Название работы: Работа с цветами

Цель работы: Целью данной работы является изучение базовых операций над цветовыми каналами изображений и реализация некоторых фильтров на их основе.

Постановка задачи:

Необходимо разработать приложение Windows Forms, способное осуществлять:

1. загрузку и отображение двух изображений по выбору пользователя;
2. возможность применения базовых операций к загруженным изображениям;
3. возможность применения оконных и комбинированных фильтров к загруженным изображениям;

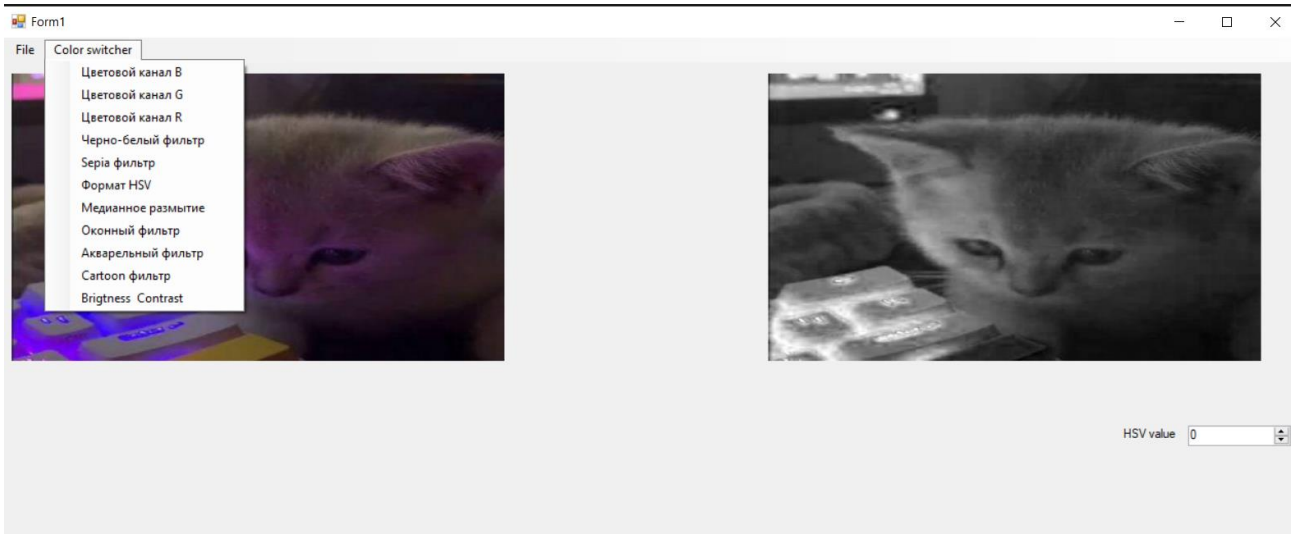
Задание:

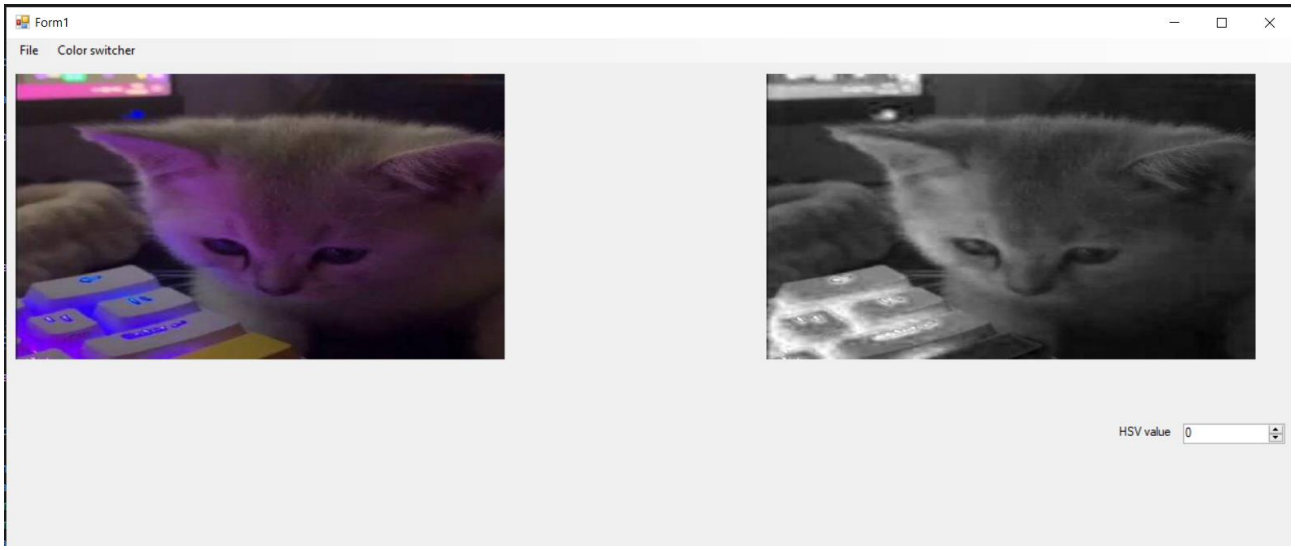
Реализовать программное средство, позволяющее отображать в одном окне два изображения, «оригинальное» слева и «результат обработки» справа. Реализовать интерфейс, позволяющий по нажатию на соответствующие кнопки выполнять следующие операции:

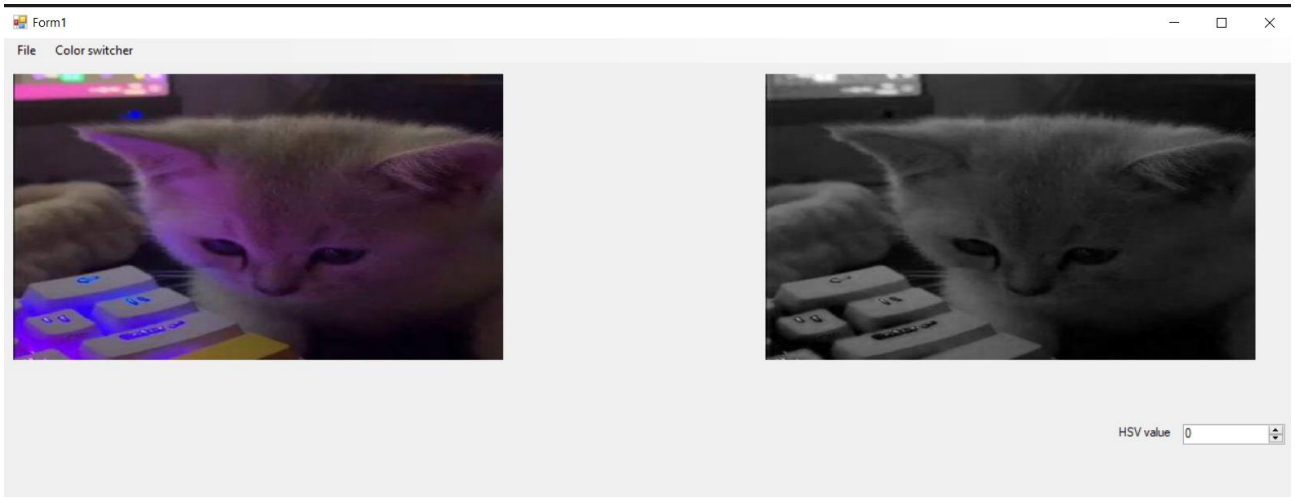
1. Вывод значений одного из трёх цветовых каналов по выбору пользователя.
2. Вывод чёрно-белой версии изображения.
3. Вывод Sepia версии изображения.
4. Вывод изображения с возможностью изменения его яркости и контраста.
5. Вывод результатов логических операций «дополнение», «исключение» и «пересечение», с возможностью выбора изображения для соответствующей операции.
6. Вывод изображения преобразованного в формат HSV, с возможностью изменения значений HSV.
7. Вывод изображений с применением к ним медианного размытия.

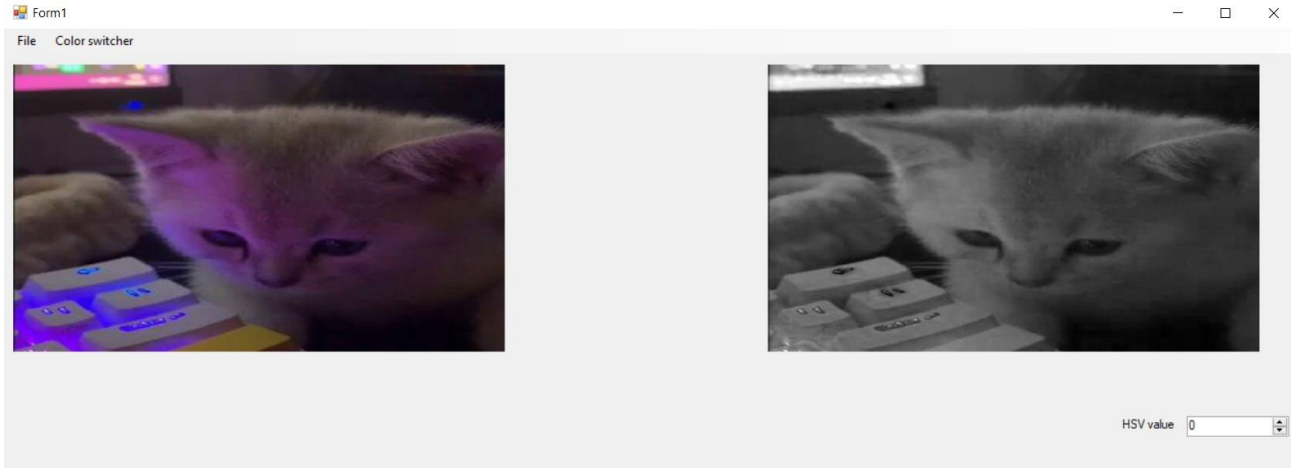
8. Вывод изображений с применением к ним оконного фильтра, с возможностью
изменения матрицы фильтра из формы приложения.
9. Вывод изображений с применением к ним «акварельного фильтра», а так же, возможностью выбора яркости, контраста и параметров смешивания изображений.
10. Вывод изображений с применением к ним «cartoon filter» и возможностью изменения
порога преобразования изображения.

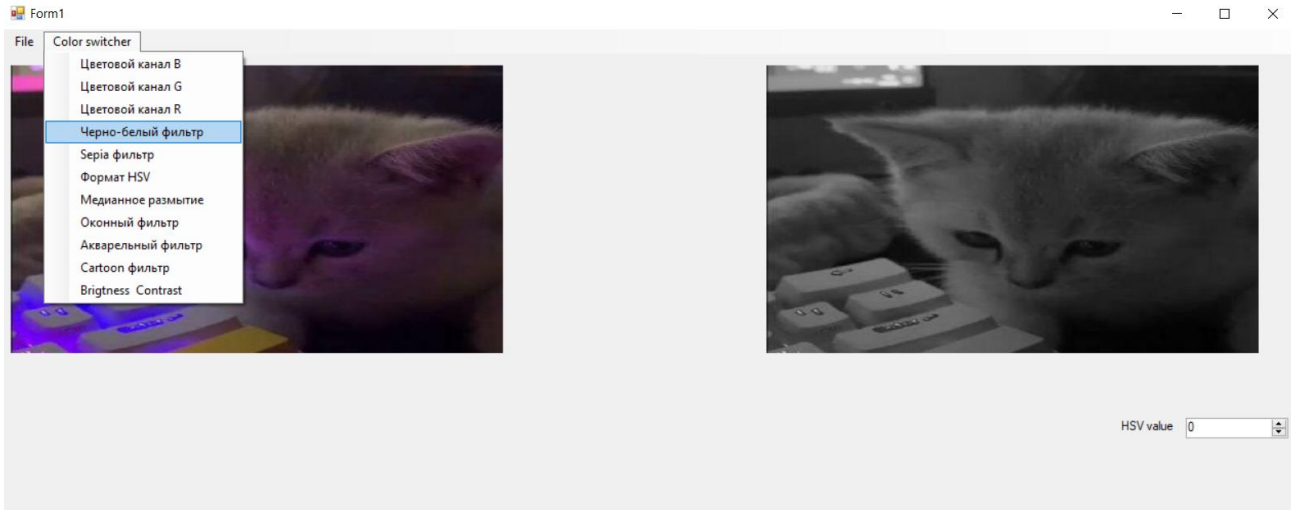
Листинг программы с комментариями:

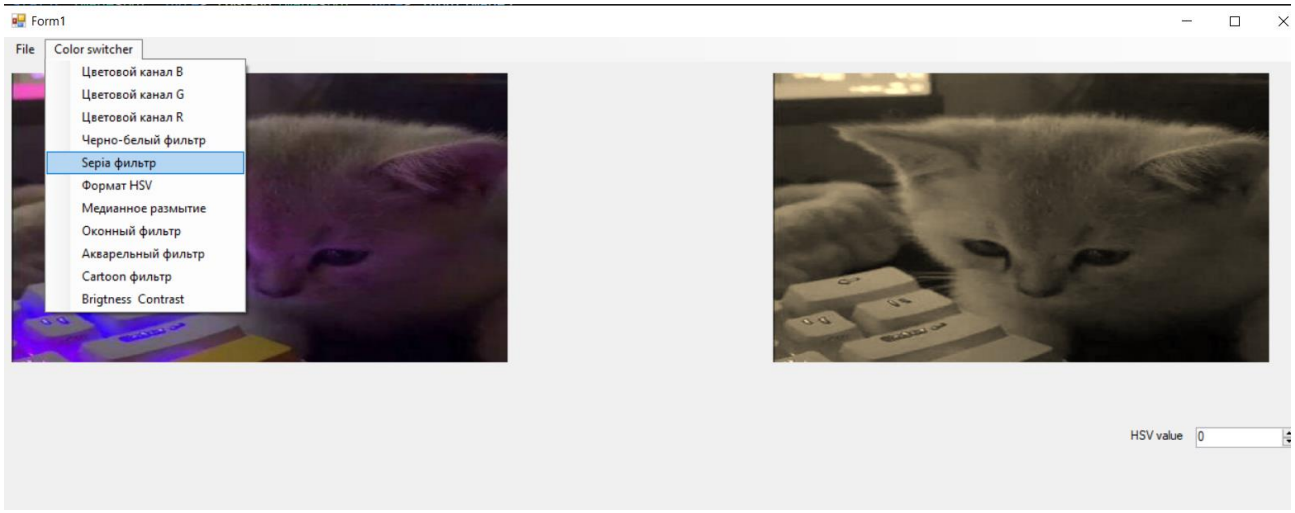


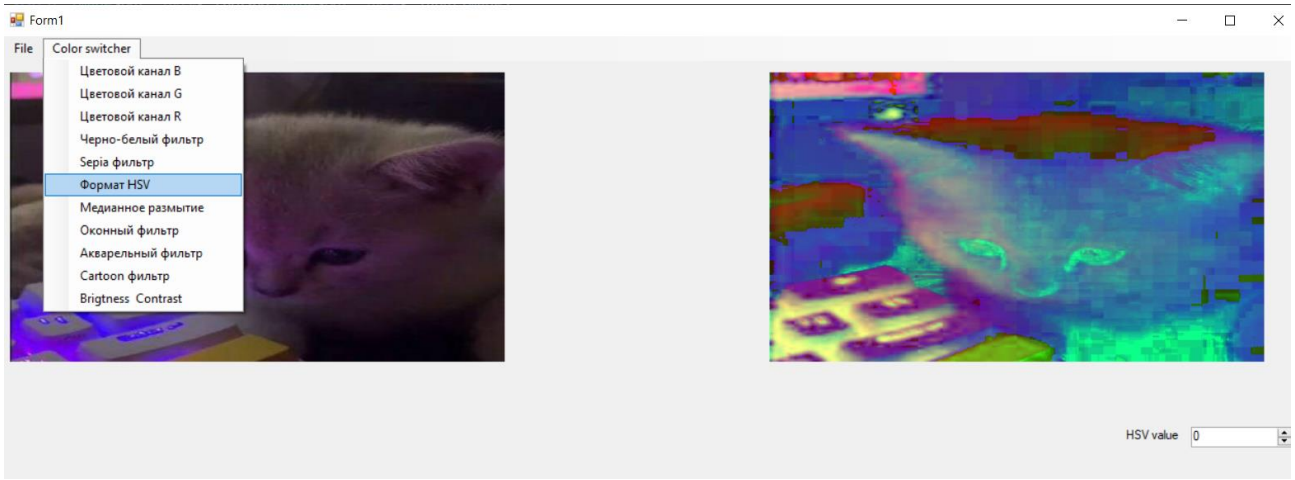


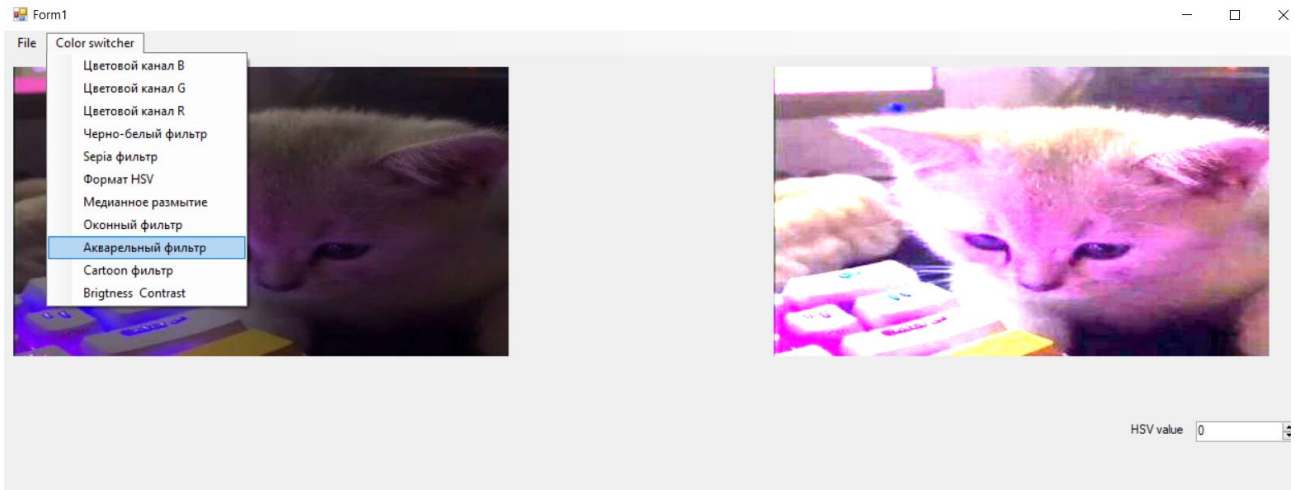

Form1
File
Color switcher
HSV value
0

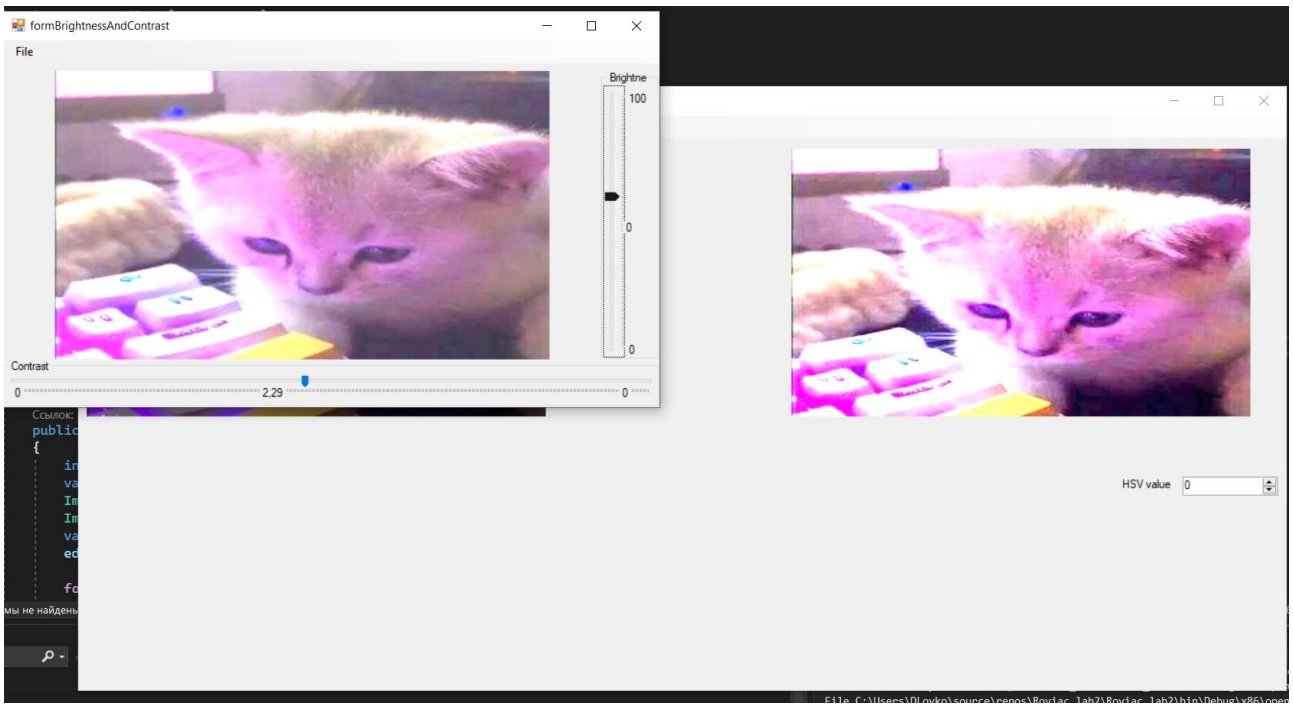

Form1
File
Color switcher
HSV value
0


Form1
File
Color switcher
Цветовой канал B
Цветовой канал G
Цветовой канал R
Черно-белый фильтр
Sepia фильтр
Формат HSV
Медианное размытие
Оконный фильтр
Акварельный фильтр
Cartoon фильтр
Brigtness Contrast
HSV value
0


Form1
File
Color switcher
Цветовой канал B
Цветовой канал G
Цветовой канал R
Черно-белый фильтр
Sepia фильтр
Формат HSV
Медианное размытие
Оконный фильтр
Акварельный фильтр
Cartoon фильтр
Brigtness Contrast
HSV value
0


Form1
File
Color switcher
Цветовой канал B
Цветовой канал G
Цветовой канал R
Черно-белый фильтр
Sepia фильтр
Формат HSV
Медианное размытие
Оконный фильтр
Акварельный фильтр
Cartoon фильтр
Brigtness Contrast
HSV value
0


Form1
File
Color switcher
Цветовой канал B
Цветовой канал G
Цветовой канал R
Черно-белый фильтр
Sepia фильтр
Формат HSV
Медианное размытие
Оконный фильтр
Акварельный фильтр
Cartoon фильтр
Brigtness Contrast
HSV value
0



Листинг кода:

```
using Emgu.CV.Structure;
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;
using Emgu.CV;
using Emgu.CV.CvEnum;
using System.Data.SqlTypes;
using static System.Net.Mime.MediaTypeNames;
using System.Drawing.Imaging;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using Emgu.CV.Cuda;
```

//1.Вывод значений одного из трёх цветовых каналов по выбору пользователя. done
//2. Вывод чёрно-белой версии изображения. done
//3. Вывод Sepia версии изображения. done
//4. Вывод изображения с возможностью изменения его яркости и контраста. https://www.youtube.com/watch?v=uLjSLEBAP0k&ab_channel=AKHTARJAMIL done
//5. Вывод результатов логических операций «дополнение», «исключение» и «пересечение», с возможностью выбора изображения для соответствующей операции.
//6. Вывод изображения преобразованного в формат HSV, с возможностью изменения значений HSV. // done
//7. Вывод изображений с применением к ним медианного размытия. // done
//8. Вывод изображений с применением к ним оконного фильтра, с возможностью изменения матрицы фильтра из формы приложения. done
//9. Вывод изображений с применением к ним «акварельного фильтра», а так же, возможностью выбора яркости, контраста и параметров смешивания изображений.
//10. Вывод изображений с применением к ним «cartoon filter» и возможностью изменения порога преобразования изображения

```
namespace Roviac_lab2
{
    public partial class Form1 : Form
    {
        private Image<Bgr, byte> sourceImage; // Глобальная переменная картинки
        public Form1()
        {
            InitializeComponent();
        }
```

```
private void imageBox1_Click(object sender, EventArgs e)
{
}

private void imageBox2_Click(object sender, EventArgs e)
{
}

private void BlackAndWhite_Click(object sender, EventArgs e)
{
    var grayImage = sourceImage.Convert<Gray, byte>();
    imageBox2.Image = grayImage;
}

private void Form1_Load(object sender, EventArgs e)
{

}

private void openToolStripMenuItem_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    var result = openFileDialog.ShowDialog();
    openFileDialog.Filter = "Picture files (*jpg, *.png)| *.jpg;*.png";
    if (result == DialogResult.OK)
    {
        string fileName = openFileDialog.FileName;
        sourceImage = new Image<Bgr, byte>(fileName);
        sourceImage = sourceImage.Resize(480, 280, Inter.Linear);
        imageBox1.Image = sourceImage;
```

```
            imageBox2.Image = sourceImage;
        }
    }

    private void exitToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void цветовойКаналBToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        var channel = sourceImage.Split()[0];
        imageBox2.Image = channel;
    }

    private void цветовойКаналGToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        var channel = sourceImage.Split()[1];
        imageBox2.Image = channel;
    }

    private void цветовойКаналRToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        var channel = sourceImage.Split()[2];
        imageBox2.Image = channel;
    }
```

```
        private void чернобелыйФильтрToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            var grayImage = new Image<Gray, byte>(sourceImage.Size);
            for (int y = 0; y < grayImage.Height; y++)
                for (int x = 0; x < grayImage.Width; x++)
                {
                    byte color = grayImage.Data[y, x, 0];
                    grayImage.Data[y, x, 0] = Convert.ToByte(0.299 * sourceImage.Data[y,
x, 2] + 0.587 * sourceImage.Data[y, x, 1] + 0.114 * sourceImage.Data[y, x, 0]);
                };
            imageBox2.Image = grayImage;
        }

        private void форматHSVToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (sourceImage == null)
            {
                MessageBox.Show("Выберите картинку.");
                return;
            }
            Image<Hsv, byte> imageOutput = new Image<Hsv,
byte>(sourceImage.Width, sourceImage.Height);
            Image<Bgr, byte> imageFinalOutput = new Image<Bgr,
byte>(sourceImage.Width, sourceImage.Height);
            CvInvoke.CvtColor(sourceImage, imageOutput, ColorConversion.Bgr2Hsv +
(int)numericUpDown1.Value);
            imageFinalOutput.Data = imageOutput.Data;
            imageBox2.Image = imageFinalOutput;
```

```
        }

        private void sepiaФильтрToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            var sepiaImage = new Image<Bgr, byte>(sourceImage.Size);
            for (int x = 0; x < sourceImage.Width; x++)
                for (int y = 0; y < sourceImage.Height; y++)
                {
                    byte colorBlue = sourceImage.Data[y, x, 0];
                    byte colorGreen = sourceImage.Data[y, x, 1];
                    byte colorRed = sourceImage.Data[y, x, 2];

                    var sepiaBlue = colorRed * 0.272 + colorGreen * 0.534 + colorBlue *
0.131; // blue
                    var sepiaGreen = colorRed * 0.349 + colorGreen * 0.686 + colorBlue *
0.168; // green
                    var sepiaRed = colorRed * 0.393 + colorGreen * 0.769 + colorBlue *
0.189; // red

                    if (sepiaBlue > 255)
                    {
                        sepiaBlue = 255;
                    }
                    if (sepiaGreen > 255)
                    {
                        sepiaGreen = 255;
                    }
                    if (sepiaRed > 255)
                    {
```

```
                sepiaRed = 255;
            }
            sepiaImage.Data[y,x,0] = Convert.ToByte(sepiaBlue);
            sepiaImage.Data[y, x, 1] = Convert.ToByte(sepiaGreen);
            sepiaImage.Data[y, x, 2] = Convert.ToByte(sepiaRed);
        }
    imageBox2.Image = sepiaImage;
}

    private void brigtnessContrastToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        try
        {
            if (sourceImage == null)
            {
                throw new Exception("Select Image");
            }

            formBrightnessAndContrast form = new formBrightnessAndContrast();
            form.sourceImage = sourceImage.Clone();
            form.ShowDialog();
        }
        catch (Exception)
        {

            throw;
        }
    }
```

```
private void медианноеРазмытиеToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        var blurImage = new Image<Bgr, byte>(sourceImage.Size);
        MessageBox.Show(blurImage.Size.ToString());
        for (int x = 0;x < sourceImage.Width-3;x++)
            for (int y = 0;y < sourceImage.Height-3;y++)
            {
                for(int z = 0;z < 3; z++)
                {
                    List<int> window = new List<int>();
                    //byte color = blurImage.Data[y, x, 0];
                    for (int j = x; j <= x + 2; j++)
                    {
                        for (int k = y; k <= y + 2; k++)
                        {
                            window.Add(sourceImage.Data[k, j, z]);
                        }
                    }
                    window.Sort();
                    if (window.Count < 9)
                    {
                        MessageBox.Show(window.Count().ToString());
                    }
                    byte color = Convert.ToByte(window[window.Count / 2]);
                    blurImage.Data[y, x, z] = color;
                }
            }
        imageBox2.Image = blurImage;
    }
```

```
        private void оконныйФильтрToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            try
            {
                if (sourceImage == null)
                {
                    throw new Exception("Select Image");
                }

                Form2 form = new Form2();
                form.sourceImage = sourceImage.Clone();
                form.ShowDialog();
            }
            catch (Exception)
            {

                throw;
            }
        }
        public static double addColor(double color1, double color2, double color3)
        {

            if (color1 + color2 + color3 > 255)
                return 255;
            else if (color1 + color2 + color3 < 0)
                return 0;
            else return color1 + color2 + color3;
        }
```

```
        public static Image<Bgr, byte> changeBrightness(Image<Bgr, byte>
inputImage, double brightnessDiff)
        {
            var resultImage = new Image<Bgr, byte>(inputImage.Size);
            for (int channel = 0; channel < resultImage.NumberOfChannels; channel++)
                for (int x = 0; x < resultImage.Width; x++)
                    for (int y = 0; y < resultImage.Height; y++)
                    {
                        byte red = inputImage.Data[y, x, 2];
                        byte green = inputImage.Data[y, x, 1];
                        byte blue = inputImage.Data[y, x, 0];
                        red = Convert.ToByte(addColor(red, brightnessDiff, 0));
                        green = Convert.ToByte(addColor(green, brightnessDiff, 0));
                        blue = Convert.ToByte(addColor(blue, brightnessDiff, 0));
                        if (channel == 2)
                            resultImage.Data[y, x, channel] = red;
                        if (channel == 1)
                            resultImage.Data[y, x, channel] = green;
                        if (channel == 0)
                            resultImage.Data[y, x, channel] = blue;
                    }

            return resultImage;
        }

        public static double multiplyColor(double color1, double color2)
        {
            if (color1 * color2 > 255)
                return 255;
            else if (color1 * color2 < 0)
```

```
            return 0;
        else return color1 * color2;
    }
    public static Image<Bgr, byte> changeContrast(Image<Bgr, byte> inputImage,
double contrastDiff)
    {
        contrastDiff /= 100;
        var resultImage = new Image<Bgr, byte>(inputImage.Size);
        for (int channel = 0; channel < resultImage.NumberOfChannels; channel++)
            for (int x = 0; x < resultImage.Width; x++)
                for (int y = 0; y < resultImage.Height; y++)
                {
                    byte red = inputImage.Data[y, x, 2];
                    byte green = inputImage.Data[y, x, 1];
                    byte blue = inputImage.Data[y, x, 0];
                    red = Convert.ToByte(multiplyColor(red, contrastDiff));
                    green = Convert.ToByte(multiplyColor(green, contrastDiff));
                    blue = Convert.ToByte(multiplyColor(blue, contrastDiff));
                    if (channel == 2)
                        resultImage.Data[y, x, channel] = red;
                    if (channel == 1)
                        resultImage.Data[y, x, channel] = green;
                    if (channel == 0)
                        resultImage.Data[y, x, channel] = blue;
                }

        return resultImage;
    }
```

```
public static Image<Bgr, byte> sumImages(Image<Bgr, byte> inputImage1, Image<Bgr, byte> inputImage2, double cf1, double cf2)
{
    var resultImage = new Image<Bgr, byte>(inputImage1.Size);
    for (int channel = 0; channel < resultImage.NumberOfChannels; channel++)
        for (int x = 0; x < resultImage.Width; x++)
            for (int y = 0; y < resultImage.Height; y++)
            {
                byte red = Convert.ToByte(addColor(cf1 * inputImage1.Data[y, x, 2], cf2 * inputImage2.Data[y, x, 2], 0));
                byte green = Convert.ToByte(addColor(cf1 * inputImage1.Data[y, x, 1], cf2 * inputImage2.Data[y, x, 1], 0));
                byte blue = Convert.ToByte(addColor(cf1 * inputImage1.Data[y, x, 0], cf2 * inputImage2.Data[y, x, 0], 0));
                if (channel == 2)
                    resultImage.Data[y, x, channel] = red;
                if (channel == 1)
                    resultImage.Data[y, x, channel] = green;
                if (channel == 0)
                    resultImage.Data[y, x, channel] = blue;
            }
    return resultImage;
}

public static Image<Bgr, byte> medianBlur(Image<Bgr, byte> inputImage, int mCoef)
{
    var blurImage = new Image<Bgr, byte>(inputImage.Size);
    for (int x = 0; x < inputImage.Width - mCoef; x++)
        for (int y = 0; y < inputImage.Height - mCoef; y++)
```

```
            {
                for (int z = 0; z < 3; z++)
                {
                    List<int> window = new List<int>();
                    for (int j = x; j <= x + mCoef - 1; j++)
                    {
                        for (int k = y; k <= y + mCoef - 1; k++)
                        {
                            window.Add(inputImage.Data[k, j, z]);
                        }
                    }
                    window.Sort();
                    if (window.Count < Math.Pow(mCoef, 2))
                    {
                        MessageBox.Show(window.Count().ToString());
                    }
                    byte color = Convert.ToByte(window[window.Count / 2]);
                    blurImage.Data[y, x, z] = color;
                }
            }
        return blurImage;
    }
    private void акварельныйФильтрToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        Image<Bgr, byte> brightImg = changeBrightness(sourceImage, 2);
        Image<Bgr, byte> contrastImg = changeContrast(brightImg,2);
        Image<Bgr, byte> blurImg = medianBlur(contrastImg, 2);
        Image<Bgr, byte> sumImg = sumImages(blurImg,
sourceImage.Resize(blurImg.Width, blurImg.Height, Inter.Linear), 2, 4);
```

```
            imageBox2.Image = sumImg;
        }


        private void cartoonФильтрToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            cartoonFilter(sourceImage);
        }

        public static Image<Bgr, byte> toGray(Image<Bgr, byte> inputImage)
        {
            var grayImage = new Image<Bgr, byte>(inputImage.Size);

            for (int x = 0; x < grayImage.Width; x++)
                for (int y = 0; y < grayImage.Height; y++)
                {
                    grayImage.Data[y, x, 0] = Convert.ToByte(0.299 * inputImage.Data[y,
x, 2] + 0.587 *
                        inputImage.Data[y, x, 1] + 0.114 * inputImage.Data[y, x, 0]);
                }

            return grayImage;
        }

        public static double intersectionColors(double color1, double color2)
        {
            return Math.Min(color1, color2);
        }
        public static Image<Bgr, byte> cartoonFilter(Image<Bgr, byte> inputImage)
```

```
        {
            int mCoef = 3;
            var resultImage = new Image<Bgr, byte>(inputImage.Size);
            Image<Bgr, byte> grayImg = toGray(inputImage);
            Image<Bgr, byte> blurImg = medianBlur(grayImg, mCoef);
            var edges = grayImg.Convert<Gray, byte>();
            edges = edges.ThresholdAdaptive(new Gray(100),
AdaptiveThresholdType.MeanC, ThresholdType.Binary, 3, new Gray(0.03));

            for (int channel = 0; channel < resultImage.NumberOfChannels; channel++)
                for (int x = 0; x < resultImage.Width; x++)
                    for (int y = 0; y < resultImage.Height; y++)
                    {
                        byte red = inputImage.Data[y, x, 2];
                        byte green = inputImage.Data[y, x, 1];
                        byte blue = inputImage.Data[y, x, 0];
                        red = Convert.ToByte(intersectionColors(red, edges.Data[y, x, 0]));
                        green = Convert.ToByte(intersectionColors(green, edges.Data[y, x,
0]));
                        blue = Convert.ToByte(intersectionColors(blue, edges.Data[y, x, 0]));
                        if (channel == 2)
                            resultImage.Data[y, x, channel] = red;
                        if (channel == 1)
                            resultImage.Data[y, x, channel] = green;
                        if (channel == 0)
                            resultImage.Data[y, x, channel] = blue;
                    }
            return resultImage;
        }
    }
```

}