Реферат

32стр., 11 рис.

ГРАФИЧЕСКИЙ РЕДАКТОР, C#, MS VISUAL STUDIO 2017, WINDOWS FORMS, MDI

Объектом исследования являются языковые средства C# для создания графических интерфейсов пользователя Windows Forms в интегрированной среде разработки Microsoft Visual Studio.

Целью работы является создание графического редактора на языке C# с использованием Windows Forms MDI , позволяющего открывать изображения из графических файлов, изменять яркость , создавать размытие изображения (Box Blur), сохранять измененные изображения в графические файлы.

В процессе работы проводилось изучение среды разработки Microsoft Visual Srudio и средств языка программирования C#.

В результате исследования была созданана программа позволяющая открывать изображения из графических файлов, изменять яркость изображений, создавать размытие изображения (Box Blur), сохранять измененные изображения в графические файлы.

Эффективность программы определяет скорость и ресурсоемкость обработки изображения.

Основные конструктивные и технико-эксплуатационные показатели: удобный интерфейс, скорость обработки изображения, поддержка 4 форматов изображения.

Программа может применятся для редактирования изображений, фотографий.

**Содержание**

Введение3

Список терминов4

Главные преимущества Microsoft Visual Studio5

Подготовка к разработке ПО6

Описание используемых алгоритмов7

Инструкция пользователю по работе с программой 15

Листинг кода программы21

Вывод31

Список использованных источников и литературы32

**Введение**

В настоящее время рынок программного обеспечения переполнен различными программами и редакторами, позволяющими обрабатывать и редактировать цифровые фото. Человеку, не слишком хорошо понимающему особенности тех или иных программных средств, порой очень сложно разобраться в этом многообразии софта. Однако, правильный выбор программных средств для решения конкретной задачи по обработке фотоснимков является одним из залогов успеха получения законченных фотографий.

**Список терминов**

1. Изображение – способ представления графической информации в виде цифровых данных.
2. Метод – это блок кода, содержащий ряд инструкций. Программа инициирует выполнение инструкций, вызывая метод и указывая все аргументы, необходимые для этого метода.
3. Многодокументный интерфейс (MDI) – способ организации графического интерфейса пользователя, предполагающий использование оконного интерфейса, в котором большинство окон (исключая, как правило, только модальные окна) расположены внутри одного общего окна.
4. Программный интерфейс приложения (API) – описания способов, с помощью которых одно программное приложение может взаимодействовать с другим.
5. Bitmap – Специальный формат изображений, разработанный Microsoft. В данном формате можно хранить только однослойные растры. На каждый пиксель в разных файлах может приходиться разное количество бит (глубина цвета). Microsoft предлагает битности 1, 2, 4, 8, 16, 24, 32, 48 и 64. В битностях 8 и ниже цвет указывается индексом из таблицы цветов (палитры), а при бо́льших – непосредственным значением. Цвет же в любом случае можно задать только в цветовой модели RGB, но в битностях 16 и 32 можно получить оттенки серого с глубиной до 16 и 32 бит, соответственно.

**Главные преимущества Microsoft Visual Studio**

1. Большое количество функций;
2. Наличие отладчика для JS;
3. Поддержка нескольких мониторов;
4. Использование сплит-окон XML и CSS;
5. Удобная регулировка рабочей области;
6. Продвинутый конструктор и редактор кода;
7. Простой и интуитивно понятный интерфейс на русском языке;
8. Возможность разрабатывать графические элементы Windows Forms;
9. Регулярные автообновления от официального сайта Майкрософт.

Исходя из вышеперечисленного можно сделать вывод, что Visual Studio – среда, дающая широкие возможности в разработке программного обеспечения, что делает её идеальным выбором для реализации проекта.

**Подготовка к разработке ПО**

В ходе реализации проекта необходимо разработать приложение, которое способно создавать новые изображения из фрагментов существующего и сохранять эти изображения.

Программа должна осуществлять:

* Открытие изображений;
* Изменять яркость изображений;
* Создавать размытие изображения;
* Сохранять измененные изображения в графические файлы;

**Описание используемых алгоритмов**

**1)Метод яркостьконтрастностьToolStripMenuItem\_Click вызывает форму со шкалой яркости изображения.**

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

{

BrightnessForm BrightnessForm = new BrightnessForm(this);

BrightnessForm.ShowDialog();

}

**2)Класс Brightness отвечает за обработку каждого отдельного пикселя.**

using System;

namespace Program

{

class Brightness

{

public static UInt32 brightness (UInt32 dot, int lenght,int pozition)

{

int R;

int G;

int B;

**Вычисление коэффициента яркости в из значения текущей позиции слайдера и его максимального значения.**

int N = (100 / lenght) \* pozition;

**Для изменения яркости на N процентов используется следующая формула: I = I + N • 128 / 100, где I – соответственно R, G, B каналы каждой точки изображения.**

R = (int)(((dot & 0x00FF0000) >> 16) + N \* 128 / 100);

G = (int)(((dot & 0x0000FF00) >> 8) + N \* 128 / 100);

B = (int)((dot & 0x000000FF) + N \* 128 / 100);

**контролируем переполнение переменных, потому что R,G,B каналы имеют диапазон от 0 до 255.**

if (R > 255) R = 255;

if (R < 0) R = 0;

if (G > 255) G = 255;

if (G < 0) G = 0;

if (B > 255) B = 255;

if (B < 0) B = 0;

dot = 0xFF000000| ((UInt32)B) | ((UInt32)R << 16) | ((UInt32)G << 8) ;

return dot;

}

}

}

**3)Форма BrightnessForm работает с целым изображением.**

1. **изменение яркости**

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

{

if (MainForm.imageName!="\0")

{

UInt32 pix;

for (int i = 0; i < MainForm.image.Height; i++)

for (int j = 0; j < MainForm.image.Width; j++)

{

pix = Brightness.brightness(MainForm.pixels[i,j], trackBar1.Maximum, trackBar1.Value);

MainForm.FromOnePixelToBitmap(i, j, pix);

}

FromBitmapToScreen();

}

}

1. **сохранение изменений яркости**

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

{

if (MainForm.imageName!="\0")

{

for (int i = 0; i < MainForm.image.Height; i++)

for (int j = 0; j < MainForm.image.Width; j++)

MainForm.pixels[i, j] = (UInt32)(MainForm.image.GetPixel(j, i).ToArgb());

trackBar1.Value = 0;

}

}

1. **вывод изображения на экран**

void FromBitmapToScreen()

{

OwnersForm.FromBitmapToScreen();

}

**4)Для размытия и повышения резкости изображения класс Filter и структура RGB**

using System;

namespace Program

{

struct RGB

{

public float R;

public float G;

public float B;

}

class Filter

{

public static UInt32[,] matrixFiltr(int height, int width, UInt32[,] pixels,double[,] matryx, int n )

{

int a, b, k, c, space = (int)(n / 2);

int tmpH = height + 2 \* space;

int tmpW = width + 2 \* space;

Изображение искусственно увеличивается для обработки пикселей, у которых нет некоторых соседних пикселей, например для тех, что находятся в углу изображения

UInt32[,] tpixel = new UInt32[tmpH, tmpW];

UInt32[,] cpixel = new UInt32[height, width];

#### Обработка пикселей

1. крайние левая и правая

for (a = space; a < tmpH - space; a++)

for (b = 0; b < space; b++)

{

tpixel[a, b] = pixels[a - space, b];

tpixel[a, tmpW - 1 - b] = pixels[a - space, width - 1 - b];

}

1. крайние верхняя и нижняя

for (a = 0; a < space; a++)

for (b = space; b < tmpW - space; b++)

{

tpixel[a, b] = pixels[a, b - space];

tpixel[tmpH - 1 - a, b] = pixels[height - 1 - a, b - space];

}

1. углы

for (a = 0; a < space; a++)

for (b = 0; b < space; b++)

{

tpixel[a, tmpW - 1 - b] = pixels[0, width - 1];

tpixel[tmpH - 1 - a, b] = pixels[height - 1, 0];

tpixel[a, b] = pixels[0, 0];

tpixel[tmpH - 1 - a, tmpW - 1 - b] = pixels[height - 1, width - 1];

}

1. центр

for (a = 0; a < height; a++)

for (b = 0; b < width; b++)

tpixel[a + space, b + space] = pixels[a, b];

применение ядра свертки для вычисления новых значений пикселей и последущего размытия изображения.

RGB ColorOfPixel = new RGB();

RGB ColorOfCell= new RGB();

for (a = space; a < tmpH - space; a++)

for (b = space; b < tmpW - space; b++)

{

ColorOfPixel.R = 0;

ColorOfPixel.G = 0;

ColorOfPixel.B = 0;

for (k = 0; k < n; k++)

for (c = 0; c < n; c++)

{

ColorOfCell = calculationOfColor(tpixel[a - space + k, b - space + c], matryx[k, c]);

ColorOfPixel.R += ColorOfCell.R;

ColorOfPixel.G += ColorOfCell.G;

ColorOfPixel.B += ColorOfCell.B;

}

**контролируем переполнение переменных, потому что R,G,B каналы имеют диапазон от 0 до 255.**

if (ColorOfPixel.R < 0) ColorOfPixel.R = 0;

if (ColorOfPixel.R > 255) ColorOfPixel.R = 255;

if (ColorOfPixel.G < 0) ColorOfPixel.G = 0;

if (ColorOfPixel.G > 255) ColorOfPixel.G = 255;

if (ColorOfPixel.B < 0) ColorOfPixel.B = 0;

if (ColorOfPixel.B > 255) ColorOfPixel.B = 255;

cpixel[a - space, b - space] = buildChanals(ColorOfPixel);

}

return cpixel;

}

**сборка каналов для возвращения пикселя в UInt32 формате.**

public static UInt32 buildChanals(RGB ColorOfPixel)

{

UInt32 Color;

Color = 0xFF000000 | ((UInt32)ColorOfPixel.B) | ((UInt32)ColorOfPixel.R << 16) | ((UInt32)ColorOfPixel.G << 8) ;

return Color;

}

**вычисление нового цвета**

public static RGB calculationOfColor(UInt32 pixels, double rate)

{

RGB сolor = new RGB();

сolor.R = (float)(rate \* ((pixels & 0x00FF0000) >> 16));

сolor.G = (float)(rate \* ((pixels & 0x0000FF00) >> 8));

сolor.B = (float)(rate \* (pixels & 0x000000FF));

return сolor;

}

**Матрицы sharpness и blur хранят коэффициенты, на которые будет умножатся пиксели исходной матрицы.**

**размытие**

public const int N2 = 3;

public static double[,] blur = new double[N1, N1] {{0.111, 0.111, 0.111},

{0.111, 0.111, 0.111},

{0.111, 0.111, 0.111}};

**повышение резкости**

public const int N1 = 3;

public static double[,] sharpness = new double[N1, N1] {{-1, -1, -1},

{-1, 9, -1},

{-1, -1, -1}};

}

}

**5)Метод открытьToolStripMenuItem\_Click используется для открытия изображений.**

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

{

OpenFileDialog open\_dialog = new OpenFileDialog();

open\_dialog.Filter = "Image Files(\*.BMP;\*.JPG;\*.GIF;\*.PNG)|\*.BMP;\*.JPG;\*.GIF;\*.PNG|All files (\*.\*)|\*.\*";

if (open\_dialog.ShowDialog() == DialogResult.OK)

{

try

{

imageName = open\_dialog.FileName;

image = new Bitmap(open\_dialog.FileName);

this.Height = image.Height + 80;

this.Width = image.Width + 45;

this.pictureBox1.Size = image.Size;

pictureBox1.Image = image;

pictureBox1.Invalidate();

//

pixels = new UInt32[image.Height, image.Width];

for (int x = 0; x < image.Width; x++)

for (int y = 0; y < image.Height; y++)

pixels[y,x] = (UInt32)(image.GetPixel(x, y).ToArgb());

}

catch

{

DialogResult rezult = MessageBox.Show("Открытие выбранного файла невозможно","Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

imageName = "\0";

}

}

}

**6)Метод сохранитьКакToolStripMenuItem\_Click используется для сохранения изображений**

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

{

if (pictureBox1.Image != null)

{

SaveFileDialog savedialog = new SaveFileDialog();

savedialog.CheckPathExists = true;

savedialog.OverwritePrompt = true;

savedialog.Title = "Сохранить картинку как";

savedialog.ShowHelp = true;

savedialog.Filter = "Image Files(\*.BMP)|\*.BMP|Image Files(\*.JPG)|\*.JPG|Image Files(\*.GIF)|\*.GIF|Image Files(\*.PNG)|\*.PNG|All files (\*.\*)|\*.\*";

if (savedialog.ShowDialog() == DialogResult.OK)

{

try

{

image.Save(savedialog.FileName, System.Drawing.Imaging.ImageFormat.Jpeg);

}

catch

{

MessageBox.Show("Сохранить изображение невозможно", "Ошибка",MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**Инструкция пользователю по работе с программой**

На главной форме имеется меню с пятью пунктами:

1. При нажатии на пункт «Файл» можно открыть новое изображение, сохранить текущее;

2. При нажатии на пункт «О программе» можно открыть окно с информацией о приложении и создателе приложения, так же можно выйти и приложения;

3. При нажатии на пункт «Фильтры» можно размыть, увеличить резкость или изменить яркость выбранного изображения;

4. При выборе фильтра «Яркость» откроется форма с удобной шкалой изменения яркости изображения.

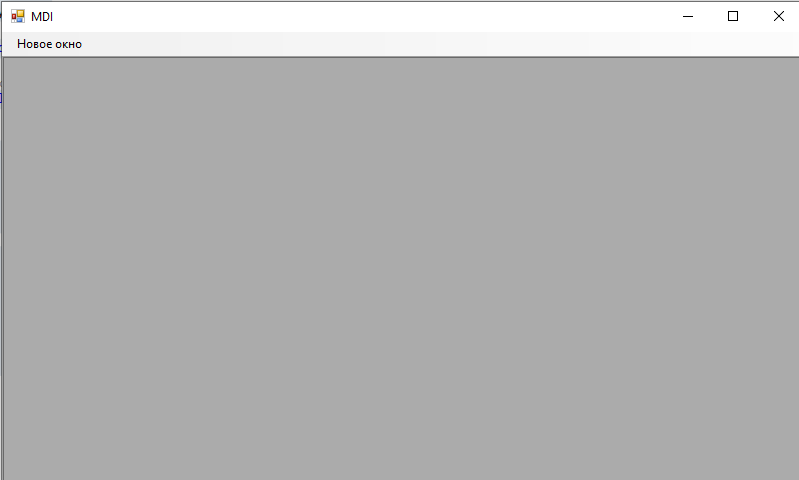


Рисунок 1 – Главная форма приложения

После нажатия на пункт «Новое окно» открывается окно для работы с изображением.

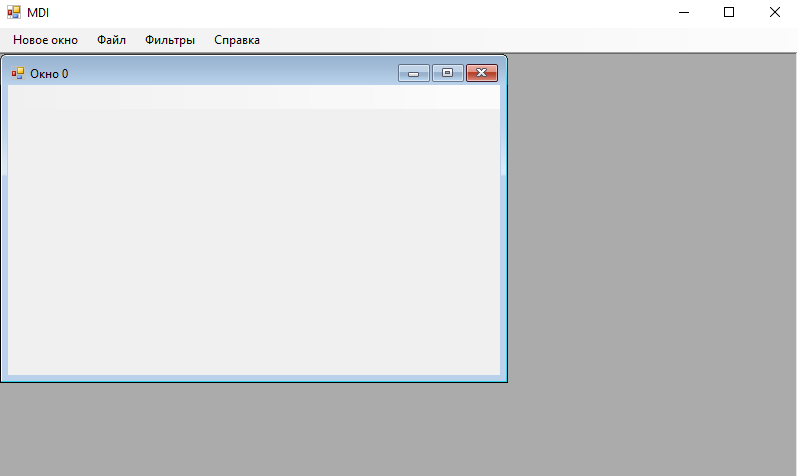


Рисунок 2 – Окно для работы с изображением

После нажатия на пункт «Открыть» открывается окно выбора файла на компьютере пользователя.

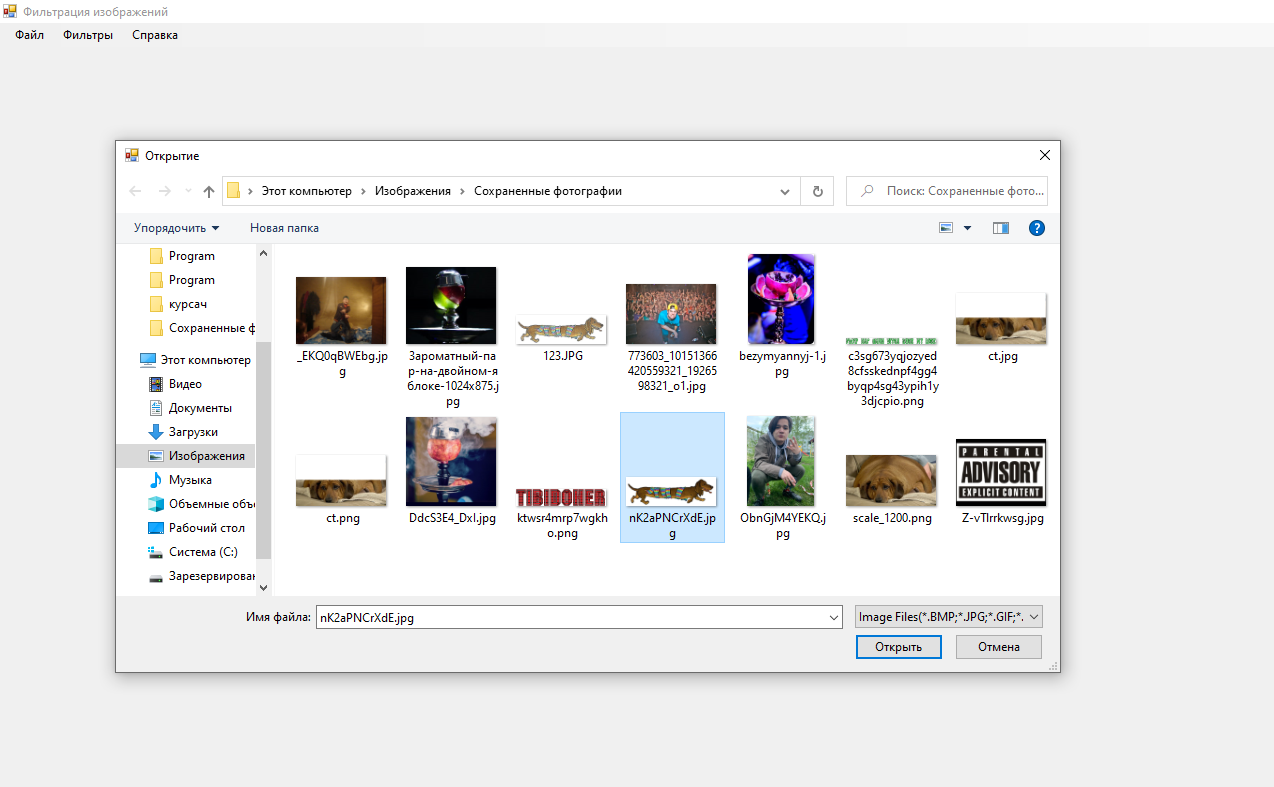


Рисунок 3 – Окно открытия файла



Рисунок 4 – Изображение открыто

После нажатия на пункт «Сохранить как...» открывается окно выбора места сохранения файла на компьютере пользователя.

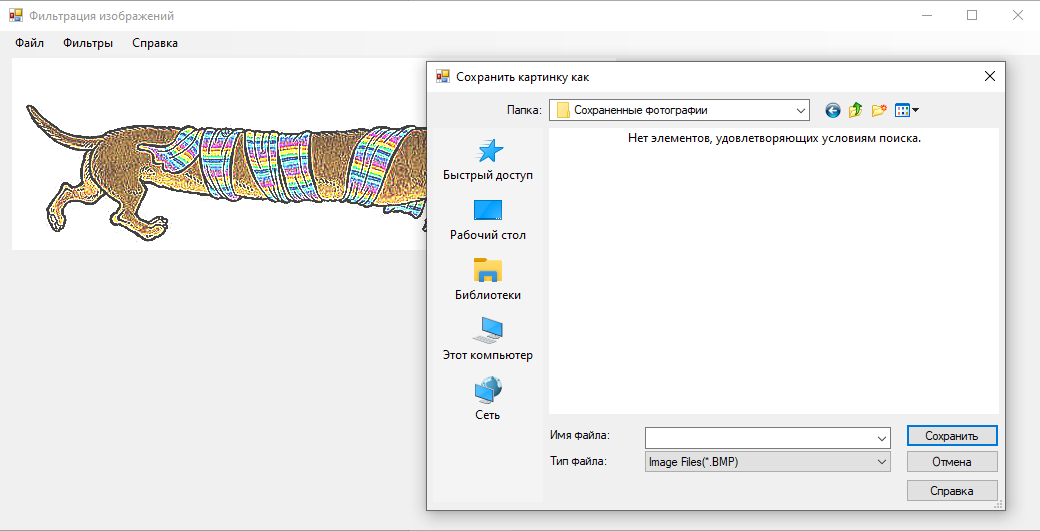


Рисунок 5 – Сохранение изображения

При нажатии пункта «Повысить резкость».



Рисунок 6 – Повышение резкости

При нажатии пункта «Размыть».



Рисунок 7 – Размытие

При выборе пункта «Яркость».

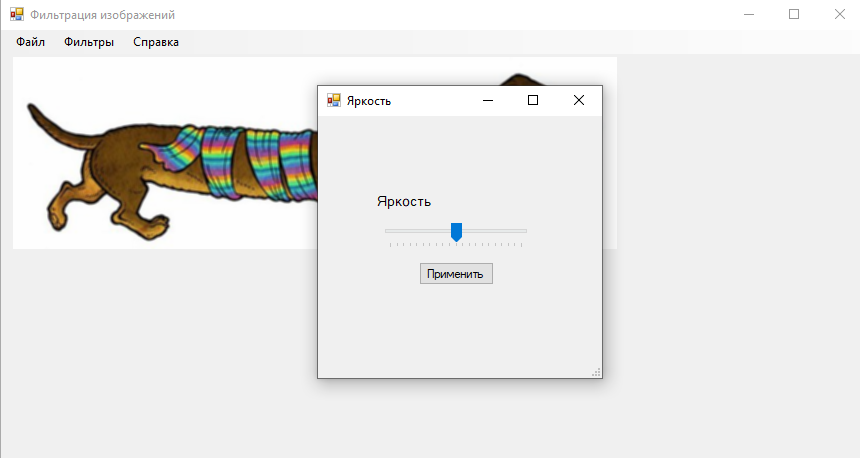


Рисунок 8 – Открытие формы для изменения яркости изображения

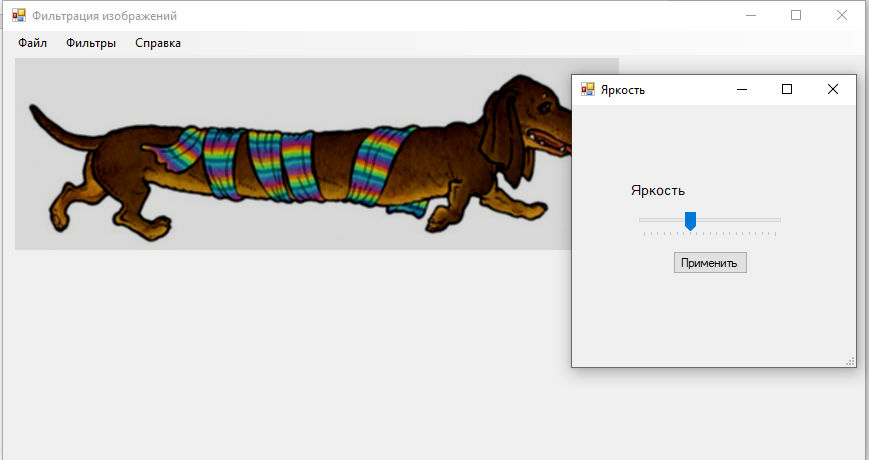


Рисунок 9 – Изменение яркости путем перемещения ползунка

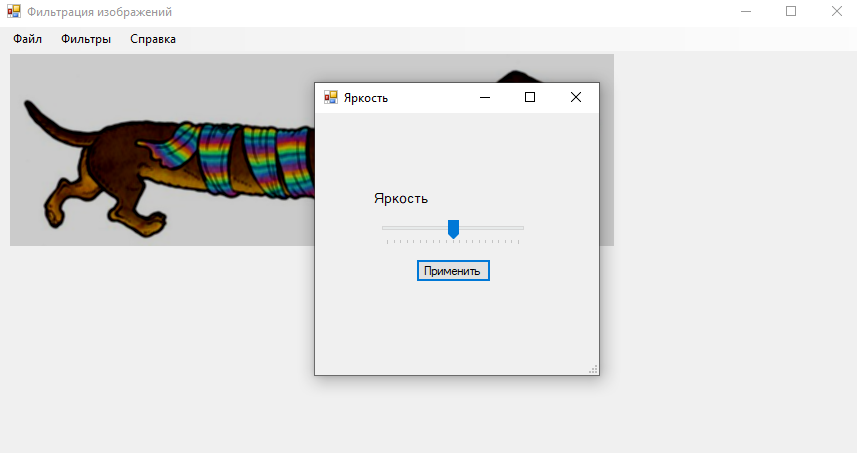
При нажатии кнопки «Применить». 

Рисунок 10 – Яркость изображения применяется

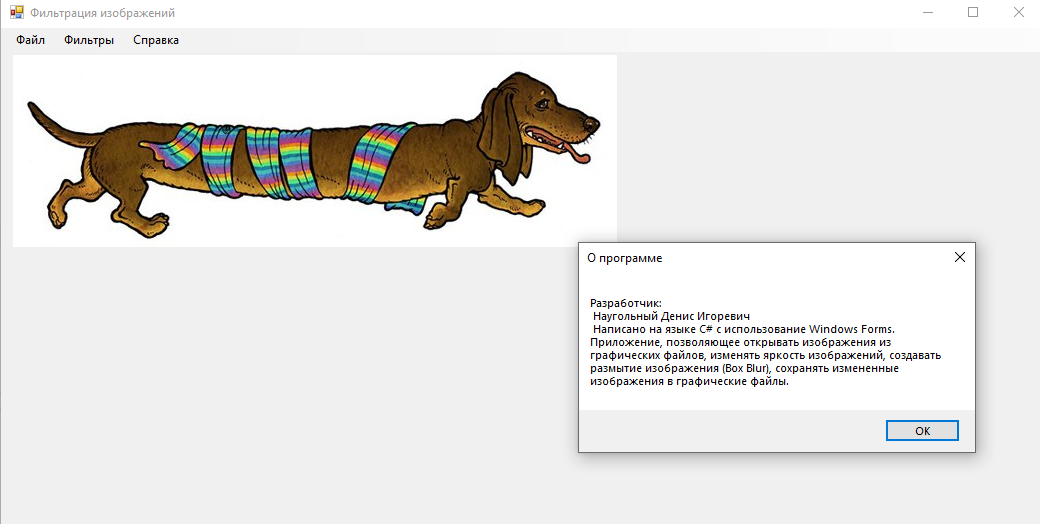
При нажатии пункта «О программе». 

Рисунок 11 – Окно с информацией о программе

**Листинг кода программы**

**MainForm.cs**

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Program

{

public partial class MainForm : Form

{

public static Bitmap image;

public static string imageName = "\0";

public static UInt32[,] pixels;

public MainForm()

{

InitializeComponent();

}

//открытие

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

{

OpenFileDialog open\_dialog = new OpenFileDialog();

open\_dialog.Filter = "Image Files(\*.BMP;\*.JPG;\*.GIF;\*.PNG)|\*.BMP;\*.JPG;\*.GIF;\*.PNG|All files (\*.\*)|\*.\*";

if (open\_dialog.ShowDialog() == DialogResult.OK)

{

try

{

imageName = open\_dialog.FileName;

image = new Bitmap(open\_dialog.FileName);

this.Height = image.Height + 80;

this.Width = image.Width + 45;

this.pictureBox1.Size = image.Size;

pictureBox1.Image = image;

pictureBox1.Invalidate();

//

pixels = new UInt32[image.Height, image.Width];

for (int x = 0; x < image.Width; x++)

for (int y = 0; y < image.Height; y++)

pixels[y,x] = (UInt32)(image.GetPixel(x, y).ToArgb());

}

catch

{

DialogResult rezult = MessageBox.Show("Открытие выбранного файла невозможно","Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

imageName = "\0";

}

}

}

//сохранение

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

{

if (pictureBox1.Image != null)

{

SaveFileDialog savedialog = new SaveFileDialog();

savedialog.CheckPathExists = true;

savedialog.OverwritePrompt = true;

savedialog.Title = "Сохранить картинку как";

savedialog.ShowHelp = true;

savedialog.Filter = "Image Files(\*.BMP)|\*.BMP|Image Files(\*.JPG)|\*.JPG|Image Files(\*.GIF)|\*.GIF|Image Files(\*.PNG)|\*.PNG|All files (\*.\*)|\*.\*";

if (savedialog.ShowDialog() == DialogResult.OK)

{

try

{

image.Save(savedialog.FileName, System.Drawing.Imaging.ImageFormat.Jpeg);

}

catch

{

MessageBox.Show("Сохранить изображение невозможно", "Ошибка",MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

//размыть

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

{

if (MainForm.imageName != "\0")

{

pixels = Filter.matrixFiltr(image.Height, image.Width, pixels, Filter.blur,Filter.N2 );

FromPixelToBitmap();

FromBitmapToScreen();

}

}

//Повышение резкости

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

{

if (MainForm.imageName != "\0")

{

pixels = Filter.matrixFiltr(image.Height,image.Width, pixels, Filter.sharpness ,Filter.N1 );

FromPixelToBitmap();

FromBitmapToScreen();

}

}

//преобразование из UINT32 to Bitmap

public static void FromPixelToBitmap()

{

for (int x = 0; x < image.Width; x++)

for (int y = 0; y < image.Height; y++)

image.SetPixel(x, y, Color.FromArgb((int)pixels[y, x]));

}

//преобразование UINT32 в Bitmap по пикселю

public static void FromOnePixelToBitmap(int x, int y, UInt32 pixels)

{

image.SetPixel(y, x, Color.FromArgb((int)pixels));

}

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

{

BrightnessForm BrightnessForm = new BrightnessForm(this);

BrightnessForm.ShowDialog();

}

//вывод на экран

public void FromBitmapToScreen()

{

pictureBox1.Image = image;

}

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

{

MessageBox.Show("Разработчик:\n Наугольный Денис Игоревич\n Написано на языке C# с использование Windows Forms. " +

"Приложение, позволяющее открывать изображения из графических файлов, изменять яркость изображений, создавать размытие изображения (Box Blur)," +

" сохранять измененные изображения в графические файлы.", "О программе");

}

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

{

Close();

}

}

}

## BrightnessForms.cs

using System;

using System.Windows.Forms;

namespace Program

{

public partial class BrightnessForm : Form

{

MainForm OwnersForm;

public BrightnessForm(MainForm ownerForm)

{

InitializeComponent();

this.OwnersForm = ownerForm;

this.button1.Click += new System.EventHandler(this.button\_Click);

}

//изменение яркости

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

{

if (MainForm.imageName!="\0")

{

UInt32 pix;

for (int i = 0; i < MainForm.image.Height; i++)

for (int j = 0; j < MainForm.image.Width; j++)

{

pix = Brightness.brightness(MainForm.pixels[i,j], trackBar1.Maximum, trackBar1.Value);

MainForm.FromOnePixelToBitmap(i, j, pix);

}

FromBitmapToScreen();

}

}

//сохранение изменений яркости или контрастности

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

{

if (MainForm.imageName != "\0")

{

for (int i = 0; i < MainForm.image.Height; i++)

for (int j = 0; j < MainForm.image.Width; j++)

MainForm.pixels[i, j] = (UInt32)(MainForm.image.GetPixel(j, i).ToArgb());

trackBar1.Value = 0;

}

}

//вывод изображения на экран

void FromBitmapToScreen()

{

OwnersForm.FromBitmapToScreen();

}

}

}

## Brightness.cs

using System;

namespace Program

{

class Brightness

{

//якрость

public static UInt32 brightness (UInt32 dot, int lenght,int pozition)

{

int R;

int G;

int B;

int N = (100 / lenght) \* pozition; //кол-во процентов

R = (int)(((dot & 0x00FF0000) >> 16) + N \* 128 / 100);

G = (int)(((dot & 0x0000FF00) >> 8) + N \* 128 / 100);

B = (int)((dot & 0x000000FF) + N \* 128 / 100);

//контролируем переполнение переменных

if (R > 255) R = 255;

if (R < 0) R = 0;

if (G > 255) G = 255;

if (G < 0) G = 0;

if (B > 255) B = 255;

if (B < 0) B = 0;

dot = 0xFF000000| ((UInt32)B) | ((UInt32)R << 16) | ((UInt32)G << 8) ;

return dot;

}

}

## }

## MatrixConvolution.cs

using System;

namespace Program

{

struct RGB

{

public float R;

public float G;

public float B;

}

class Filter

{

public static UInt32[,] matrixFiltr(int height, int width, UInt32[,] pixels,double[,] matryx, int n )

{

int a, b, k, c, space = (int)(n / 2);

int tmpH = height + 2 \* space;

int tmpW = width + 2 \* space;

UInt32[,] tpixel = new UInt32[tmpH, tmpW];

UInt32[,] cpixel = new UInt32[height, width];

//крайние левая и правая

for (a = space; a < tmpH - space; a++)

for (b = 0; b < space; b++)

{

tpixel[a, b] = pixels[a - space, b];

tpixel[a, tmpW - 1 - b] = pixels[a - space, width - 1 - b];

}

//крайние верхняя и нижняя

for (a = 0; a < space; a++)

for (b = space; b < tmpW - space; b++)

{

tpixel[a, b] = pixels[a, b - space];

tpixel[tmpH - 1 - a, b] = pixels[height - 1 - a, b - space];

}

//углы

for (a = 0; a < space; a++)

for (b = 0; b < space; b++)

{

tpixel[a, tmpW - 1 - b] = pixels[0, width - 1];

tpixel[tmpH - 1 - a, b] = pixels[height - 1, 0];

tpixel[a, b] = pixels[0, 0];

tpixel[tmpH - 1 - a, tmpW - 1 - b] = pixels[height - 1, width - 1];

}

//центр

for (a = 0; a < height; a++)

for (b = 0; b < width; b++)

tpixel[a + space, b + space] = pixels[a, b];

//применение ядра свертки

RGB ColorOfPixel = new RGB();

RGB ColorOfCell= new RGB();

for (a = space; a < tmpH - space; a++)

for (b = space; b < tmpW - space; b++)

{

ColorOfPixel.R = 0;

ColorOfPixel.G = 0;

ColorOfPixel.B = 0;

for (k = 0; k < n; k++)

for (c = 0; c < n; c++)

{

ColorOfCell = calculationOfColor(tpixel[a - space + k, b - space + c], matryx[k, c]);

ColorOfPixel.R += ColorOfCell.R;

ColorOfPixel.G += ColorOfCell.G;

ColorOfPixel.B += ColorOfCell.B;

}

//контролируем переполнение переменных

if (ColorOfPixel.R < 0) ColorOfPixel.R = 0;

if (ColorOfPixel.R > 255) ColorOfPixel.R = 255;

if (ColorOfPixel.G < 0) ColorOfPixel.G = 0;

if (ColorOfPixel.G > 255) ColorOfPixel.G = 255;

if (ColorOfPixel.B < 0) ColorOfPixel.B = 0;

if (ColorOfPixel.B > 255) ColorOfPixel.B = 255;

cpixel[a - space, b - space] = buildChanals(ColorOfPixel);

}

return cpixel;

}

//сборка каналов

public static UInt32 buildChanals(RGB ColorOfPixel)

{

UInt32 Color;

Color = 0xFF000000 | ((UInt32)ColorOfPixel.B) | ((UInt32)ColorOfPixel.R << 16) | ((UInt32)ColorOfPixel.G << 8) ;

return Color;

}

//вычисление нового цвета

public static RGB calculationOfColor(UInt32 pixels, double rate)

{

RGB сolor = new RGB();

сolor.R = (float)(rate \* ((pixels & 0x00FF0000) >> 16));

сolor.G = (float)(rate \* ((pixels & 0x0000FF00) >> 8));

сolor.B = (float)(rate \* (pixels & 0x000000FF));

return сolor;

}

//размытие

public const int N2 = 3;

public static double[,] blur = new double[N1, N1] {{0.111, 0.111, 0.111},

{0.111, 0.111, 0.111},

{0.111, 0.111, 0.111}};

//повышение резкости

public const int N1 = 3;

public static double[,] sharpness = new double[N1, N1] {{-1, -1, -1},

{-1, 9, -1},

{-1, -1, -1}};

}

## }

### Program.cs

using System;

using System.Windows.Forms;

namespace Program

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

## }

### MDI.cs

using System;

using System.Windows.Forms;

namespace Program

{

public partial class MDI : Form

{

public MDI()

{

InitializeComponent();

}

private int childFormNumber = 0;

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

{

MainForm childForm = new MainForm();

childForm.MdiParent = this;

childForm.Text = "Окно " + childFormNumber++;

childForm.Show();

}

}

}

**Вывод**

В ходе курсового проекта было разработано приложение, позволяющее размывать, повышать резкость и изменять яркость изображения и сохранять изменения.

**Список использованных источников и литературы**

1. Книга Герберта Шилдта. C# 4.0: полное руководство.

2. Статья “Фильтрация изображений на C#” URL: <https://vscode.ru/prog-lessons/filtratsiya-izobrazheniy-na-c-sharp.html> .