Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

" Разработка программы для определения доминирующего цвета изображения в дизайне на С++"

Техническое задание

Листов 30

Выполнил студент гр. РИС-24-2б

Иванова Елена Олеговна

г. Пермь, 2025

Содержание

[1 Введение 3](#_Toc1956637434)

[1.1 Наименование программы 3](#_Toc626218503)

[1.2 Краткая характеристика области применения 3](#_Toc1415076890)

[2 Основания для проведения разработки 3](#_Toc1221269048)

[3 Назначение АДЦ 3](#_Toc2007101155)

[3.1 Функциональное назначение 3](#_Toc301427629)

[3.2 Эксплуатационное назначение 3](#_Toc1571194748)

[4 Требования к АДЦ 4](#_Toc587570727)

[4.1 Требования к функциональным характеристикам 4](#_Toc170033779)

[4.1.1 Требования к составу выполняемых функций 4](#_Toc904569944)

[4.1.2 Требование к организации входных данных 4](#_Toc418011283)

[4.1.3 Требование к организации выходных данных 4](#_Toc913629244)

[5 UML-диаграмма классов 4](#_Toc288531380)

[6 Используемые инструменты и технологии 5](#_Toc479426792)

[7 Код программы 6](#_Toc1460975753)

[8 Примеры работы программы 7](#_Toc637582012)

[9 Заключение 7](#_Toc1227725793)

[10 Список используемых источников 7](#_Toc232598897)

# 1 Введение

## 1.1 Наименование программы

Наименование программы: “ Анализатор доминирующего цвета”. Далее по тексту – АДЦ.

## 1.2 Краткая характеристика области применения

АДЦ разработано для точного определения доминирующих цветов и предназначено для использования в современных дизайнерских, фотографических и реставрационных проектах.

Это решение помогает специалистам оперативно и с высокой точностью анализировать цветовые палитры, что особенно ценно для реставраторов — при восстановлении оригинальных оттенков живописи и старых фотографий и для фотографов и дизайнеров — при выполнении цветокоррекции и создании визуально гармоничных композиций.

# 2 Основания для проведения разработки

“Разработка программы для определения доминирующего цвета изображения в дизайне на С++” разрабатывается в соответствии с творческим заданием по разработке автоматизированного рабочего места специалиста, утвержденного доцентом кафедры ИТАС Поляковой О. А.

Согласно требованиям Поляковой О. А., именуемой заказчиком, исполнитель Иванова Е. О.

# 3 Назначение АДЦ

## 3.1 Функциональное назначение

Приложение "Анализатор доминирующего цвета" предоставляет следующие возможности:

1. Определение основного цвета на загруженном изображении (JPG, PNG, BMP) и отображение его информации в окне.
2. Приблизительное название цвета (например, "Красный", "Бирюзовый").
3. Отображение значения в форматах RGB и HEX.
4. Сохранение обработанного изображения (JPG, PNG).
5. Отображение данных (Имя файла, Размер файла)

## 3.2 Эксплуатационное назначение

Целью работы АДЦ является быстрое и точное определения основного цвета на изображении.

# 4 Требования к АДЦ

## 4.1 Требования к функциональным характеристикам

### 4.1.1 Требования к составу выполняемых функций

Программа должна обеспечивать возможность выполнения перечисленных ниже функций.

* Анализ изображения
  + Загрузка изображений в форматах: JPG, PNG, BMP
  + Автоматическое определение доминирующего цвета
  + Вывод данных о цвете
* Дополнительные функции
  + Отображение данных файла (название, размер)
  + Сохранение обработанного изображения (JPG, PNG)

### 4.1.2 Требование к организации входных данных

*Пользователь системы*

* Файл в форматах JPG, PNG, BMP

### 4.1.3 Требование к организации выходных данных

* Приблизительное название цвета
* Цвет в форматах RGB и HEX
* Название загруженного файла и его размер
* Визуальное отображение цвета
* Визуальное отображение оригинального изображения и после обработки кодом

# 5 UML-диаграмма классов

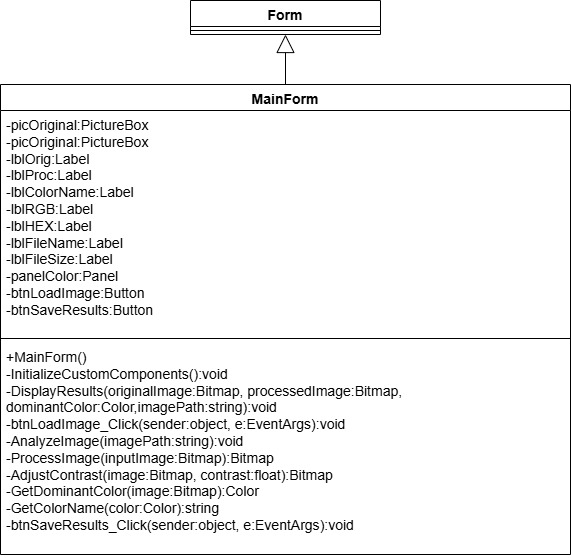
Система включает следующие классы:

* Form

класс WindowsForms

* MainForm

главная форма приложения, которая содержит все элементы пользовательского интерфейса и отвечает за логику работы программы.



# 6 Используемые инструменты и технологии

* Microsoft Visual Studio 2022
  + Основная среда разработки для создания приложения на C# (Windows Forms).
  + Использованные компоненты:
    - Windows Forms – для проектирования графического интерфейса.
    - .NET Framework 4.8 – платформа для работы приложения.
* Asеprite
  + Применялся при создании иконки приложения.
* OBS Studio
  + Программа для записи видео с экрана монитора при демонстрации работы приложения.
* Shotcut
  + Видеоредактор для монтажа записанных материалов.

# 7 Код программы

using System; using System.Collections.Generic; using System.Drawing; using System.Drawing.Imaging; using System.IO; using System.Linq; using System.Windows.Forms;

namespace ColorDetectorApp

{ public partial class MainForm : Form {

private PictureBox picOriginal;

private PictureBox picProcessed;

private Label lblOrig;

private Label lblProc;

private Label lblColorName;

private Label lblRGB;

private Label lblHEX;

private Label lblFileName;

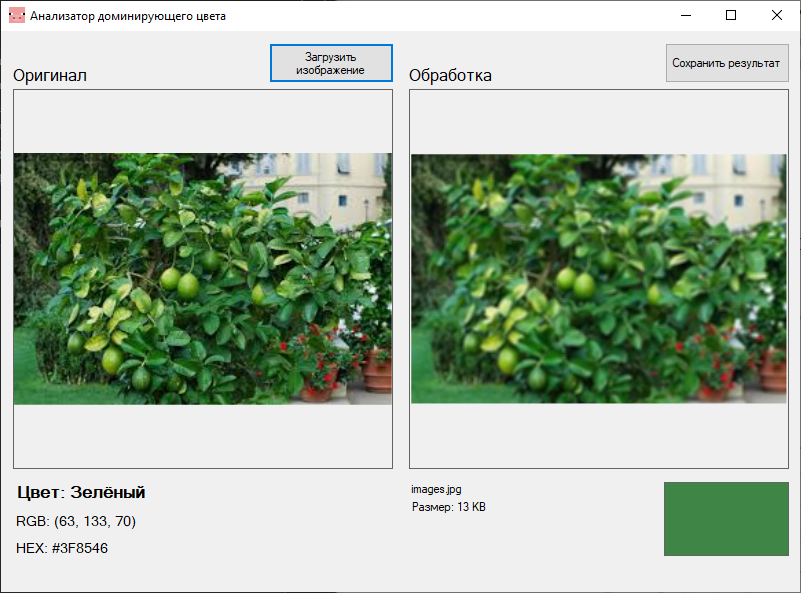
private Label lblFileSize;

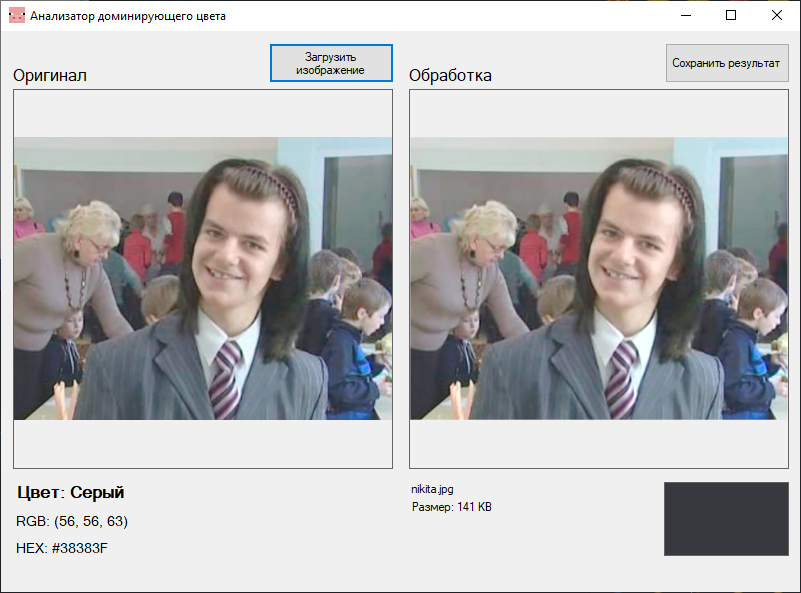
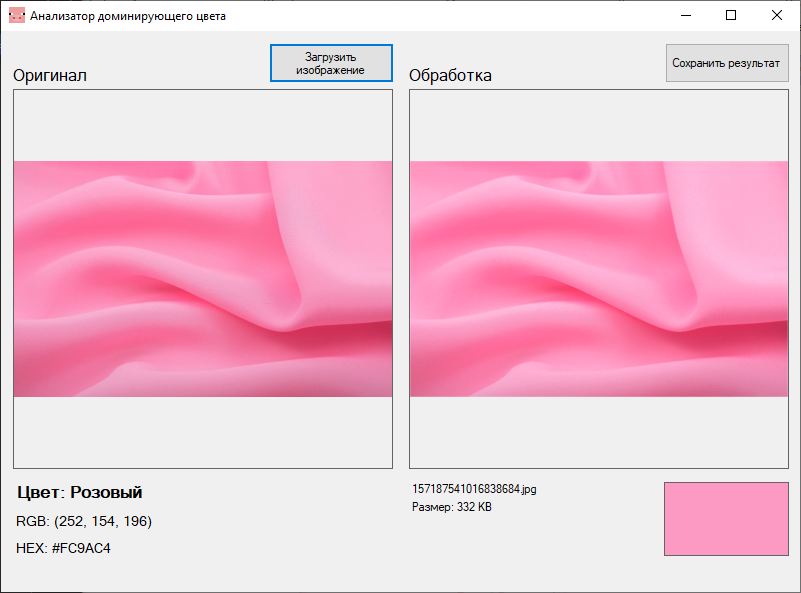
private Panel panelColor;

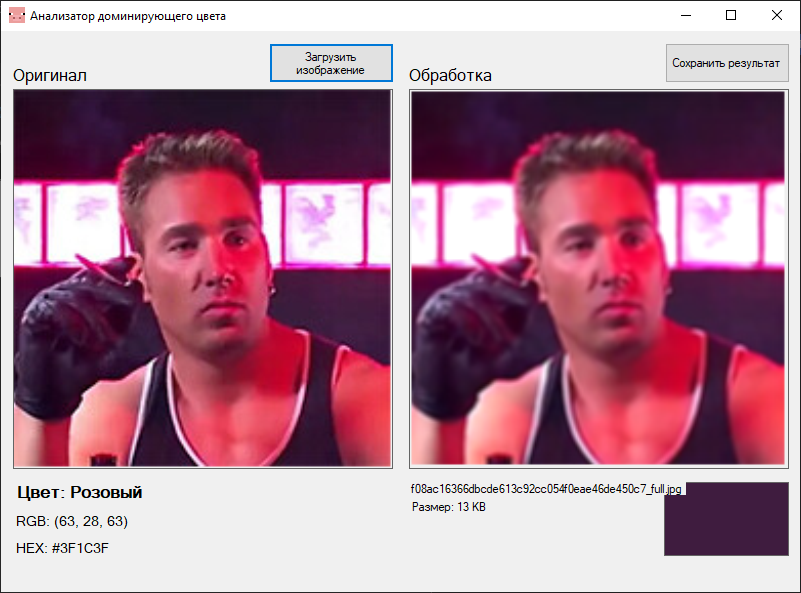
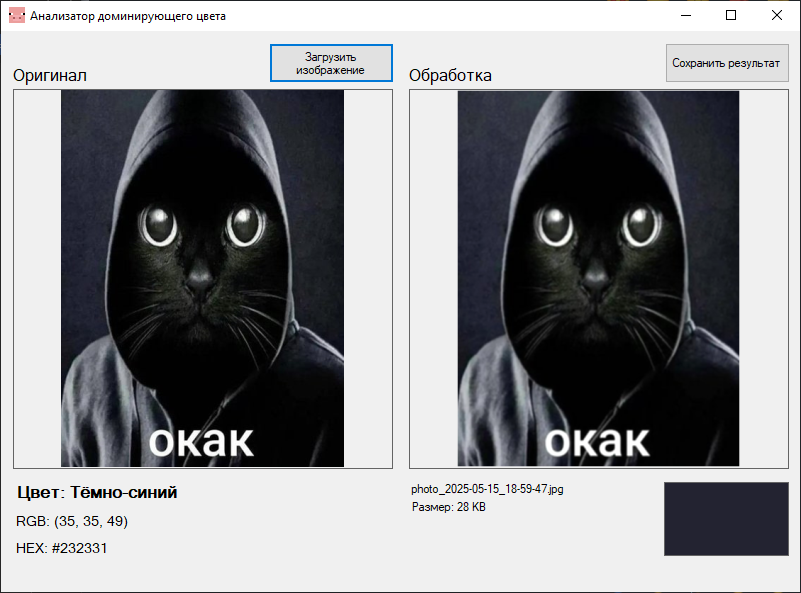
private Button btnLoadImage;

private Button btnSaveResults; public MainForm()  
 {  
 InitializeComponent();  
 InitializeCustomComponents();}  
  
 private void InitializeCustomComponents(){  
 // Инициализация всех элементов управления  
 this.picOriginal = new PictureBox();  
 this.picProcessed = new PictureBox();  
 this.lblOrig = new Label();  
 this.lblProc = new Label();  
 this.lblColorName = new Label();  
 this.lblRGB = new Label();  
 this.lblHEX = new Label();  
 this.lblFileName = new Label();  
 this.lblFileSize = new Label();  
 this.panelColor = new Panel();  
 this.btnLoadImage = new Button();  
 this.btnSaveResults = new Button();  
  
 // Настройка свойств элементов  
 // PictureBox для оригинального изображения  
 picOriginal.Location = new System.Drawing.Point(12, 58);  
 picOriginal.Size = new System.Drawing.Size(380, 380);  
 picOriginal.BorderStyle = BorderStyle.FixedSingle;  
 picOriginal.SizeMode = PictureBoxSizeMode.Zoom;  
  
 // PictureBox для обработанного изображения  
 picProcessed.Location = new System.Drawing.Point(408, 58);  
 picProcessed.Size = new System.Drawing.Size(380, 380);  
 picProcessed.BorderStyle = BorderStyle.FixedSingle;  
 picProcessed.SizeMode = PictureBoxSizeMode.Zoom;  
  
 // Надписи  
 lblOrig.Location = new System.Drawing.Point(8, 34);  
 lblOrig.Font = new Font("Microsoft Sans Serif", 12F);  
 lblOrig.AutoSize = true;  
 lblOrig.Text = "Оригинал";   
  
 lblProc.Location = new System.Drawing.Point(404, 34);  
 lblProc.Font = new Font("Microsoft Sans Serif", 12F);  
 lblProc.AutoSize = true;  
 lblProc.Text = "Обработка";  
  
 lblColorName.Location = new System.Drawing.Point(12, 451);  
 lblColorName.Font = new Font("Microsoft Sans Serif", 12F, FontStyle.Bold);  
 lblColorName.AutoSize = true;  
  
 lblRGB.Location = new System.Drawing.Point(12, 481);  
 lblRGB.Font = new Font("Microsoft Sans Serif", 10F);  
 lblRGB.AutoSize = true;  
  
 lblHEX.Location = new System.Drawing.Point(12, 508);  
 lblHEX.Font = new Font("Microsoft Sans Serif", 10F);  
 lblHEX.AutoSize = true;  
  
 lblFileName.Location = new System.Drawing.Point(408, 451);  
 lblFileName.AutoSize = true;  
  
 lblFileSize.Location = new System.Drawing.Point(408, 469);  
 lblFileSize.AutoSize = true;  
  
 // Панель для отображения цвета  
 panelColor.Location = new System.Drawing.Point(663, 451);  
 panelColor.Size = new System.Drawing.Size(125, 74);  
 panelColor.BorderStyle = BorderStyle.FixedSingle;  
  
 // Кнопки  
 btnLoadImage.Location = new System.Drawing.Point(268, 12);  
 btnLoadImage.Size = new System.Drawing.Size(125, 40);  
 btnLoadImage.Text = "Загрузить изображение";  
 btnLoadImage.Click += btnLoadImage\_Click;  
  
 btnSaveResults.Location = new System.Drawing.Point(664, 12);  
 btnSaveResults.Size = new System.Drawing.Size(125, 40);  
 btnSaveResults.Text = "Сохранить результат";  
 btnSaveResults.Click += btnSaveResults\_Click;  
  
 // Добавление элементов на форму  
 this.Controls.Add(picOriginal);  
 this.Controls.Add(picProcessed);  
 this.Controls.Add(lblOrig);  
 this.Controls.Add(lblProc);   
 this.Controls.Add(lblColorName);  
 this.Controls.Add(lblRGB);  
 this.Controls.Add(lblHEX);  
 this.Controls.Add(lblFileName);  
 this.Controls.Add(lblFileSize);  
 this.Controls.Add(panelColor);  
 this.Controls.Add(btnLoadImage);  
 this.Controls.Add(btnSaveResults);  
  
 // Настройка формы  
 this.Text = "Анализатор доминирующего цвета";  
 this.ClientSize = new System.Drawing.Size(800, 537);  
 this.MinimumSize = new Size(815, 600);  
}  
  
 private void DisplayResults(Bitmap originalImage, Bitmap processedImage, Color dominantColor, string imagePath)  
 {  
 if (originalImage != null && picOriginal != null)  
 {  
 picOriginal.Image = new Bitmap(originalImage);  
 }  
  
 if (processedImage != null && picProcessed != null)  
 {  
 picProcessed.Image = new Bitmap(processedImage);  
 }  
  
 if (lblColorName != null)  
 {  
 lblColorName.Text = $"Цвет: {GetColorName(dominantColor)}";  
 }  
  
 if (lblRGB != null)  
 {  
 lblRGB.Text = $"RGB: ({dominantColor.R}, {dominantColor.G}, {dominantColor.B})";  
 }  
  
 if (lblHEX != null)  
 {  
 lblHEX.Text = $"HEX: #{dominantColor.R:X2}{dominantColor.G:X2}{dominantColor.B:X2}";  
 }  
  
 if (panelColor != null)  
 {  
 panelColor.BackColor = dominantColor;  
 }  
  
 if (lblFileName != null && !string.IsNullOrEmpty(imagePath))  
 {  
 lblFileName.Text = Path.GetFileName(imagePath);  
 }  
  
 if (lblFileSize != null && !string.IsNullOrEmpty(imagePath) && File.Exists(imagePath))  
 {  
 lblFileSize.Text = $"Размер: {new FileInfo(imagePath).Length / 1024} KB";  
 }  
 }  
  
 private void btnLoadImage\_Click(object sender, EventArgs e)  
 {  
 using (OpenFileDialog openFileDialog = new OpenFileDialog())  
 {  
 openFileDialog.Filter = "Изображения|\*.jpg;\*.jpeg;\*.png;\*.bmp|Все файлы|\*.\*";  
 openFileDialog.Title = "Выберите изображение для анализа";  
  
 if (openFileDialog.ShowDialog() == DialogResult.OK)  
 {  
 AnalyzeImage(openFileDialog.FileName);}  
 }  
 }  
  
 private void AnalyzeImage(string imagePath) //главная хрень  
 {  
 try  
 {  
 // Загрузка изображения  
 Bitmap originalImage = new Bitmap(imagePath);  
  
 // Обработка изображения  
 Bitmap processedImage = ProcessImage(originalImage);  
  
 // Получение доминирующего цвета  
 Color dominantColor = GetDominantColor(processedImage);  
  
 // Отображение результатов  
 DisplayResults(originalImage, processedImage, dominantColor, imagePath);  
 }  
 catch (Exception ex)  
 {  
 MessageBox.Show($"Ошибка при анализе изображения: {ex.Message}", "Ошибка",  
 MessageBoxButtons.OK, MessageBoxIcon.Error);  
 }  
 }  
  
 private Bitmap ProcessImage(Bitmap inputImage)  
 {  
 Bitmap processed = new Bitmap(inputImage.Width, inputImage.Height);  
  
 // Упрощенное сглаживание  
 for (int i = 1; i < inputImage.Width - 1; i++)  
 {  
 for (int j = 1; j < inputImage.Height - 1; j++)  
 {  
 // Простое усреднение 3x3  
 Color c1 = inputImage.GetPixel(i - 1, j - 1);  
 Color c2 = inputImage.GetPixel(i, j - 1);  
 Color c3 = inputImage.GetPixel(i + 1, j - 1);  
 Color c4 = inputImage.GetPixel(i - 1, j);  
 Color c5 = inputImage.GetPixel(i, j);  
 Color c6 = inputImage.GetPixel(i + 1, j);  
 Color c7 = inputImage.GetPixel(i - 1, j + 1);  
 Color c8 = inputImage.GetPixel(i, j + 1);  
 Color c9 = inputImage.GetPixel(i + 1, j + 1);  
  
 int r = (c1.R + c2.R + c3.R + c4.R + c5.R + c6.R + c7.R + c8.R + c9.R) / 9;  
 int g = (c1.G + c2.G + c3.G + c4.G + c5.G + c6.G + c7.G + c8.G + c9.G) / 9;  
 int b = (c1.B + c2.B + c3.B + c4.B + c5.B + c6.B + c7.B + c8.B + c9.B) / 9;  
  
 processed.SetPixel(i, j, Color.FromArgb(r, g, b));  
 }  
 }  
  
 // увеличиваем контрастность  
 processed = AdjustContrast(processed, 1.05f);  
 return processed;  
 }  
  
 private Bitmap AdjustContrast(Bitmap image, float contrast)  
 {  
 Bitmap adjustedImage = new Bitmap(image.Width, image.Height);  
  
 // Матрица контрастности  
 float[][] colorMatrixElements = {  
 new float[] {contrast, 0, 0, 0, 0},  
 new float[] {0, contrast, 0, 0, 0},  
 new float[] {0, 0, contrast, 0, 0},  
 new float[] {0, 0, 0, 1, 0},  
 new float[] {0, 0, 0, 0, 1}  
 };  
  
 ColorMatrix colorMatrix = new ColorMatrix(colorMatrixElements);  
  
 using (ImageAttributes attributes = new ImageAttributes())  
 {  
 attributes.SetColorMatrix(colorMatrix);  
  
 using (Graphics g = Graphics.FromImage(adjustedImage))  
 {  
 g.DrawImage(image, new Rectangle(0, 0, image.Width, image.Height),  
 0, 0, image.Width, image.Height, GraphicsUnit.Pixel, attributes);} }  
  
 return adjustedImage;}  
  
 private Color GetDominantColor(Bitmap image)  
 {  
 // Уменьшение размера для ускорения обработки  
 Bitmap resized = new Bitmap(image, new Size(100, 100));  
  
 // Словарь для подсчета цветов  
 Dictionary<Color, int> colorCounts = new Dictionary<Color, int>();  
  
 // Проход по всем пикселям  
 for (int x = 0; x < resized.Width; x++){  
 for (int y = 0; y < resized.Height; y++)  
 {  
 Color pixelColor = resized.GetPixel(x, y);  
  
 // Игнорируем слишком светлые и темные пиксели  
 if (pixelColor.GetBrightness() < 0.15f || pixelColor.GetBrightness() > 0.85f)  
 continue;  
  
 // Квантование цвета для уменьшения вариативности  
 Color quantizedColor = Color.FromArgb(  
 pixelColor.R / 7 \* 7,  
 pixelColor.G / 7 \* 7,  
 pixelColor.B / 7 \* 7);  
  
 if (colorCounts.ContainsKey(quantizedColor))  
 colorCounts[quantizedColor]++;  
 else  
 colorCounts[quantizedColor] = 1;  
 }  
 }  
  
 // Находим наиболее часто встречающийся цвет  
 if (colorCounts.Count == 0)  
 return Color.White; // Возвращаем belый по умолчанию  
  
 var dominantColor = colorCounts.OrderByDescending(pair => pair.Value).First().Key;  
 return dominantColor;  
 }  
  
 private string GetColorName(Color color)  
 {  
 int r = color.R;  
 int g = color.G;  
 int b = color.B;  
  
 // Определяем максимальный и минимальный каналы  
 int max = Math.Max(r, Math.Max(g, b));  
 int min = Math.Min(r, Math.Min(g, b));  
 int diff = max - min;  
  
 // Яркость (0-1)  
 float brightness = (r + g + b) / 765f;  
  
 // Насыщенность (0-1)  
 float saturation = max == 0 ? 0 : diff / (float)max;  
  
 // ахроматические цвета (белый, серый, черный)  
 if (diff < 10)  
 {  
 if (brightness > 0.8f) return "Белый";  
 if (brightness < 0.2f) return "Чёрный";  
 return "Серый";}  
  
 if (r > 220 && g > 220 && b > 220) return "Белый";  
 if (r < 35 && g < 35 && b < 35) return "Чёрный";  
  
 //другие цвета  
 if (r > 180 && g > 160 && b < 100){  
 if (b < 50) return "Жёлтый";  
 return "Светло-жёлтый";  
 }  
  
 if (r > 110 && g < 160 && b < 80)  
 {  
 if (r > 180 && g > 120 && b < 60) return "Оранжевый";   
 if (r <= 180 && g > 60) return "Коричневый";  
 }  
  
 if (r > 100 && g > 200 && b > 180 && b < 230) return "Бирюзовый";   
 if (r > 210 && g > 150 && g < 210 && b > 170) return "Розовый";  
 if (r > 140 && r < 230 && g < 80 && b > 140 && b < 230) return "Пурпурный";   
  
 // проверяем основные цветовые группы  
 if (max == r){  
 if (g > b + 50 && r > g + 30) return "Оранжевый";  
 if (g > b + 30) return "Красно-оранжевый";  
 if (b > g + 50 && saturation > 0.6f) return "Пурпурный";  
 if (b > g + 30) return "Розовый";   
 if (r > 160 && g < 100 && b < 100) return "Красный";   
 }  
 else if (max == b)  
 {  
 if (g > r + 40) return "Голубой";  
 if (r >= g + 30 && saturation > 0.5f) return "Фиолетовый";  
 if (r > g + 30) return "Сиреневый";  
 if (brightness > 0.8f) return "Светло-синий";  
 if (brightness < 0.3f) return "Тёмно-синий";  
 if (r < 100 && g < 100 && b > 160) return "Синий";  
 }  
 else if (max == g)  
 {  
 if (r > b + 40 && g > r + 30) return "Лаймовый";  
 if (r > b + 30) return "Жёлто-зелёный";  
 if (b > r + 60 && saturation > 0.6f) return "Бирюзовый";  
 if (brightness > 0.8f) return "Светло-зелёный";  
 if (brightness < 0.3f) return "Тёмно-зелёный";  
 if (r < 100 && g >130 && b < 100) return "Зелёный";}  
 return "Неопределённый";}  
  
 private void btnSaveResults\_Click(object sender, EventArgs e){  
 if (picProcessed.Image == null){  
 MessageBox.Show("Нет обработанного изображения для сохранения", "Информация",  
 MessageBoxButtons.OK, MessageBoxIcon.Information);  
 return;  
 }  
 using (SaveFileDialog saveDialog = new SaveFileDialog())  
 {  
 saveDialog.Filter = "JPEG Image|\*.jpg|PNG Image|\*.png";  
 saveDialog.Title = "Сохранить обработанное изображение";  
  
 if (saveDialog.ShowDialog() == DialogResult.OK){  
 try{  
 picProcessed.Image.Save(saveDialog.FileName);  
 MessageBox.Show("Изображение успешно сохранено", "Успех",  
 MessageBoxButtons.OK, MessageBoxIcon.Information); }  
 catch (Exception ex){  
 MessageBox.Show($"Ошибка при сохранении: {ex.Message}", "Ошибка",  
 MessageBoxButtons.OK, MessageBoxIcon.Error);}}}}}}

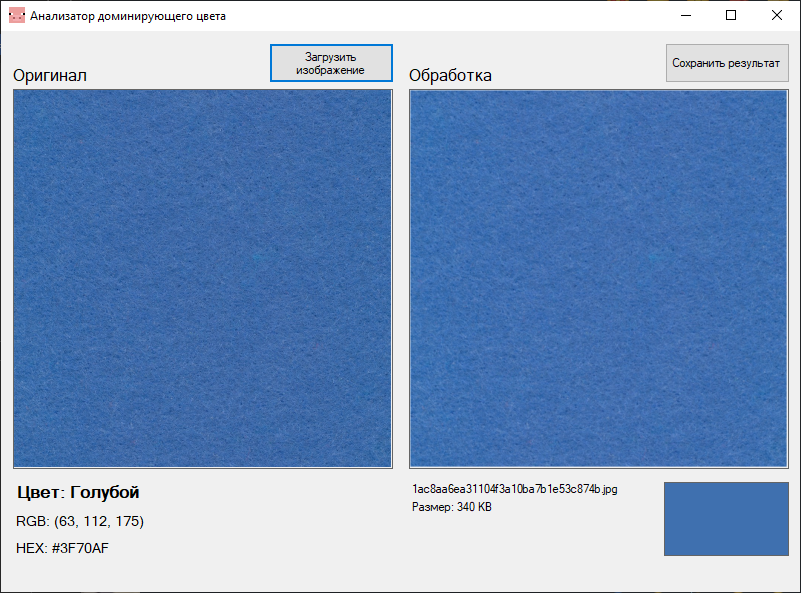
# 8 Примеры работы программы

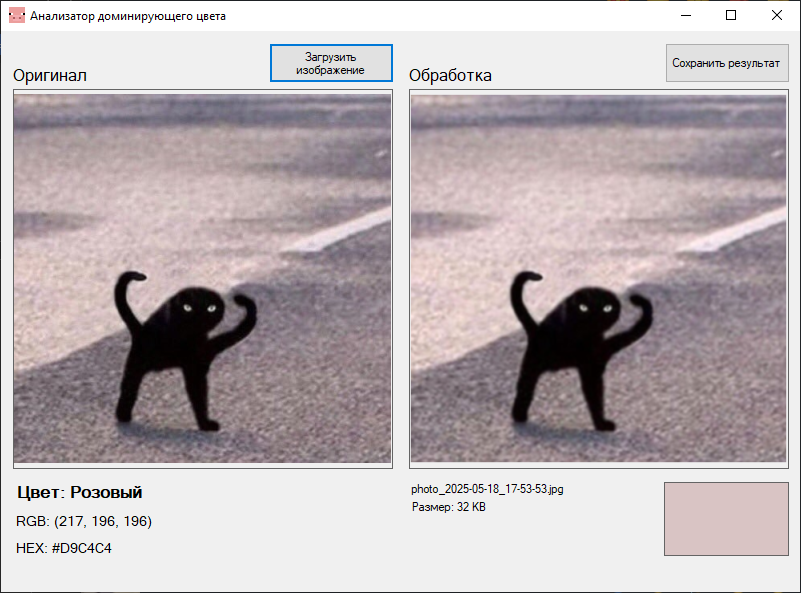


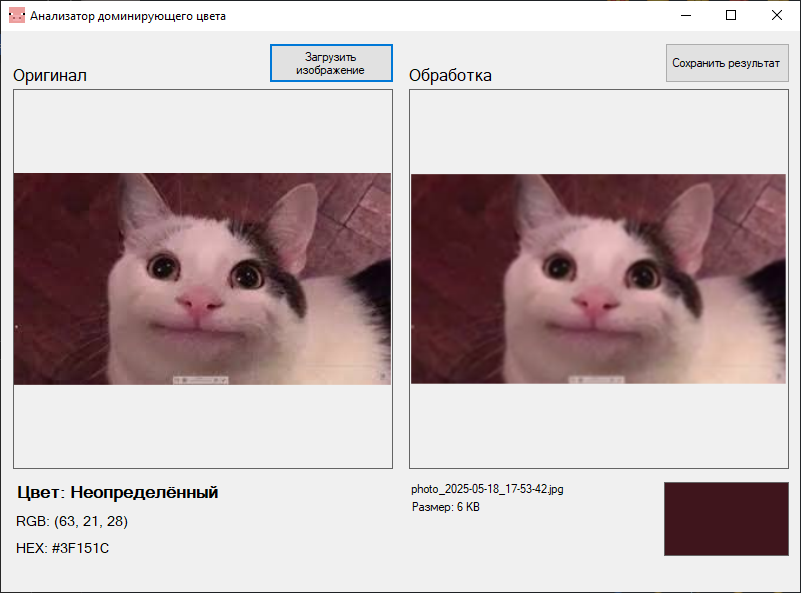


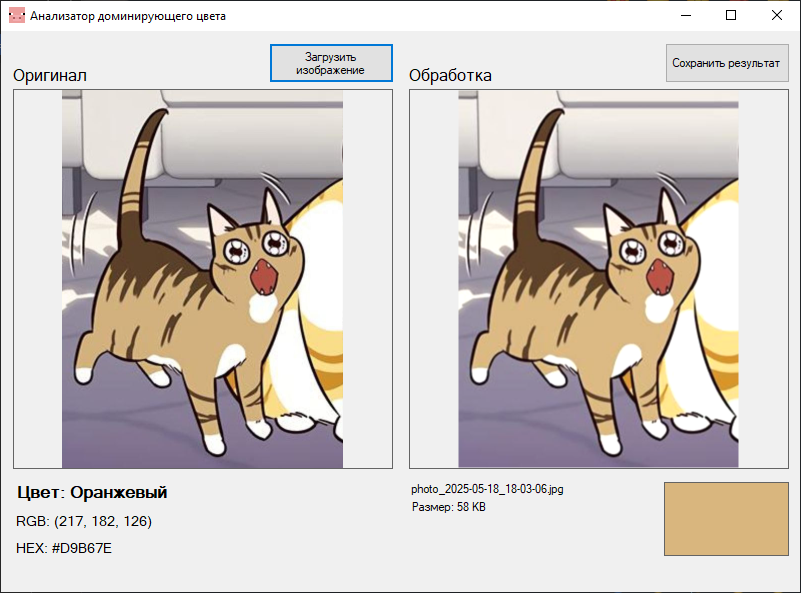


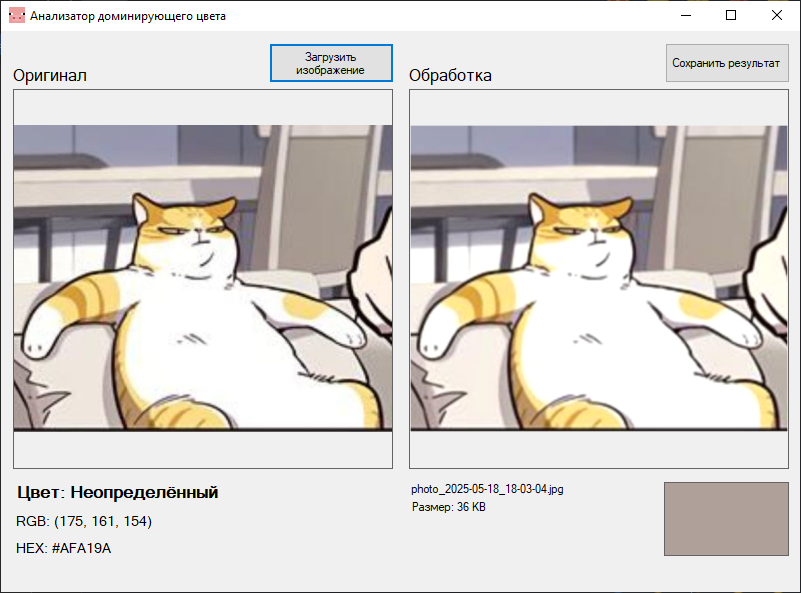


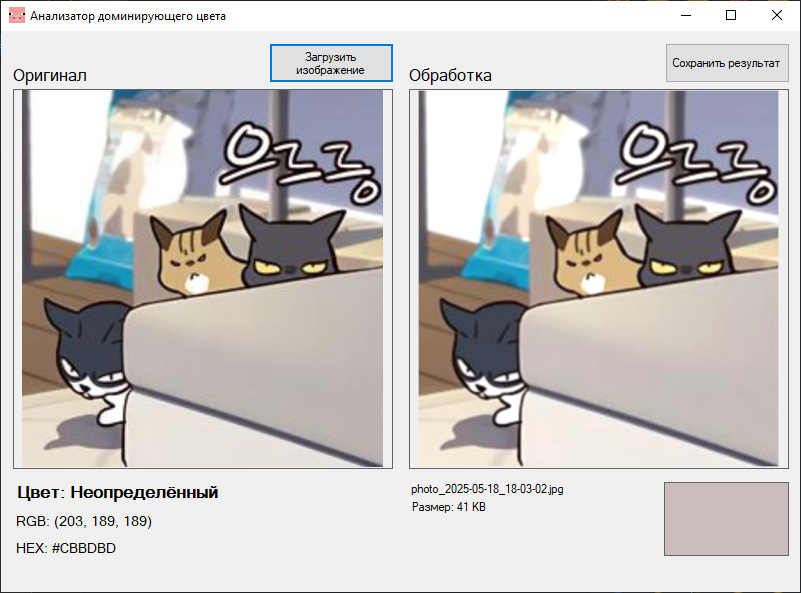


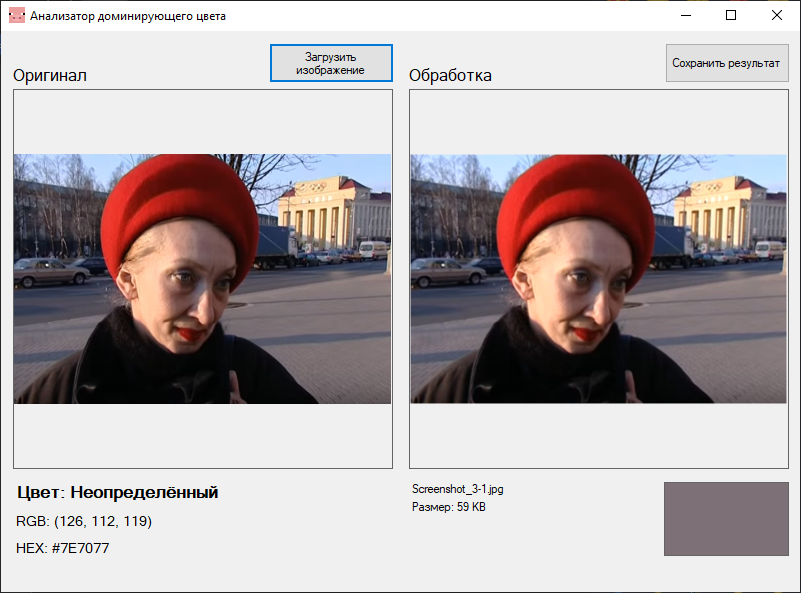


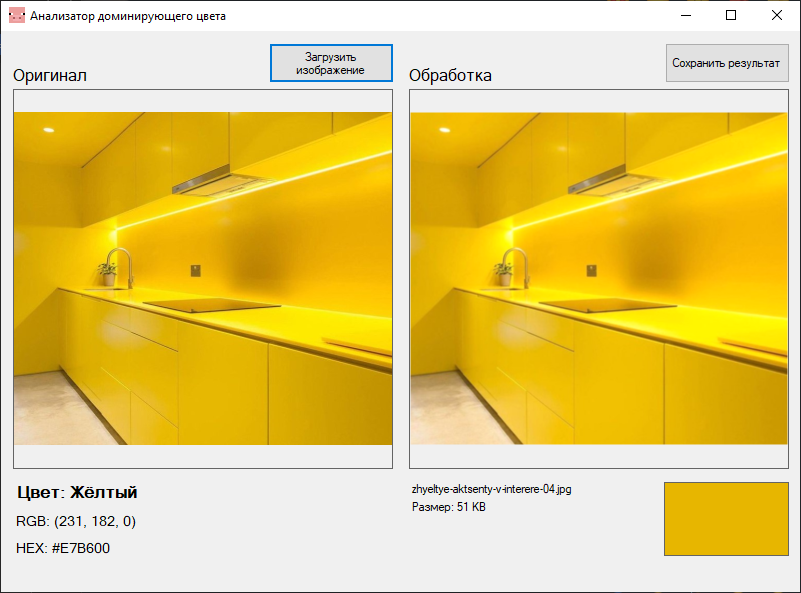












# 9 Заключение

Разработанное приложение "Анализатор доминирующего цвета" успешно решает поставленные задачи:

1. Автоматизация анализа цвета – определение доминирующего оттенка занимает секунды вместо ручного подбора.
2. Точность обработки – использование алгоритмов сглаживания и повышения контрастности для корректного результата.
3. Удобство использования – простой интерфейс с выводом названия цвета, HEX- и RGB-кодов.

Перспективы развития:

1. Определение нескольких доминирующих цветов
2. Улучшение алгоритма поиска доминирующего цвета
3. Увеличение количества названий цветов

# 10 Список используемых источников

* <https://media.contented.ru/glossary/hex-kod/>
* https://sky.pro/wiki/javascript/rgb-cveta-kak-rabotaet-vybor-i-sravnenie-s-hex/