Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра кібербезпеки та програмного забезпечення

**Звіт**

з виконаної лабораторної роботи № 4

Дисципліна: Комп'ютерна графіка

Виконав :

студент академічної групи КІ-15

Аннаєв А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перевірив :

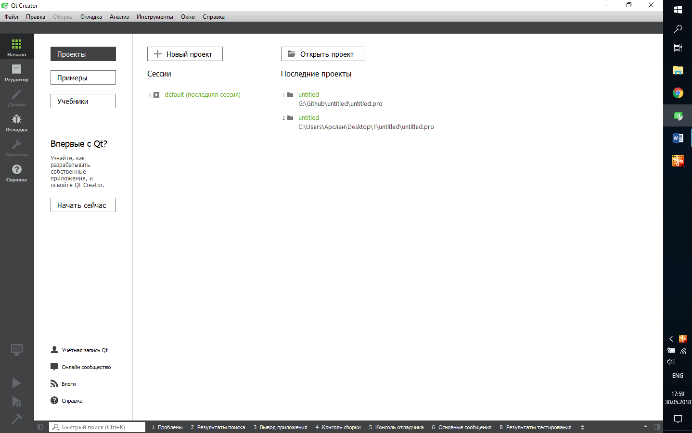
Викладач

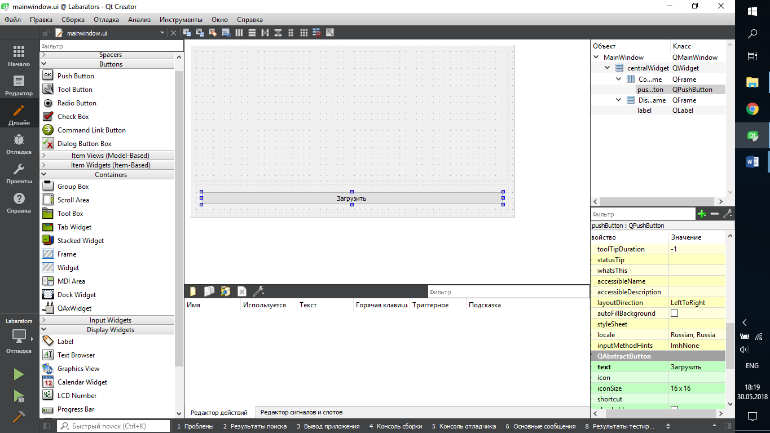
ЛадигiнаО.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кропивницкий- 2018

**Лабораторная работа №4**

Задача для первого раздела:

1. Установите систему Qt Creator.

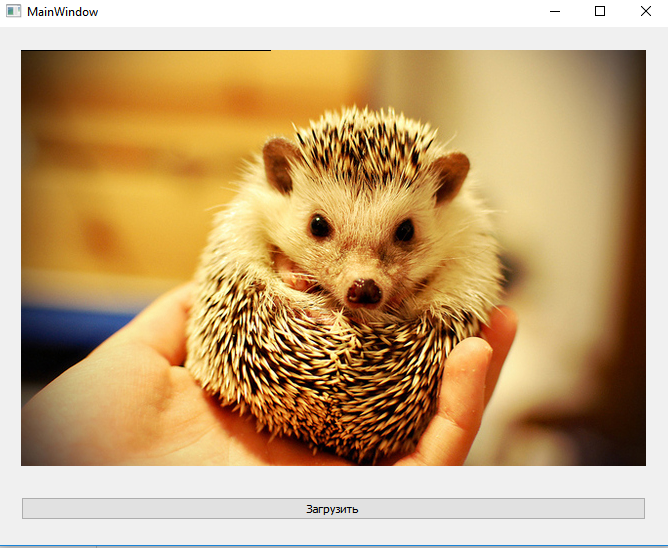


1. Создайте программное обеспечение, как показано в главе 1.2.
2. Откройте одно из изображений, изменяйте размер окна, укажите поведение программы. Запишите пожелания для улучшения действий интерфейса программы.

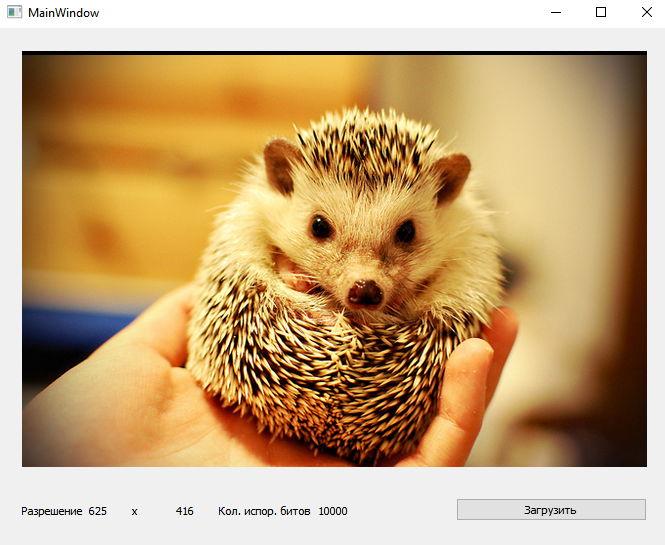
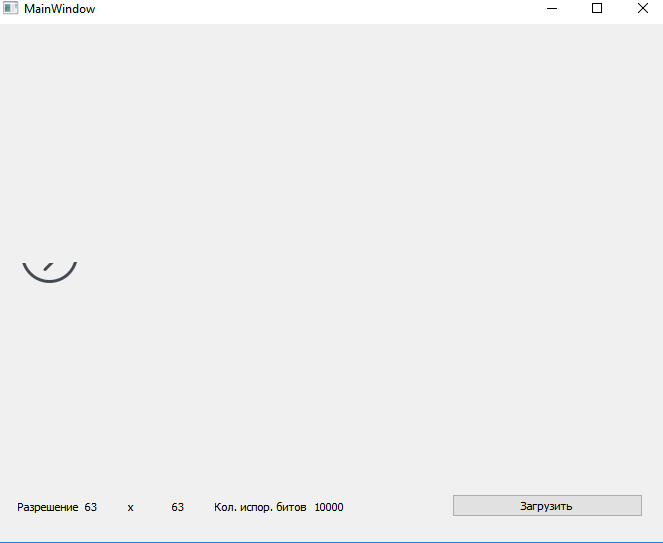
При запуске программы, отображается текст метки, при нажатии на кнопку открывается диалоговое окно выбора изображения. Выбранное изображение отображается в области метки. При изменении размера окна, как в ширину так и в высоту, изображение сохраняет размер а вот главное окно программы изменяется.

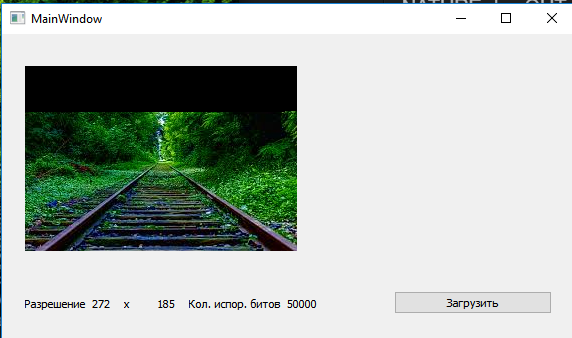
1. Выполните программу с изменением 1000 байтов. Какие результаты от действий замены? Какая часть рисунка изменилась?

При загрузке картинки верхняя часть начиная с верхнего левого края, отображается черной полоской, в зависимости от разрешения загружаемого изображения, меняется ширина полоски.



1. Сделает скриншоты программы с загруженными картинками, где испорченные первые 1000 байтов. В графическом редакторе определите для рисунков разного формата количество испорченных пикселей. Количество байт используется для задания одного пикселя?





#include "task1.h"

#include "ui\_mainwindow.h"

#include <QFileDialog>

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**on\_pushButton\_clicked**()

{

uint32\_t bitsCount = 50000;

QString fileName = QFileDialog::getOpenFileName(this, "Выбор картинки", "C:\\Users\\user\\", "\*.jpg \*.bmp \*.png");

QImage image(fileName);

uchar \*data = image.bits();

for(uint32\_t i = 0; i < bitsCount; i++)

{

data[i] =0;

}

ui->lblBitsCount->setText(QString::number(bitsCount));

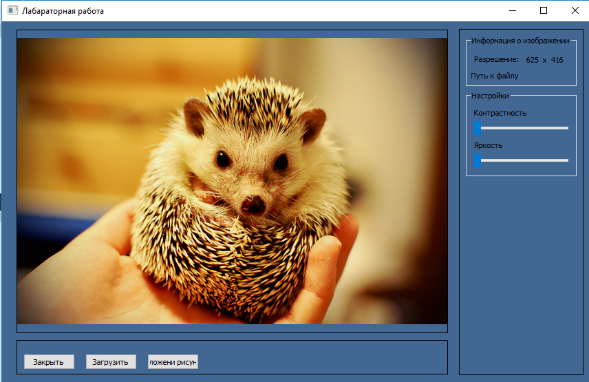
ui->lblWidth->setText(QString::number(image.width()));

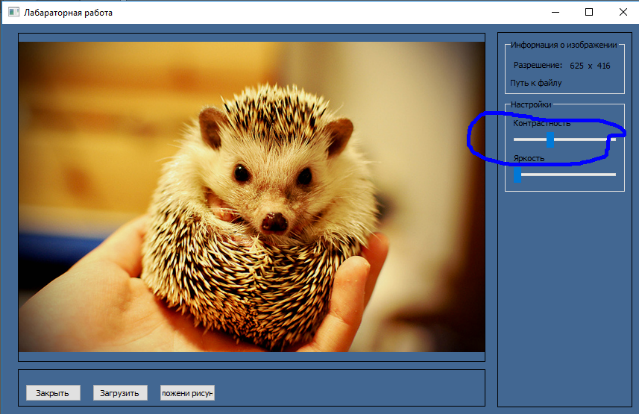
ui->lblHeight->setText(QString::number(image.height()));

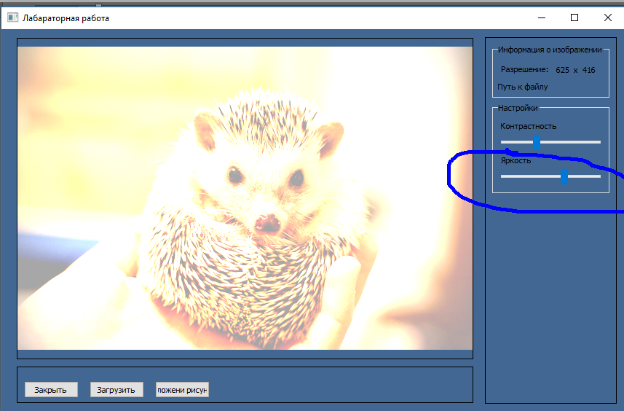
ui->label->setPixmap(QPixmap::fromImage(image));

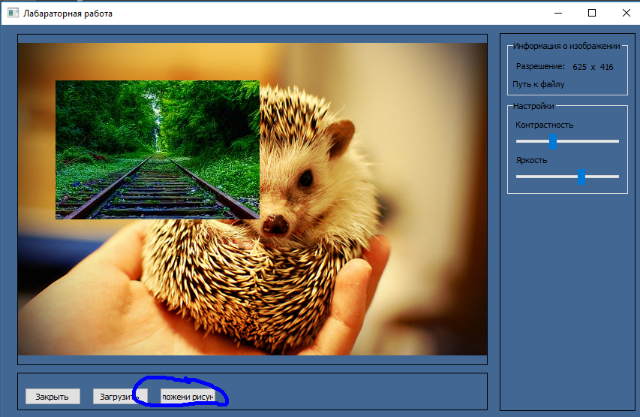
}

Задача для второго раздела:

1. Напишите программу, которая будет содержать место для изображения загруженного рисунка.



1. Реализуйте с помощью ползунков регуляцию яркости и контраста рисунка.
2. Реализуйте автоматическую коррекцию гистограммы загруженного рисунка.
3. Сделайте скриншоты, демонстрирующие работу программы.



1. Реализуйте наложения на загруженный рисунок полуясном элементов.
2. Добавьте в программу средства измерения времени выполнения преобразований, сравните время выполнения преобразований с использованием команд MMX.

**Листинг MainWindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <Mmintrin.h>

#include <QFIleDialog>

#include <QTimer>

#include <QDebug>

#include <QTime>

const int maxByteValue = 255;

const int minByteValue = 0;

const int prescaler = 128;

const int pixelSize = sizeof(uint32\_t);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*// Встраиваемая функция проверки переполнения байта

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

inline int **min**(int a, int b)

{

return (a < b) ? a: b;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

inline int **max**(int a, int b)

{

return (a > b) ? a: b;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Конструктор

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MainWindow::**MainWindow**(QWidget \*parent) : QMainWindow(parent), ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Деструктор

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MainWindow::~***MainWindow***()

{

delete ui;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Изменения контрастности

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MainWindow::**contrast**(double value, QImage \*image)

{

uint8\_t \*pixels = (uchar \*)image->bits();

for(int numPixel = 0; numPixel <(image->width()\*image->height()); numPixel++)

{

pixels[numPixel\*pixelSize+0] = min(maxByteValue,(int)pixels[numPixel\*pixelSize+0]\*value/prescaler);

pixels[numPixel\*pixelSize+1] = min(maxByteValue,(int)pixels[numPixel\*pixelSize+1]\*value/prescaler);

pixels[numPixel\*pixelSize+2] = min(maxByteValue,(int)pixels[numPixel\*pixelSize+2]\*value/prescaler);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Изменения яркости c использование библиотеки mmx

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MainWindow:: **brightnessWithMMX**(double value,QImage \*image)

{

\_\_m64\*pixels = (\_\_m64 \*)image->bits();

\_\_m64 temp = \_mm\_set\_pi8( 0, value, value, value, 0, value, value, value);

for(int numPixel = 0; numPixel <(image->width()\*image->height())/2; numPixel++)

{

pixels[numPixel] = \_mm\_adds\_pu8(pixels[numPixel],temp);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Изменения яркости без использование библиотеки mmx

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MainWindow:: **brightnessWithOutMMX**(double value,QImage \*image)

{

uchar\*pixels = (uchar \*)image->bits();

for(int numPixel = 0; numPixel <(image->width()\*image->height()); numPixel++)

{

pixels[numPixel\*pixelSize+0] = min(maxByteValue,(int)pixels[numPixel\*pixelSize+0] + value);

pixels[numPixel\*pixelSize+1] = min(maxByteValue,(int)pixels[numPixel\*pixelSize+1] + value);

pixels[numPixel\*pixelSize+2] = min(maxByteValue,(int)pixels[numPixel\*pixelSize+2] + value);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Автокоррекция

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MainWindow::**autoCorrection**(QImage \*images)

{

uchar\*pixels = (uchar \*)images->bits();

int pixelCount = (images->width()\*images->height());

uint32\_t correctionTable[maxByteValue];

for(uint32\_t i = 0; i < maxByteValue; i++)

{

correctionTable[i] = 0;

}

for(int numPixel = 0; numPixel <pixelCount; numPixel++)

{

int sum = pixels[numPixel\*pixelSize+0] + 4\* pixels[numPixel\*pixelSize+1] + 3\* pixels[numPixel\*pixelSize+2];

int t = sum >> 3;

correctionTable[t] += 1;

}

double coefficient = 0.025 \* pixelCount;

uint32\_t A = minByteValue;

uint32\_t black = correctionTable[A];

while(true)

{

if(black >= coefficient)

break;

A++;

black += correctionTable[A];

}

uint32\_t B = maxByteValue;

uint32\_t white = correctionTable[B];

while(true)

{

if(white >= coefficient)

break;

B--;

white += correctionTable[B];

}

for(int numPixel = 0; numPixel <pixelCount; numPixel++)

{

pixels[numPixel\*pixelSize+0] = max(0,(int)pixels[numPixel\*pixelSize+0] - A);

pixels[numPixel\*pixelSize+1] = max(0,(int)pixels[numPixel\*pixelSize+1] - A);

pixels[numPixel\*pixelSize+2] = max(0,(int)pixels[numPixel\*pixelSize+2] - A);

}

double value = maxByteValue/(B-A);

for(int numPixel = 0; numPixel < pixelCount; numPixel++)

{

pixels[numPixel\*pixelSize+0] = min(maxByteValue,(int)pixels[numPixel\*pixelSize+0]+value);

pixels[numPixel\*pixelSize+1] = min(maxByteValue,(int)pixels[numPixel\*pixelSize+1]+value);

pixels[numPixel\*pixelSize+2] = min(maxByteValue,(int)pixels[numPixel\*pixelSize+2]+value);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Обработчик нажатия кнопки закрытия приложения

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MainWindow::**on\_btnClose\_clicked**()

{

close();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Обработчик нажатия кнопки загрузки изображения

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MainWindow::**on\_btnLoad\_clicked**()

{

QString filename = QFileDialog::getOpenFileName(this, "Выбор изображения", "C:\\Users\\user\\" ,"Images(\*.png \*.jpg)");

baseImage.load(filename);

autoCorrection(&baseImage);

ui->lblWidth->setText(QString::number(baseImage.width()));

ui->lblHeigth->setText(QString::number(baseImage.height()));

//ui->lblFileName->setText(filename);

ui->imagePointer->setPixmap(QPixmap::fromImage(baseImage));

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Изменения контрастности

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MainWindow::**on\_horizontalSlider\_valueChanged**(int value)

{

QImage image(baseImage);

contrast(value, &image);

ui->imagePointer->setPixmap(QPixmap::fromImage(image));

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Изменения яркости

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MainWindow::**on\_horizontalSlider\_2\_valueChanged**(int value)

{

QTime timer;

timer.start();

QImage image(baseImage);

brightnessWithMMX(value, &image);

// brightnessWithOutMMX(value, &image);

qDebug() << "Время наложения рисунков" << timer.elapsed() << endl;

ui->imagePointer->setPixmap(QPixmap::fromImage(image));

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Наложение

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MainWindow::**overlayWithOutMMX**(QImage \*fone, QImage \*insert)

{

uchar\* fonePixels = (uchar\* )fone->bits();

uchar\* insertPixels = (uchar\* )insert->bits();

int insertImageWidth = insert->width();

int insertImageHeigth = insert->height();

int foneImageWidth = fone->width();

int foneImageHeigth = fone->height();

if(insertImageHeigth > foneImageHeigth || insertImageWidth > foneImageWidth)

{

return;

}

int start\_x = 50;

int start\_y = 50;

int Rf,Bf,Gf,Rn,Gn,Bn,A,fonefixel,insertPixel;

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

{

for(int j = 0; j < insertImageWidth;j++)

{

fonefixel = ((i+start\_x)\*foneImageWidth + j+start\_y)\*pixelSize;

Rf = fonePixels[fonefixel + 2];

Gf = fonePixels[fonefixel + 1];

Bf = fonePixels[fonefixel + 0];

insertPixel = (i\*insertImageWidth + j)\*pixelSize;

A = insertPixels[insertPixel + 3];

Rn = insertPixels[insertPixel + 2];

Gn = insertPixels[insertPixel + 1];

Bn = insertPixels[insertPixel + 0];

fonePixels[fonefixel + 2] = (Rf\*(255 - A) + Rn\*A)/255;

fonePixels[fonefixel + 1] = (Gf\*(255 - A) + Gn\*A)/255;

fonePixels[fonefixel + 0] = (Bf\*(255 - A) + Bn\*A)/255;

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Наложение рисунка

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MainWindow::**on\_pushButton\_clicked**()

{

QString filename = QFileDialog::getOpenFileName(this, "Выбор изображения", "C:\\Users\\user\\" ,"Images(\*.png \*.jpg)");

QImage foneImage(filename);

filename = QFileDialog::getOpenFileName(this, "Выбор изображения", "C:\\Users\\user\\" ,"Images(\*.png \*.jpg)");

QImage insertImage(filename);

overlayWithOutMMX(&foneImage, &insertImage);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

QImage baseImage;

explicit MainWindow(QWidget \*parent = 0);

~***MainWindow***();

void **autoCorrection**(QImage \*image);

void **contrast**(double value,QImage \*image);

void **brightnessWithMMX**(double value,QImage \*image);

void **brightnessWithOutMMX**(double value,QImage \*image);

void **overlayWithOutMMX**(QImage \*fone, QImage \*insert);

private slots:

void **on\_btnClose\_clicked**();

void **on\_btnLoad\_clicked**();

void **on\_horizontalSlider\_valueChanged**(int value);

void **on\_horizontalSlider\_2\_valueChanged**(int value);

void **on\_pushButton\_clicked**();

private:

Ui::MainWindow \*ui;

};

