету **«Программирование на С++»**

Выполнил: студент группы МО-203Б

Гиндуллин Р. И.

Проверил:  
доцент каф. ВМиК

Котельников В.А.

**Уфа 2025 г.**

# **Цель лабораторной работы**

В рамках лабораторной работы необходимо разобраться:

* каким образом создаётся интерфейс приложения из стандартных объектов;
* каким образом изменяются до и после запуска программы свойства объектов;
* каким образом одни объекты вызывают методы других объектов;

в каких случаях срабатывают какие события из стандартного списка

# **Задание**

* Создание простейшего приложения с GUI
  + форма, меню, пункты меню
  + добавлять на форму разнообразные стандартные объекты и изучать их свойства, влияющие на внешний вид и поведение
* Управление простейшими событиями (при происходящих событиях должна появляться какая-либо реакция приложения, позволяющая понять, что событие произошло)
  + нажатие/отпускание мыши, перемещение мыши, нажатие/отпускание клавиши
  + выбор пункта меню
  + перерисовка (как и для чего использовать событие Paint – обратите внимание, не надо делать программу Paint, надо разобраться, как использовать событие Paint: показать это событие в списке событий, разобраться, когда оно происходит и как его использовать)
  + таймер (как использовать событие таймера для организации задержек)
  + изменение размера окна (как и для чего использовать событие SizeChanged)
* Управление событиями, влияние объектов друг на друга (при каком-либо событии, связанным с одним объектом, должно что-то меняться в другом объекте).
* Управление событиями и обработчиками событий: один обработчик на несколько событий разных объектов (например, три различные кнопки, один обработчик, нажатая кнопка меняет цвет), программный вызов обработчиков (вызов метода-обработчика в программе), программный вызов событий (вызов методов типа PerformClick).
* Управление свойствами объектов в design time и в runtime: выделить два главных свойства каждого объекта и научиться изменять их как до запуска программы, так и во время работы приложения.
* Изучить (знать, для каких целей используются и как работают) все стандартные элементы управления:
  + В случае Visual Studio - Windows Forms Application: все компоненты с закладок «Стандартные элементы управления», «Контейнеры», плюс меню и таймер.
  + В случае C++ Builder - VCL Application: все компоненты с закладок «Standard», «Additional», «Win32», «System».
  + В остальных случаях – аналогичный набор основных компонент
* Для всех стандартных компонентов из предыдущего пункта необходимо понять их назначение (для чего используется чаще всего), и выделить основное свойство (одно или два) или основной метод (например, для текстового поля ввода – свойство, хранящее текст, для выпадающего списка – список элементов и номер выбранного и т.д.). Необходимо у каждого типового компонента выделить его главные свойства (одно-два) и научиться манипулировать ими до запуска программы и во время её работы.
* Динамическое создание и уничтожение объектов интерфейса:
  + создание элементов интерфейса как реакция на события: при нажатии мышкой на любое место на форме, должен создаваться объект любого типового класса (например, кнопка). Элемент обязательно должен иметь обработчик события (например, если создаются кнопки, то при нажатии на них должна появляться реакция)

# **Ход выполнения лабораторной работы**

Разработку простейшего приложения я начал с создания файла в Visual Studio “Windows Forms”, где, в первую очередь, нас встречает вкладка “Form1.cs [Конструктор]”:

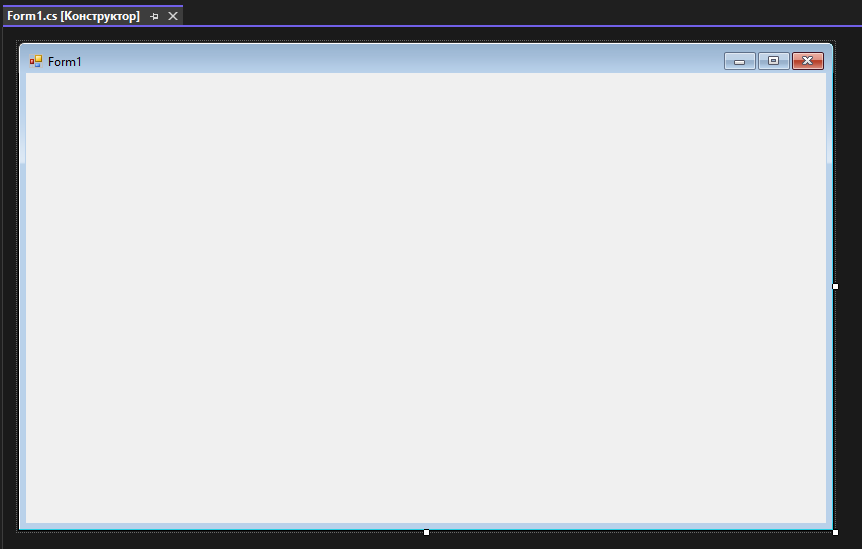


Рис. 1. Конструктор

Далее, после нажатия клавиши “F7”, нас встречает вкладка “Form1.cs”:

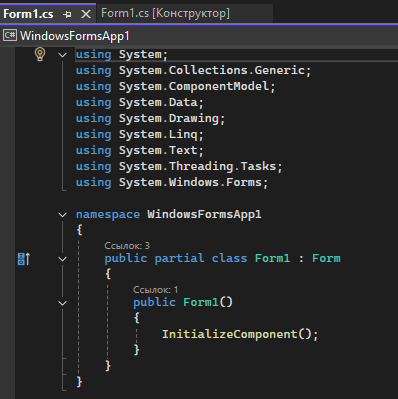
****

Рис. 2. Библиотеки

Начав разработку приложения, я открыл вкладку “Панель элементов”, где можно было воспользоваться готовыми элементами:

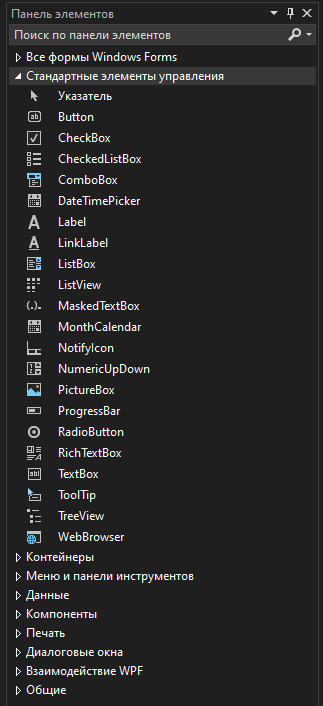


Рис. 3. Панель элементов

С помощью двойного нажатия ПКМ по нужному элементу, я выводил его на “Конструктор”, в левый верхний угол:

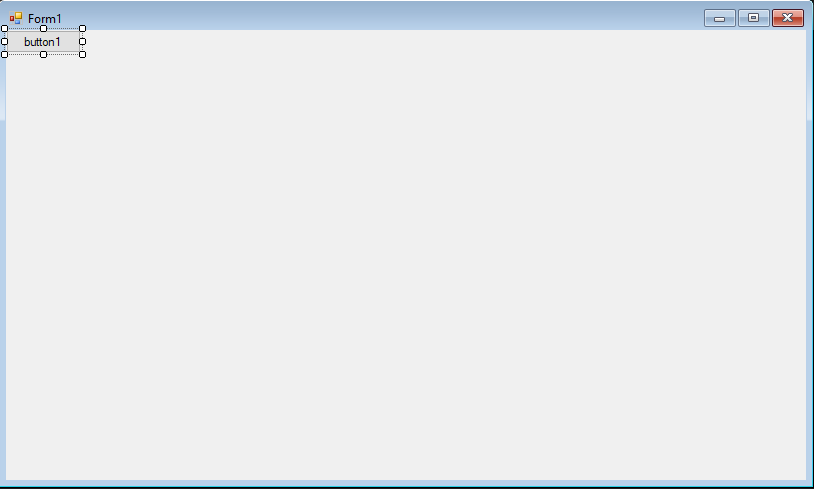


Рис. 4. Вывод элемента на конструктор

Также, для изменения названия, текста, размера, цвета кнопок и действия, на которые они откликаются, использовалась панель свойств:

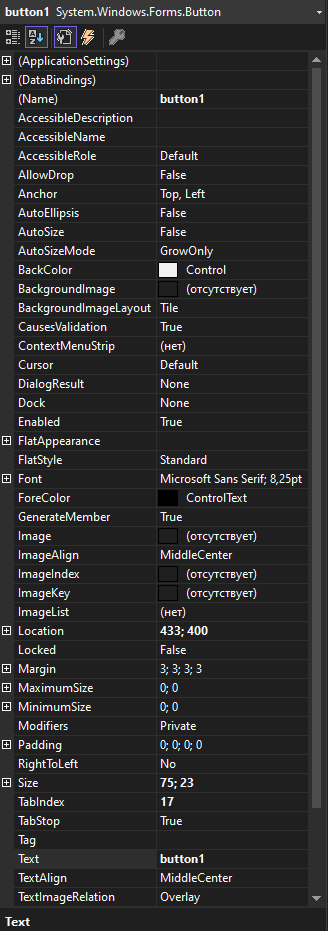


Рис. 5. Панель свойств

## **Описание создания и реализации кнопок**

**1. Кнопка Очистки (button1)**

**Назначение:** очищает содержимое текстового поля. **Функциональность:** после нажатия очищает textBox1 и обновляет label1 с сообщением "Строка очищена".

**Реализация:** Использована обработка события Click для кнопки и событие MouseDown для текстового поля, что позволяет реагировать на действия пользователя.

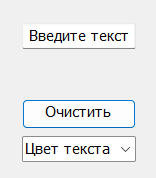


Рис. 6. Поле ввода текста

**2. Кнопка Запуска Таймера (button2)**

**Назначение:** запускает таймер обратного отсчета.

**Функциональность:** определяет количество секунд на основе значений, выбранных в comboBox, и запускает таймер, отображая оставшееся время.

**Реализация:** Использовано событие Click для запуска таймера и обработка события Tick для обновления отображаемого времени.

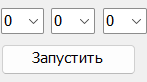


Рис. 7. Таймер

**3. Кнопка Создания Дополнительной Кнопки (button3)**

**Назначение:** динамически создает новую кнопку button4. **Функциональность:** после нажатия добавляет кнопку button4 в указанную область.

**Реализация:** Использован метод Controls.Add() для добавления элемента в форму, а также задание обработчика событий для созданной кнопки.

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 8. Динамическая кнопка

**5. Кнопка Очистки Панели (button6)**

**Назначение:** Очищает нарисованные элементы на panel3. **Функциональность:** При нажатии вызывает метод Invalidate(), который перерисовывает panel3 и удаляет все нарисованные линии.

**Реализация:** Используется метод Invalidate() для обновления графической области и сброса отрисованных элементов.

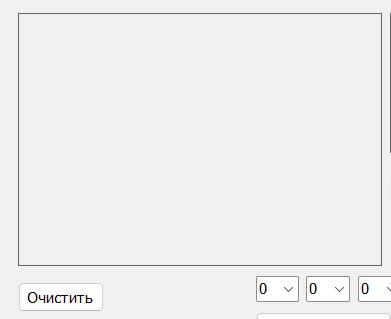


Рис. 9. Панель для рисования

**6. Мигающая панель**

**Назначение:** Панель мигает выбранным цветом.

**Функциональность:** При нажатии на цвет используется метод Color(),.

**Реализация:** Используется метод Invalidate() для обновления графической области.

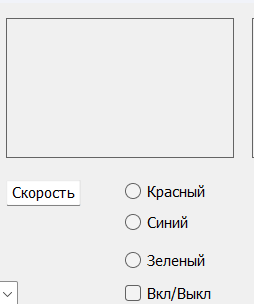


Рис. 10. Мигающая панель

**7. Панель с фигурами**

**Назначение:** рисуется выбранная фигура выбранным цветом.

**Функциональность:** При нажатии на цвет используется метод Color(),.

**Реализация:** Используется метод Graphics() для обновления графической области.

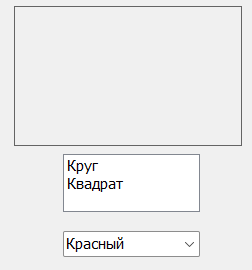


Рис.11. Панель с фигурами

**Вывод**

В ходе выполнения лабораторной работы было изучено создание простейшего desktop-приложения с графическим интерфейсом пользователя (GUI) в среде Windows Forms. В процессе разработки были освоены основные аспекты работы с формами, кнопками и другими элементами управления, а также принципы обработки событий.

Были реализованы различные компоненты интерфейса, включая кнопки с разными функциями, такие как очистка текстового поля, запуск таймера, динамическое создание и удаление элементов. Также изучены принципы работы с событиями, такими как нажатие мыши, изменение размеров окна, запуск таймера и перерисовка элементов.

Особое внимание уделено управлению свойствами объектов как в design time, так и в runtime. В ходе работы изучены стандартные элементы управления Windows Forms, их свойства и способы взаимодействия между ними. Реализована динамическая работа с элементами: создание и удаление интерфейсных объектов во время выполнения программы.

Таким образом, лабораторная работа позволила разобраться с основными принципами разработки GUI-приложений, изучить механизмы обработки событий и управления элементами интерфейса. Полученные знания могут быть использованы при создании более сложных приложений с расширенным функционалом.

**Приложение 1**

namespace WinFormsApp2

{

public partial class Form1 : Form

{

private int clickCount = 0; //счетчик кликов

private Color selectedColor = Color.Red; // цвет фигуры красный

private PictureBox clockIcon; // переменная иконки

// Таймер для отсчета времени

private System.Windows.Forms.Timer alarmTimer;

private int remainingSeconds;

private System.Windows.Forms.Timer removeIconTimer; // Таймер для удаления иконки

private System.Windows.Forms.Timer blinkTimer;

private Color currentColor;

private bool isBlinking = false; // Флаг мигания

private Button button4; // Объявляем кнопку button4

private bool isDrawing = false; // Флаг для отслеживания состояния рисования

private Point lastPoint; // Текущая позиция мыши

public Form1()

{

InitializeComponent();

;// Добавление цветов в ComboBox

comboBox1.Items.Add("Красный");

comboBox1.Items.Add("Оранжевый");

comboBox1.Items.Add("Желтый");

comboBox1.Items.Add("Зеленый");

comboBox1.Items.Add("Голубой");

comboBox1.Items.Add("Синий");

comboBox1.Items.Add("Фиолетовый");

comboBox1.Items.Add("Белый");

comboBox1.Items.Add("Черный");

// Настройка таймера для мигания

blinkTimer = new System.Windows.Forms.Timer();

blinkTimer.Interval = 100; // Интервал 500 мс для мигания

blinkTimer.Tick += BlinkTimer\_Tick; // Подписка на событие Tick

button6.Click += button6\_Click;// Устанавливаем событие для button6 (очистка)

//начальный цвет

currentColor = Color.Red; //красный по умолчанию

// Привязка события для textBox2

textBox2.TextChanged += textBox2\_TextChanged;

}

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

{

textBox1.Text = "";

label1.Text = "Строка очищена";

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

if (textBox1.Text != "")

label1.Text = "Ввод текста";

}

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

{

if (textBox1.Text != "")

label1.Text = "Есть текст";

}

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

{

// Получаем выбранный цвет из ComboBox

string selectedColor = comboBox1.SelectedItem.ToString();

// Меняем цвет текста в textBox1 в зависимости от выбранного цвета

switch (selectedColor)

{

case "Красный":

textBox1.ForeColor = Color.Red;

break;

case "Оранжевый":

textBox1.ForeColor = Color.Orange;

break;

case "Желтый":

textBox1.ForeColor = Color.Yellow;

break;

case "Зеленый":

textBox1.ForeColor = Color.Green;

break;

case "Голубой":

textBox1.ForeColor = Color.LightBlue;

break;

case "Синий":

textBox1.ForeColor = Color.Blue;

break;

case "Фиолетовый":

textBox1.ForeColor = Color.Violet;

break;

case "Белый":

textBox1.ForeColor = Color.White;

break;

case "Черный":

textBox1.ForeColor = Color.Black;

break;

}

}

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

{

if (radioButton1.Checked)

{

currentColor = Color.Red; // Устанавливаем красный цвет

}

}

private void progressBar1\_Click(object sender, EventArgs e)

{

}

private void trackBar1\_Scroll(object sender, EventArgs e)

{

progressBar1.Value = trackBar1.Value;

trackBar1.Minimum = 0;

trackBar1.Maximum = 100;

}

private void timer1\_Tick(object sender, EventArgs e)

{

if (remainingSeconds > 0)

{

remainingSeconds--; // Отнимаем секунду

int hoursLeft = remainingSeconds / 3600;

int minutesLeft = (remainingSeconds % 3600) / 60;

int secondsLeft = remainingSeconds % 60;

// Обновляем текст на экране

label2.Text = $"{hoursLeft:D2}:{minutesLeft:D2}:{secondsLeft:D2} осталось";

}

else

{

// Таймер завершен

alarmTimer.Stop();

label2.Text = "Время пришло!";

// Показываем иконку времени

clockIcon = new PictureBox();

Icon customIcon = new Icon("C:\\Users\\qpasa\\Desktop\\clock.ico"); // Указывайте путь к файлу

clockIcon.Image = customIcon.ToBitmap(); // Преобразуем Icon в Bitmap и устанавливаем его в PictureBox

clockIcon.Size = new Size(200, 200);

clockIcon.Location = new Point(690, 348); // Устанавливаем позицию иконки на форме

this.Controls.Add(clockIcon); // Добавляем иконку на форму

// Запускаем таймер для удаления иконки через 2 секунды

removeIconTimer.Start();

}

}

private void button2\_Click(object sender, EventArgs e)

{

// Получаем количество секунд для будильника

int hours = int.Parse(comboBox2.SelectedItem.ToString());

int minutes = int.Parse(comboBox3.SelectedItem.ToString());

int seconds = int.Parse(comboBox4.SelectedItem.ToString());

// Вычисляем общее количество секунд

remainingSeconds = hours \* 3600 + minutes \* 60 + seconds;

// Запускаем таймер

alarmTimer.Start();

label2.Text = "Таймер запущен";

}

private void panel1\_Paint(object sender, PaintEventArgs e)

{

// Получаем текущий выбранный элемент в listBox1

string selectedShape = listBox1.SelectedItem?.ToString();

// Рисуем фигуру в зависимости от выбранного элемента

if (selectedShape == "Круг")

{

// Рисуем круг с выбранным цветом

Graphics g = e.Graphics;

int diameter = 98; // Диаметр круга

int x = (panel1.Width - diameter) / 2; // Центрируем круг по панели по оси X

int y = (panel1.Height - diameter) / 2; // Центрируем круг по панели по оси Y

g.FillEllipse(new SolidBrush(selectedColor), x, y, diameter, diameter); // Рисуем круг с выбранным цветом

}

else if (selectedShape == "Квадрат")

{

// Рисуем квадрат с выбранным цветом

Graphics g = e.Graphics;

int size = 98;//Размер квадрата

int x = (panel1.Width - size) / 2; // Центрируем квадрат по оси X

int y = (panel1.Height - size) / 2; // Центрируем квадрат по оси Y

g.FillRectangle(new SolidBrush(selectedColor), x, y, size, size); // Рисуем квадрат с выбранным цветом

}

}

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

{

this.MouseClick += new MouseEventHandler(Form1\_MouseClick);

// Инициализация компонентов, если необходимо

label3.Text = "Количество кликов: 0"; // Начальный текст для отображения

// Устанавливаем начальный цвет

comboBox5.SelectedIndex = 0;

// Заполнение ComboBox для времени

for (int i = 0; i < 24; i++)

{

comboBox2.Items.Add(i.ToString("D2")); // Часы (от 00 до 23)

}

for (int i = 0; i < 60; i++)

{

comboBox3.Items.Add(i.ToString("D2")); // Минуты (от 00 до 59)

comboBox4.Items.Add(i.ToString("D2")); // Секунды (от 00 до 59)

}

// Устанавливаем начальные значения

comboBox2.SelectedIndex = 0;

comboBox3.SelectedIndex = 0;

comboBox4.SelectedIndex = 0;

// Инициализация таймера

alarmTimer = new System.Windows.Forms.Timer();

alarmTimer.Interval = 1000; // Таймер срабатывает каждую секунду

alarmTimer.Tick += timer1\_Tick;

// Инициализация таймера для удаления иконки

removeIconTimer = new System.Windows.Forms.Timer();

removeIconTimer.Interval = 2000; // Устанавливаем интервал в 2 секунды

removeIconTimer.Tick += removeIconTimer\_Tick;

// Устанавливаем событие для panel3

panel3.MouseDown += panel3\_MouseDown;

panel3.MouseMove += panel3\_MouseMove;

panel3.MouseUp += panel3\_MouseUp;

}

private void removeIconTimer\_Tick(object sender, EventArgs e)

{

// Останавливаем таймер удаления иконки

removeIconTimer.Stop();

// Удаляем иконку с формы

if (clockIcon != null)

{

this.Controls.Remove(clockIcon);

clockIcon.Dispose(); // Освобождаем память

}

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

// Когда пользователь выбирает элемент, перерисовываем панель

panel1.Invalidate(); // Вызываем событие Paint для перерисовки

}

private void comboBox5\_SelectedIndexChanged(object sender, EventArgs e)

{

// Меняем цвет фигуры из combobox5

string selectedColorName = comboBox5.SelectedItem.ToString();

switch (selectedColorName)

{

case "Красный":

selectedColor = Color.Red;

break;

case "Оранжевый":

selectedColor = Color.Orange;

break;

case "Желтый":

selectedColor = Color.Yellow;

break;

case "Зеленый":

selectedColor = Color.Green;

break;

case "Голубой":

selectedColor = Color.LightBlue;

break;

case "Синий":

selectedColor = Color.Blue;

break;

case "Фиолетовый":

selectedColor = Color.Violet;

break;

case "Белый":

selectedColor = Color.White;

break;

case "Черный":

selectedColor = Color.Black;

break;

}

// Перерисовываем панель для изменения цвета фигуры

panel1.Invalidate();

}

private void label3\_Click(object sender, EventArgs e)

{

}

// MouseClick для отслеживания кликов левой кнопкой мыши

private void Form1\_MouseClick(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left) // Проверяем, что клик был ЛКМ

{

clickCount++; // Увеличиваем счетчик кликов

label3.Text = $"Количество кликов: {clickCount}"; // Обновляем текст на метке

}

}

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

{

if (checkBox1.Checked)

{

isBlinking = true; // Включаем мигание

blinkTimer.Start(); // Запускаем таймер

}

else

{

isBlinking = false; // Выключаем мигание

blinkTimer.Stop(); // Останавливаем таймер

panel2.BackColor = currentColor; // Устанавливаем постоянный цвет

}

}

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

{

if (radioButton2.Checked)

{

currentColor = Color.Blue; // Устанавливаем синий цвет

}

}

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

{

if (radioButton3.Checked)

{

currentColor = Color.Green; // Устанавливаем зеленый цвет

}

}

private void BlinkTimer\_Tick(object sender, EventArgs e)

{

if (isBlinking)

{

if (panel2.BackColor == currentColor)

{

panel2.BackColor = Color.Transparent; // Прозрачный цвет (мигающий эффект)

}

else

{

panel2.BackColor = currentColor; // Устанавливаем выбранный цвет

}

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

int newInterval;

// Попытка преобразовать введенный текст в число

if (int.TryParse(textBox2.Text, out newInterval))

{

// Убедимся, что интервал больше нуля, иначе установим минимальное значение

if (newInterval > 0)

{

blinkTimer.Interval = newInterval; // Обновляем интервал таймера

}

}

else

{

// Если введено некорректное значение, игнорируем изменения

// Можно также поставить обработку ошибки или показать сообщение

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (button4 == null) // Проверяем, не существует ли уже button4

{

// Создаем новую кнопку button4

button4 = new Button();

button4.Size = new Size(85, 25); // Размер кнопки

button4.Location = new Point(47, 453); // Позиция кнопки

button4.Text = "Кнопка"; // Текст на кнопке

// Добавляем обработчик для клика на button4

button4.Click += Button4\_Click;

// Добавляем кнопку на форму

this.Controls.Add(button4);

}

}

private void button5\_Click(object sender, EventArgs e)

{

if (button4 != null) // Если button4 существует

{

// Удаляем button4 с формы

this.Controls.Remove(button4);

button4.Dispose(); // Освобождаем память

button4 = null; // Сбрасываем ссылку на кнопку

}

}

// Обработчик для клика на button4

private void Button4\_Click(object sender, EventArgs e)

{

MessageBox.Show("Вы нажали на Кнопку!");

}

private void panel3\_MouseDown(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left) // Проверка на левую кнопку мыши

{

isDrawing = true; // Включаем рисование

lastPoint = e.Location; // Сохраняем начальную точку

}

}

private void panel3\_MouseMove(object sender, MouseEventArgs e)

{

if (isDrawing) // Если рисуем

{

using (Graphics g = panel3.CreateGraphics())

{

g.DrawLine(Pens.Black, lastPoint, e.Location); // Рисуем линию между предыдущей и текущей точкой

}

lastPoint = e.Location; // Обновляем точку для следующей линии

}

}

private void panel3\_MouseUp(object sender, MouseEventArgs e)

{

isDrawing = false; // Останавливаем рисование

}

private void button6\_Click(object sender, EventArgs e)

{

{

panel3.Invalidate(); // Очищаем панель (перерисовываем её)

}

}

}

}