Министерство Образования и Науки РФ

Новосибирский Государственный Технический Университет

Кафедра вычислительной техники

**Курсовая работа**

по дисциплине «Языки программирования»

на тему «Обработка изображения в двоичных файлах»

Факультет: **АВТФ**

Группа: **АВТ-615**

Студент: **Кузенков В.С.**

Преподаватель: **Балагуров М.В.**

Новосибирск

2017

Оглавление

[Задание 2](#_Toc485938655)

[Структурное описание разработки 2](#_Toc485938656)

[Информация о графических форматах 2](#_Toc485938657)

[Описание формата BMP 2](#_Toc485938658)

[Структура формата BMP 3](#_Toc485938659)

[Преобразования изображения 6](#_Toc485938660)

[Яркость 6](#_Toc485938661)

[Контрастность 7](#_Toc485938662)

[Уменьшение разрешающей способности (2х2, 3х3) 8](#_Toc485938663)

[«Линза» 9](#_Toc485938664)

[Проецирование на сферическую поверхность 10](#_Toc485938665)

[Рисование элементарных графических объектов 10](#_Toc485938666)

[Модульная организация программы 11](#_Toc485938667)

[Класс изображения в программе 12](#_Toc485938668)

[Методы класса изображения 13](#_Toc485938669)

[Функциональное описание разработки 13](#_Toc485938670)

[Вспомогательные функции 13](#_Toc485938671)

[Функции загрузки, сохранения 14](#_Toc485938672)

[Функции редактирования 14](#_Toc485938673)

[Функции отображения, слоты, обработчики событий 14](#_Toc485938674)

[Описание работы программы 16](#_Toc485938675)

[Статистические данные 20](#_Toc485938676)

[Приложение 21](#_Toc485938677)

[Файл bmpEditor.pro 21](#_Toc485938678)

[Файл bmpimage.cpp 22](#_Toc485938679)

[Файл bmpimage.h 33](#_Toc485938680)

[Файл drawalgorithms.cpp 35](#_Toc485938681)

[Файл drawbmpimage.cpp 36](#_Toc485938682)

[Файл drawbmpimage.h 37](#_Toc485938683)

[Файл editcontrastbrightwidget.cpp 37](#_Toc485938684)

[Файл editcontrastbrightwidget.h 38](#_Toc485938685)

[Файл main.cpp 39](#_Toc485938686)

[Файл mainwindow.cpp 39](#_Toc485938687)

[Файл mainwindow.h 43](#_Toc485938688)

[Файл mainwindow.ui 44](#_Toc485938689)

# Задание

Разработать программу, «вручную» считывающую растровый файл с изображением, формата BMP, и выполняющую некоторые преобразования.

1. Изменение яркости и контрастности изображения.
2. Уменьшение разрешающей способности путем усреднения групп пикселей (2х2, 3х3).
3. Искажение изображения («линза», проецирование на сферическую поверхность).
4. Рисование элементарных графических объектов (линии, окружности, точки).

# Структурное описание разработки

Данная программа представляет собой оконное приложение с использованием популярной кроссплатформенной библиотеки QT, обеспечивающей удобную реализацию создания графического интерфейса (GUI).

Сперва стоит получить краткую сводку о графических форматах в общем, затем рассмотреть формат BMP, и его структуру в частности, и уже после только перейти к изложению реализации каждой отдельной функции.

## Информация о графических форматах

В компьютерной графике применяют, по меньшей мере, три десятка форматов файлов для хранения изображений. Форматы графических файлов определяют способ хранения информации в файле (растровый, векторный), а также форму хранения информации (используемый алгоритм сжатия).

Сжатие применяется для растровых графических файлов, т.к. они имеют достаточно большой объем. Существуют различные алгоритмы сжатия, причем для различных типов изображения целесообразно применять подходящие типы алгоритмов сжатия.

По типу хранимой графической информации форматы файлов для хранения изображений разделяются на

— растровые (TIFF, GIF, BMP, JPEG);

— векторные (AI, CDR, FH7, DXF);

— смешанные (универсальные) (EPS, PDF).

Следует учитывать, что файлы практически любого векторного формата позволяют хранить в себе и растровую графику. Однако часто это приводит к искажениям в цветопередаче, поэтому, если изображение не содержит векторных объектов, то предпочтительнее использовать растровые форматы.

Остановимся подробнее на растровом формате BMP.

## Описание формата BMP

BMP или Bitmap Picture – формат растровых изображений, разработанный компанией Microsoft. Файлы этого формата могут иметь расширение .bmp, .dib, а также .rle.

С форматом BMP работает огромное количество программ, так как его поддержка интегрирована в операционные системы [Windows](https://ru.wikipedia.org/wiki/Windows) и [OS/2](https://ru.wikipedia.org/wiki/OS/2). Кроме того, данные этого формата включаются в двоичные файлы ресурсов RES и в [PE-файлы](https://ru.wikipedia.org/wiki/Portable_Executable).

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 больших непосредственным значением. Цвет же в любом случае можно задать только в цветовой модели [RGB](https://ru.wikipedia.org/wiki/RGB) (как при непосредственном указании в пикселе, так и в таблице цветов), но в битностях 16 и 32 можно получить [Grayscale](https://ru.wikipedia.org/wiki/Grayscale) с глубиной до 16 и 32 бит соответственно. Частичная прозрачность реализована [альфа-каналом](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D1%8C%D1%84%D0%B0-%D0%BA%D0%B0%D0%BD%D0%B0%D0%BB) различных битностей, но при этом прозрачность без градаций можно косвенно получить RLE-кодированием.

В последних версиях формата BMP также появились возможности по управлению цветом. В частности, можно указывать конечные точки, производить гамма-коррекцию и встраивать цветовые профили ICC.

## Структура формата BMP

Программа работает с BMP форматом с глубиной цвета 16, 24, 32 бита на пиксель, без сжатия, без палитры.

Для хранения используется двумерный массив RGB структуры, индексами в котором служат y и x координаты, а полями три значения яркости составляющих цвета: Red(красный), Green(зелёный), Blue(Синий).

Данные о формате в BMP состоят из трех блоков различного размера:

1. Заголовок состоит из структуры BITMAPFILEHEADER и блока BITMAPINFO.
2. Цветовой профиль.
3. Пиксельные данные.

В свою очередь структура BITMAPFILEHEADER содержит следующие поля:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поз. (hex)** | **Размер (байты)** | **Имя** | **Тип WinAPI** | **Описание** |
| 00 | 2 | bfType | WORD | Отметка для отличия формата от других (сигнатура формата). Может содержать единственное значение 4D4216/424D16 (little-endian/big-endian). |
| 02 | 4 | bfSize | DWORD | Размер файла в байтах. |
| 06 | 2 | bfReserved1 | WORD | Зарезервированы и должны содержать ноль. |
| 08 | 2 | bfReserved2 | WORD |
| 0A | 4 | bfOffBits | DWORD | Положение пиксельных данных относительно начала данной структуры (в байтах). |

Содержание блока BITMAPINFO разнится в зависимости от версии, а также типа, разрядности и наличии или отсутствии палитры. В среднем состоит из трех частей:

1. Структура с информационными полями.
2. Битовые маски для извлечения значений цветовых каналов (присутствуют не всегда).
3. Таблицы цветов (присутствуют не всегда).

В ходе разработки функции загрузки изображения, было решено остановиться на использовании безпалитрового типа, но при этом поддерживать все версии структуры с информационными полями, которых, на момент написания, четыре: CORE, v3, v4, v5. Для того, чтобы узнать версию, достаточно знать размер структуры: 12 байт для CORE, 40, 108, 124 байта для остальных версий соответственно.

Рассмотрим подробнее структуру для каждой версии.

CORE – 16-битное информационное поле:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Позиция в файле (hex)** | **Позиция в структуре (hex)** | **Размер (байты)** | **Имя** | **Тип WinAPI** | **Описание** |
| 0E | 00 | 4 | bcSize | DWORD | Размер данной структуры в байтах, указывающий также на версию структуры (здесь должно быть значение 12). |
| 12 | 04 | 2 | bcWidth | WORD | Ширина (bcWidth) и высота (bcHeight) растра в пикселях. Указываются целым числом без знака. Значение 0 не документировано. |
| 14 | 06 | 2 | bcHeight | WORD |
| 16 | 08 | 2 | bcPlanes | WORD | В BMP допустимо только значение 1. Это поле используется в значках и курсорах Windows. |
| 18 | 0A | 2 | bcBitCount | WORD | Количество бит на пиксель |

Для достаточного обзора 32-битных информационных полей можно ознакомиться со следующей таблицей:

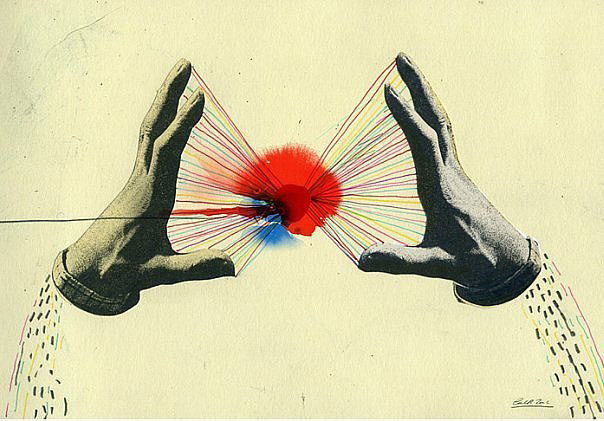
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Позиция в файле (hex)** | **Позиция в структуре (hex)** | **Размер**  **(байты)** | **Имя** | **Тип WinAPI** | **Описание** |
| 0E | 00 | 4 | biSize | DWORD | Размер данной структуры в байтах, указывающий также на версию структуры |
| 12 | 04 | 4 | biWidth | LONG | Ширина растра в пикселях. Указывается целым числом со знаком. |
| 16 | 08 | 4 | biHeight | LONG | Целое число со знаком, содержащее два параметра: высота растра в пикселях и порядок следования строк в двумерных массивах. |
| 1A | 0C | 2 | biPlanes | WORD | В BMP допустимо только значение 1. |
| 1C | 0E | 2 | biBitCount | WORD | Количество бит на пиксель. |
| 1E | 10 | 4 | biCompression | DWORD | Указывает на способ хранения пикселей |
| 22 | 14 | 4 | biSizeImage | DWORD | Размер пиксельных данных в байтах. Может быть обнулено если хранение осуществляется двумерным массивом. |
| 26 | 18 | 4 | biXPelsPerMeter | LONG | Количество пикселей на метр по горизонтали. |
| 2A | 1C | 4 | biYPelsPerMeter | LONG |
| 2E | 20 | 4 | biClrUsed | DWORD | Размер [таблицы цветов](https://ru.wikipedia.org/wiki/BMP#.D0.A2.D0.B0.D0.B1.D0.BB.D0.B8.D1.86.D0.B0_.D1.86.D0.B2.D0.B5.D1.82.D0.BE.D0.B2) в ячейках. |
| 32 | 24 | 4 | biClrImportant | DWORD | Количество ячеек от начала таблицы цветов до последней используемой (включая её саму). |
| 36 | 28 | 4 | biRedMask | DWORD | [Битовые маски для извлечения значений каналов](https://ru.wikipedia.org/wiki/BMP#.D0.9C.D0.B0.D1.81.D0.BA.D0.B8_.D0.BA.D0.B0.D0.BD.D0.B0.D0.BB.D0.BE.D0.B2): интенсивность красного, зелёного, синего и значение альфа-канала. |
| 3A | 2C | 4 | biGreenMask | DWORD |
| 3E | 30 | 4 | biBlueMask | DWORD |
| 42 | 34 | 4 | biAlphaMask | DWORD |
| 46 | 38 | 4 | biCSType | DWORD | Вид [цветового пространства](https://ru.wikipedia.org/wiki/BMP#.D0.A6.D0.B2.D0.B5.D1.82.D0.BE.D0.B2.D0.BE.D0.B5_.D0.BF.D1.80.D0.BE.D1.81.D1.82.D1.80.D0.B0.D0.BD.D1.81.D1.82.D0.B2.D0.BE). |
| 4A | 3C | 36 | biEndpoints | CIEXYZTRIPLE | Значение этих четырёх полей берётся во внимание только если поле CSType содержит 0 (LCS\_CALIBRATED\_RGB). Тогда [конечные точки и значения гаммы](https://ru.wikipedia.org/wiki/BMP#.D0.9A.D0.BE.D0.BD.D0.B5.D1.87.D0.BD.D1.8B.D0.B5_.D1.82.D0.BE.D1.87.D0.BA.D0.B8_.D0.B8_.D0.B7.D0.BD.D0.B0.D1.87.D0.B5.D0.BD.D0.B8.D0.B5_.D0.B3.D0.B0.D0.BC.D0.BC.D1.8B) для трёх цветовых компонент указываются в этих полях. |
| 6E | 60 | 4 | biGammaRed | DWORD[[12]](https://ru.wikipedia.org/wiki/BMP#cite_note-gamma-type-12) |
| 72 | 64 | 4 | biGammaGreen | DWORD[[12]](https://ru.wikipedia.org/wiki/BMP#cite_note-gamma-type-12) |
| 76 | 68 | 4 | biGammaBlue | DWORD[[12]](https://ru.wikipedia.org/wiki/BMP#cite_note-gamma-type-12) |
| 7A | 6C | 4 | biIntent | DWORD | [Предпочтения при рендеринге](https://ru.wikipedia.org/wiki/BMP#.D0.9F.D1.80.D0.B5.D0.B4.D0.BF.D0.BE.D1.87.D1.82.D0.B5.D0.BD.D0.B8.D1.8F_.D0.BF.D1.80.D0.B8_.D1.80.D0.B5.D0.BD.D0.B4.D0.B5.D1.80.D0.B8.D0.BD.D0.B3.D0.B5) растра. |
| 7E | 70 | 4 | biProfileData | DWORD | Смещение в байтах цветового профиля от начала BITMAPINFO. |
| 82 | 74 | 4 | biProfileSize | DWORD | Если в BMP непосредственно включается цветовой профиль, то здесь указывается его размер в байтах. |
| 86 | 78 | 4 | biReserved | DWORD | Зарезервировано и должно быть обнулено. |

## Преобразования изображения

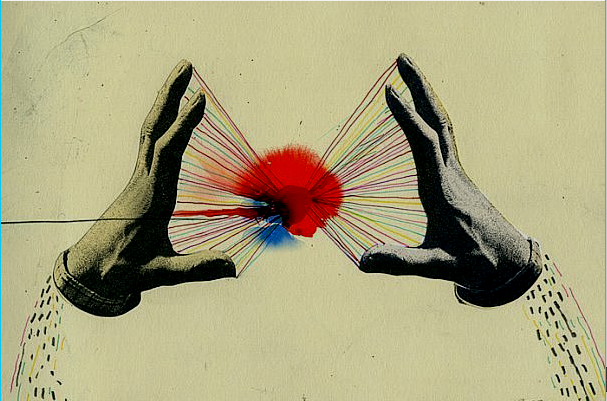
Для обработки изображения программой предусмотрены следующие функции.

### Яркость

До:



После увеличения параметра яркости на 30 единиц:



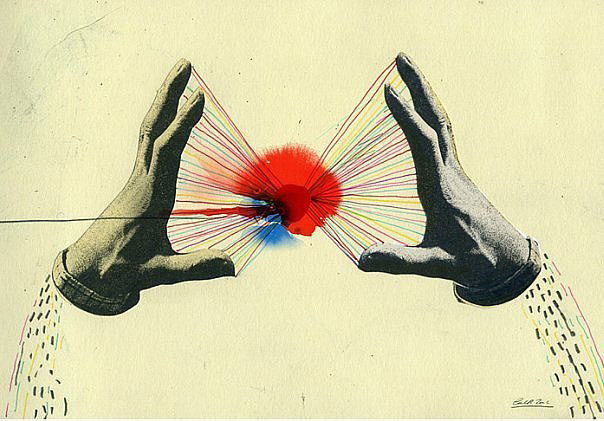
Яркость – световая характеристика тел, отношение силы света, излучаемого поверхностью, к площади ее проекции на плоскости, перпендикулярной оси наблюдения. А если проще, то это количество белого цвета на изображении, и чем выше параметр яркости, тем светлее становится изображение.

Для аддитивного RGB представления цвета, где значение каждого канала цвета – значение его яркости, достаточно прибавлять к нему константу.

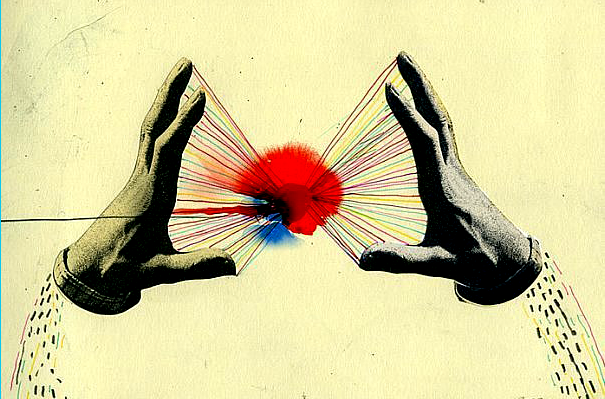
Значение яркости находится в диапазоне от 0 до 255, соответственно для того, чтобы уменьшать яркость (делать изображение темнее) мы можем использовать значение от -1 до -255, а, чтобы увеличивать яркость, будем использовать значение от 1 до 255.

### Контрастность

До:



После увеличения параметра контраста на 30 единиц:



Контрастность - разница между оттенками цвета предмета наблюдения и окружающего его фона. Если же сформулировать проще, то это разница между различными расположенными рядом цветами. Чем выше контрастность, тем более резко мы наблюдаем переход от одного цвета к другому.

При реализации увеличения и уменьшения контрастности, обычно используют значения от -100 до 100.

Для более плавного изменения контрастности, используется среднее значение яркости изображения, а затем на его основе и задаваемом значении коэффициента контраста вычисляется новая яркость каждого пикселя.

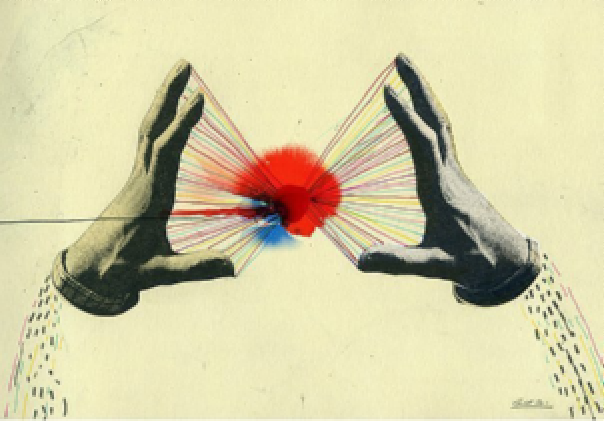
При реализации контраста в приложении, для среднего значения яркости, использовались коэффициенты значимости каждого RGB канала, как для вычисления яркости при преобразовании в YUV-цветовую модель.

Примерная формула для нового цвета красного канала:

newRed = midBright + factor \* (Red-midBright))

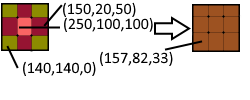
### Уменьшение разрешающей способности (2х2, 3х3)

Сохраненное изображение, после применения функции уменьшения разрешающей способности 3х3:



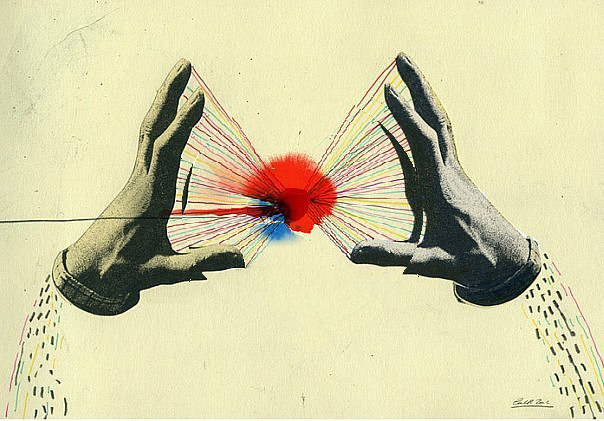
Под уменьшением расширяющей способности, понимается усреднение всех цветов пикселей изображения, относительно их соседей.

Для 3х3 усреднения используются цвета центрального пикселя и восьми его соседей. Для 2х2 – крайний левый, и его сосед справа и два нижних.



### «Линза»

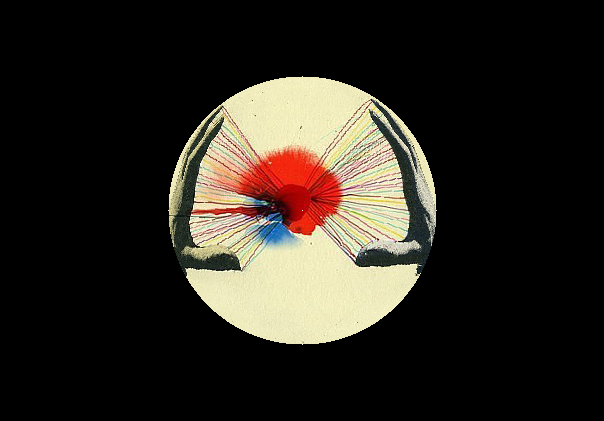
Сохраненное изображение, после применения уменьшающей линзы:



При использовании искажения уменьшающей линзой она автоматически располагается в центре изображения, с размерами и коэффициентами, подобранными эмпирическим путем, для создания достаточно наглядного эффекта искажения линзой.

### Проецирование на сферическую поверхность

Сохраненное изображение, после проецирования на сферическую поверхность:



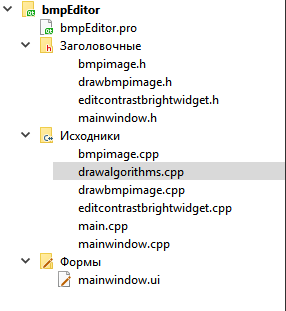
При использовании проецирования на сферическую поверхность, она автоматически располагается в центре изображения, с достаточным радиусом.

### Рисование элементарных графических объектов

Отрезки рисуются при помощи алгоритма Брезенхейма, а круги, при помощи алгоритма Минчера.

## Модульная организация программы

Программа разбита на несколько модулей, каждый из которых содержит текст своего класса (кроме drawalgorithms.cpp). Все объявления вынесены в соответствующий им заголовочный файл.



*bmpEditor.pro* – проектный файл QT, в котором находится информация о подключаемых в проект файлов и библиотек.

*bmpimage.h, bmpimage.cpp* – заголовочный файл и файл реализации класса bmpImage – основного класса изображения в программе, объектом которого и является в дальнейшем загружаемое изображение.

*drawbmbimage.h, drawbmbimage.cpp* – заголовочный файл и файл реализации класса, наследуемого от QWidget, для реализации виджета отрисовки изображения (конструктор принимает параметром указатель на само изображение типа класса bmpImage).

*editcontrastbrightwidget.h, editcontrastbrightwidget.cpp* – заголовочный файл и файл реализации класса, наследуемого от QWidget, для реализации виджета с управляющими элементами, для изменения яркости и контрастности изображения.

*drawalgorithms.cpp* – файл с функциями для рисования графических объектов.

*main.cpp* – файл с точкой входа в программу.

*mainwindow.h, mainwindow.cpp* - заголовочный файл и файл реализации класса, наследуемого от QMainWindow, являющийся классом основного окна приложения. В нем реализованы все элементы управления и логика представления программы.

*mainwindow.ui* – файл QtDesigner-а, подключаемый визуальный интерфейс к классу MainWindow.

## Класс изображения в программе

Для описания переменных в классе изображения вводятся новые структурированные типы: FILEHEADER, INFOHEADER, CIEXYZ, CIEXYZTRIPLE, RGB, а также новое имя типа int - FXPT2DOT30.

typedef struct

{

unsigned short fType;

unsigned int fSize;

unsigned short fReserved1;

unsigned short fReserved2;

unsigned int fOffBits;

}FILEHEADER;

FILEHEADER – структура данных о файле, аналогична описанной в спецификации к BMP.

typedef struct

{

unsigned int infSize;

unsigned int infWidth;

unsigned int infHeight;

unsigned short infPlanes;

unsigned short infBitCount;

unsigned int infCompression;

unsigned int infSizeImage;

unsigned int infXPelsPerMeter;

unsigned int infYPelsPerMeter;

unsigned int infClrUsed;

unsigned int infClrImportant;

unsigned int infRedMask;

unsigned int infGreenMask;

unsigned int infBlueMask;

unsigned int infAlphaMask;

unsigned int infCSType;

CIEXYZTRIPLE infEndpoints;

unsigned int infGammaRed;

unsigned int infGammaGreen;

unsigned int infGammaBlue;

unsigned int infIntent;

unsigned int infProfileData;

unsigned int infProfileSize;

unsigned int infReserved;

}INFOHEADER;

INFOHEADER – структура данных о изображении, аналогична описанной в спецификации к BMP.

typedef struct {

FXPT2DOT30 ciexyzX;

FXPT2DOT30 ciexyzY;

FXPT2DOT30 ciexyzZ;

} CIEXYZ;

typedef struct {

CIEXYZ ciexyzRed;

CIEXYZ ciexyzGreen;

CIEXYZ ciexyzBlue;

} CIEXYZTRIPLE;

CIEXYZ, CIEXYZTRIPLE – структуры данных о цветовой модели.

RGB – структура данных с полями Blue, Green, Red и Reserved.

## Методы класса изображения

bmpImage()– конструктор класса.

bmpImage(bmpImage &image) – конструктор копирования.

~bmpImage() – деструктор.

bool loadImage(ifstream &stream) – метод для загрузки изображения.

bool saveImage(ofstream &stream) – метод для сохранения изображения.

Геттеры:

int getWidth()

int getHeight()

int getRed(int x, int y)

int getGreen(int x, int y)

int getBlue(int x, int y)

int getType()

void setPixel(unsigned int x,unsigned int y,int red, int green, int blue) – метод для изменения цвета пикселя, используется в рисовании.

Методы преобразования изображения:

void editContrast(float n)

void editBright(int n)

void extension()

void extension2()

void lens()

void sphere()

# Функциональное описание разработки

## Вспомогательные функции

void read(ifstream &fp, Type &result, size\_t size)

Шаблонная функция для чтения из файла. Внутри нее вызывается стандартный метод read из ifstream, производится приведение типа Type без проверки.

void write(ofstream &fp, Type &result, size\_t size)

Шаблонная функция для записи в файл. Внутри нее вызывается стандартный метод write из ifstream, производится приведение типа Type без проверки.

Type Truncate(Type value)

Шаблонная функция для проверки значения на промежуток от 0 до 255, используется для проверки цветов, при записи в массив цветов.

unsigned char bitextract(const unsigned int byte, const unsigned int mask)

Функция для применения цветовой маски, используется при чтении цветов в массив.

int sign(float a)

Сигнум, функция индикации знака числа (1, если больше 0, 0, если 0, -1, если передаваемое значение меньше единицы)

int bmpImage::getHeight()

Геттер высоты изображения класса bmpImage.

int bmpImage::getWidth()

Геттер ширины изображения класса bmpImage.

int bmpImage::getGreen(int x, int y)

Геттер значения зеленого цвета bmpImage из массива rgb[y][x].

int bmpImage::getRed(int x, int y)

Геттер значения красного цвета bmpImage из массива rgb[y][x].

int bmpImage::getBlue(int x, int y)

Геттер значения синего цвета bmpImage из массива rgb[y][x].

int bmpImage::getType()

Геттер для типа изображения (поле fType структуры FILEHEADER), здесь -1 используется как дополнительный флаг проверки, загружено ли изображение. Стандартный конструктор инициализирует поле fType минус единицей именно для этой цели.

void bmpImage::setPixel(unsigned int x,unsigned int y,int red, int green, int blue)

Сеттер пикселя по координатам x, y и цветам.

void pixelCircleM(bmpImage \*image, int centerX, int centerY, int x, int y, int r, int g, int b)

Кусок функции алгоритма Минчера.

void BresenhamAlgorithm(bmpImage \*image, int x1, int y1, int x2, int y2, int r, int g, int b)

Функция, реализующая алгоритм Брезенхейма для рисования отрезков.

void MichinerCircle (bmpImage \*image, int centerX, int centerY, int radius, int r, int g, int b)

Функция, реализующая алгоритм Минчера для рисования окружностей.

## Функции загрузки, сохранения

bool bmpImage::loadImage(ifstream &stream)

Загрузка изображения из файла в соответствующие поля структур. Само изображение записывается в поля двумерного массива структуры RGB.

bool bmpImage::saveImage(ofstream &stream)

Сохранение всех полей, структур класса в файл.

## Функции редактирования

void bmpImage::editBright(int factor)

Функция изменения яркости.

void bmpImage::editContrast(float factor)

Функция изменения контраста.

void bmpImage::extension()

Функция уменьшения разрешающей способности путем усреднения групп пикселей 3х3.

void bmpImage::extension2()

Функция уменьшения разрешающей способности путем усреднения групп пикселей 2х2.

void bmpImage::lens()

Функция искажения изображения «линза».

void bmpImage::sphere()

Функция искажения изображения «проецирование на сферическую поверхность».

## Функции отображения, слоты, обработчики событий

void drawBMPImage::*paintEvent*(QPaintEvent \*)

Перегруженный метод рисования для отрисовки изображения на виджете.

void editContrastBrightWidget::editBright()

Слот для вызова метода изменения яркости изображения (в Qt используется механизм сигнал-слот для реализации шаблона «наблюдатель», как связь между объектами).

void editContrastBrightWidget::editContrast()

Слот для вызова метода изменения контраста изображения.

void MainWindow::on\_openFileAction\_triggered()

Стандартный сигнал, обрабатывающий выбор открытия файла в меню.

void MainWindow::on\_saveAsAction\_triggered()

Стандартный сигнал, обрабатывающий выбор сохранения файла в меню.

void MainWindow::on\_contrastAction\_triggered()

Стандартный сигнал, обрабатывающий выбор изменения яркости или контраста в меню.

void MainWindow::on\_extensionAction\_triggered()

Стандартный сигнал, обрабатывающий выбор уменьшения разрешающей способности путем усреднения групп пикселей 3х3 в меню.

void MainWindow::on\_extension2Action\_triggered()

Стандартный сигнал, обрабатывающий выбор уменьшения разрешающей способности путем усреднения групп пикселей 2х2 в меню.

void MainWindow::on\_lensAction\_triggered()

Стандартный сигнал, обрабатывающий выбор искажения линзой в меню.

void MainWindow::on\_sphereAction\_triggered()

Стандартный сигнал, обрабатывающий выбор искажения «проецированием на сферическую поверхность» в меню.

void MainWindow::on\_lineAction\_triggered()

Стандартный сигнал, обрабатывающий выбор рисования линии в меню.

void MainWindow::on\_circleAction\_triggered()

Стандартный сигнал, обрабатывающий выбор рисования окружности в меню.

void MainWindow::on\_dotAction\_triggered()

Стандартный сигнал, обрабатывающий выбор рисования точки в меню.

void MainWindow::reDrawImage()

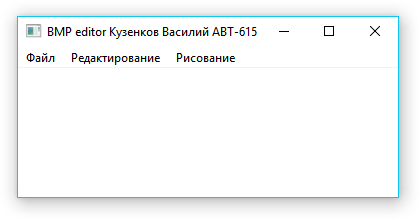
Сигнал, при вызове которого перерисовывается изображение.

void MainWindow::*mousePressEvent*(QMouseEvent \*event)

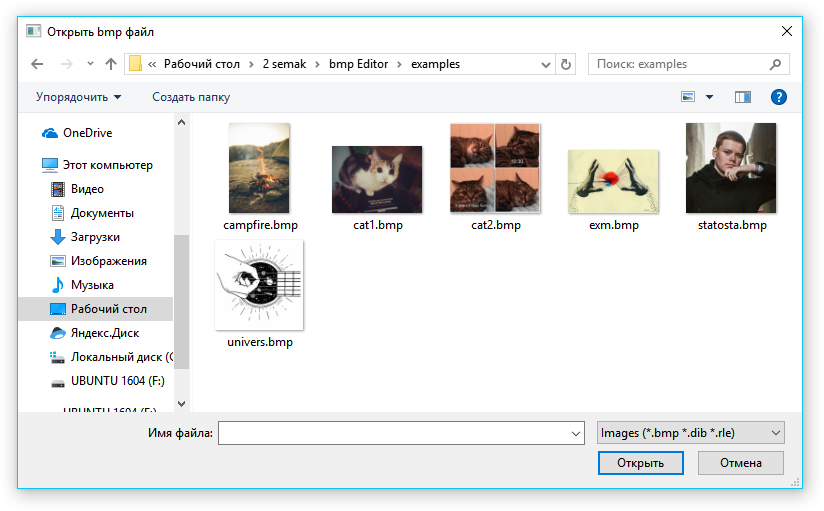
Перегруженная функция, обработчика нажатия клавиш мышки.

# Описание работы программы

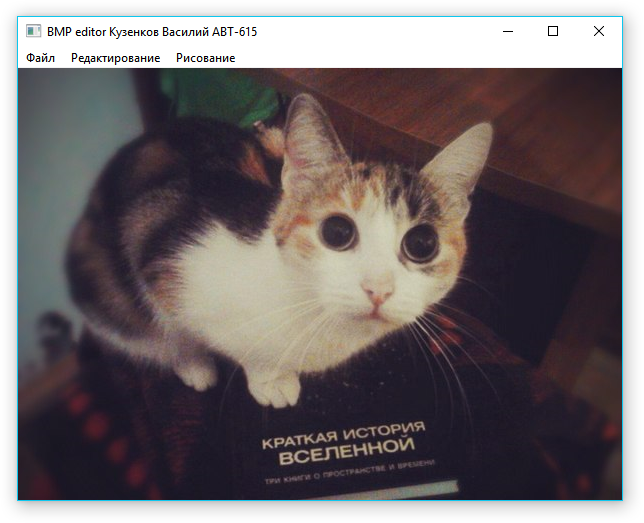
При включении программы открывается пустое окно. Для начала работы следует открыть файл. Для этого достаточно нажать *Файл -> Открыть bmp…* или нажать сочетание клавиш Ctrl+O.



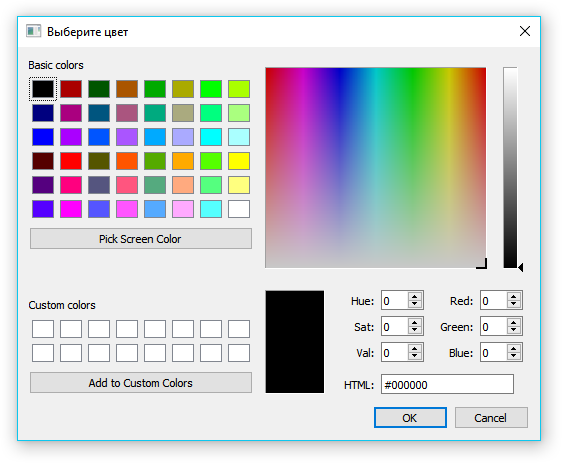
Откроется окно выбора файла. Программа поддерживает BMP изображения 16, 24, 32 разрядности. Любое jpg или png изображение можно открыть в стандартном редакторе изображений, к примеру, в Paint, и сохранить в нужной разрядности. При открытии файла, в пути допустима только латиница.



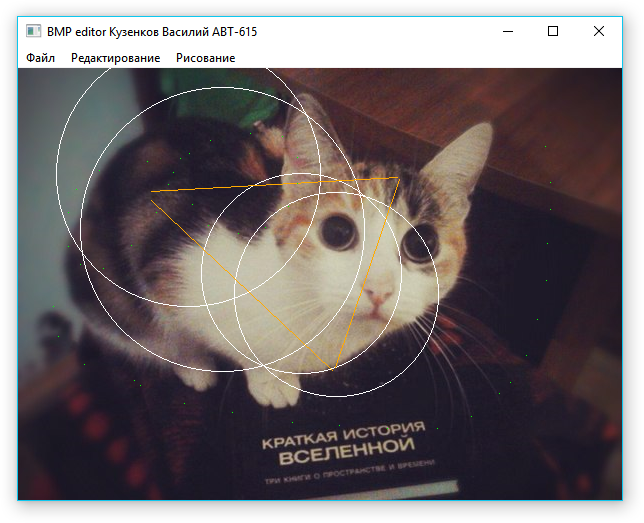
После выбора изображения, оно отобразится в программе, и, если оно больше исходного размера окна, окно автоматически подстроится под размер.



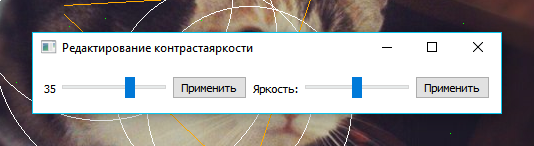
В подменю «Редактирование» находятся основные функции преобразования изображения, а в подменю «Рисование», основные элементарные графические объекты для рисования. Откроем *Рисование -> Окружности*. Отобразится диалоговое окно выбора цвета.



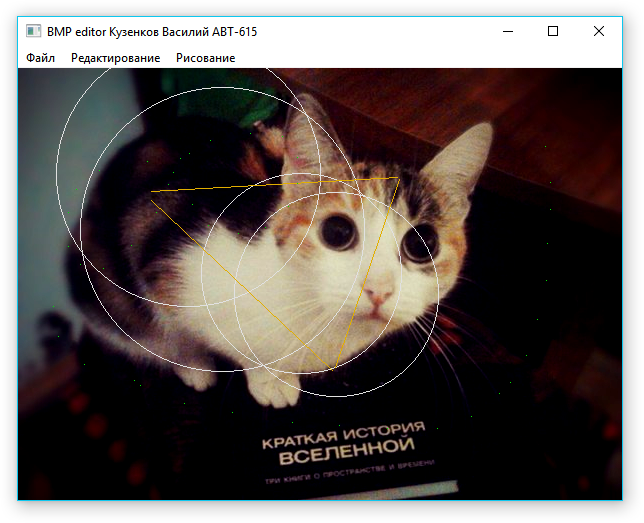
Для рисования кругов на изображении, достаточно нажать на нужное место, первая выбранная точка послужит центром окружности, а вторая его радиусом. При рисовании отрезка – первая точка будет его началом, а вторая концом.



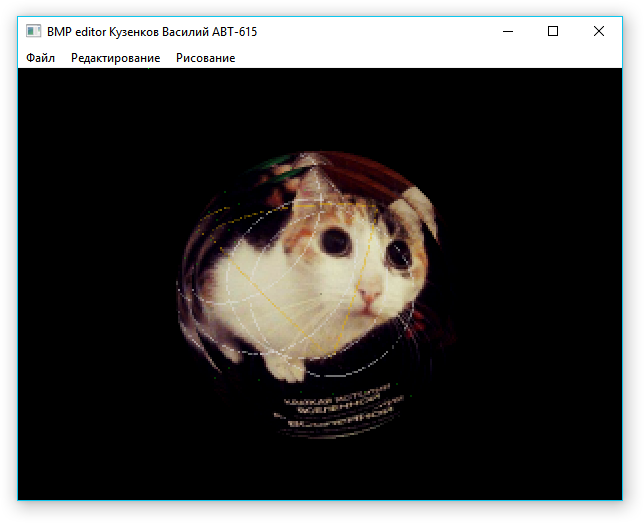
В подменю «Редактирование» выберем «Контраст\Яркость». Откроется окно задания параметров яркости и контраста. С помощью слайдера передвинем бегунок «Контраста» вправо на 35(после передвижения бегунка значение отобразится вместо «Контраст:»).



Нажмем применить, затем убавим яркость до -30, снова применим. Закроем окно редактирования, и получится следующее изображение:



В подменю «Редактирование» выберем «Сфера (fisheye)». Получим изображение, проецированное на сферическую поверхность (не правда ли похоже на известный эффект «рыбий глаз»), затем добавим искажение линзой, выбрав *Редактирование -> Линза,* а после еще изменим разрешающую способность, выбрав *Редактирование -> Разрешение 2х2.* Получим:



Все, можно сохранять нашу работу. Для этого выбираем *Файл -> Сохранить как…* или нажимаем сочетание клавиш Ctrl+Shift+S. По умолчанию файл называется untilted.bmp, переименовываем, сохраняем. Проверяем. Готово! Наш файл доступен во всех редакторах и программах просмотра изображения, поддерживающих BMP формат.

# Статистические данные

Для определения производительности и чувствительности программы к данным, протестируем ее основные методы на различных изображениях. А именно: загрузку, сохранение, изменение разрешения, а также проецирование на сферу.

Теоретически, все алгоритмы компьютерной графики имеют линейную трудоемкость т.к. обрабатывают все точки изображения или какую-то его часть, обработка которых опять же зависит от общего числа точек.

1. Загрузка из двоичного файла

|  |  |
| --- | --- |
| **Количество точек изображения** | **Время работы** |
| 374500 | 141 |
| 273612 | 127 |
| 256500 | 80 |
| 3133440 | 770 |
| 1166400 | 500 |
| 1498176 | 519 |

1. Сохранение в двоичный файл

|  |  |
| --- | --- |
| **Количество точек изображения** | **Время работы** |
| 374500 | 187 |
| 273612 | 138 |
| 256500 | 130 |
| 3133440 | 1666 |
| 1166400 | 602 |
| 1498176 | 772 |

1. Изменение разрешения

|  |  |
| --- | --- |
| **Количество точек изображения** | **Время работы** |
| 374500 | 87 |
| 273612 | 71 |
| 256500 | 63 |
| 3133440 | 393 |
| 1166400 | 172 |
| 1498176 | 204 |

1. Проецирование на сферу

|  |  |
| --- | --- |
| **Количество точек изображения** | **Время работы** |
| 374500 | 157 |
| 273612 | 108 |
| 256500 | 104 |
| 3133440 | 927 |
| 1166400 | 378 |
| 1498176 | 469 |

Из приведенных таблиц и графиков, можно сделать вывод, что экспериментальные данные согласуются с теоретическими сведениями, и при увеличении числа точек, растет и время, затрачиваемое на операции. Также при увеличении числа точек, растет «грязное» время (программа запускается на компьютере, где помимо нее, открыто еще множество других программ).

В связи с этим, можно предположить, что все остальные методы будут иметь похожую зависимость, т.к. также обрабатывают все точки изображения, или его некоторую часть, которая все равно зависит от общего числа точек.

# Приложение

## Файл bmpEditor.pro

#-------------------------------------------------

#

# Project created by QtCreator 2017-06-07T13:15:11

#

#-------------------------------------------------

QT += core gui

#CONFIG += console

greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets

TARGET = bmpEditor

TEMPLATE = app

# The following define makes your compiler emit warnings if you use

# any feature of Qt which as been marked as deprecated (the exact warnings

# depend on your compiler). Please consult the documentation of the

# deprecated API in order to know how to port your code away from it.

DEFINES += QT\_DEPRECATED\_WARNINGS

# You can also make your code fail to compile if you use deprecated APIs.

# In order to do so, uncomment the following line.

# You can also select to disable deprecated APIs only up to a certain version of Qt.

#DEFINES += QT\_DISABLE\_DEPRECATED\_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

SOURCES += main.cpp\

mainwindow.cpp \

bmpimage.cpp \

drawbmpimage.cpp \

editcontrastbrightwidget.cpp \

drawalgorithms.cpp

HEADERS += mainwindow.h \

bmpimage.h \

drawbmpimage.h \

editcontrastbrightwidget.h

FORMS += mainwindow.ui

## Файл bmpimage.cpp

#include "bmpimage.h"

/\*Программа «вручную» читает растровый файл изображения (bmp), выполняя заданные преобразования изображения\*/

template <typename Type>

void read(ifstream &fp, Type &result, size\_t size) {

fp.read(reinterpret\_cast<char\*>(&result), size);

}

template <typename Type>

void write(ofstream &fp, Type &result, size\_t size) {

fp.write(reinterpret\_cast<char\*>(&result), size);

}

template <typename Type>

Type Truncate(Type value){

if(value < 0) value = 0;

if(value > 255) value = 255;

return value;

}

unsigned char bitextract(const unsigned int byte, const unsigned int mask) {

if (mask == 0) {

return 0;

}

// определение количества нулевых бит справа от маски

int maskBufer = mask;

int maskPadding = 0;

while (!(maskBufer & 1)) {

maskBufer >>= 1;

maskPadding++;

}

// применение маски и смещение

return (byte & mask) >> maskPadding;

}

bmpImage::bmpImage()

{

fileHeader = new FILEHEADER;

infoHeader = new INFOHEADER;

rgb = new RGB\*;

fileHeader->fType = 1;

}

bmpImage::bmpImage(bmpImage &image)

{

//конструктор копирования

fileHeader = new FILEHEADER;

infoHeader = new INFOHEADER;

fileHeader->fType = image.fileHeader->fType;

fileHeader->fSize = image.fileHeader->fSize;

fileHeader->fReserved1 = image.fileHeader->fReserved1;

fileHeader->fReserved2 = image.fileHeader->fReserved2;

fileHeader->fOffBits = image.fileHeader->fOffBits;

infoHeader->infSize = image.infoHeader->infSize;

infoHeader->infWidth = image.infoHeader->infWidth;

infoHeader->infHeight = image.infoHeader->infHeight;

infoHeader->infPlanes = image.infoHeader->infPlanes;

infoHeader->infBitCount = image.infoHeader->infBitCount;

infoHeader->infCompression = image.infoHeader->infCompression;

infoHeader->infSizeImage = image.infoHeader->infSizeImage;

infoHeader->infXPelsPerMeter = image.infoHeader->infXPelsPerMeter;

infoHeader->infYPelsPerMeter = image.infoHeader->infYPelsPerMeter;

infoHeader->infClrUsed = image.infoHeader->infClrUsed;

infoHeader->infClrImportant = image.infoHeader->infClrImportant;

infoHeader->infRedMask = image.infoHeader->infRedMask;

infoHeader->infGreenMask = image.infoHeader->infGreenMask;

infoHeader->infBlueMask = image.infoHeader->infBlueMask;

infoHeader->infAlphaMask = image.infoHeader->infAlphaMask;

infoHeader->infCSType = image.infoHeader->infCSType;

infoHeader->infEndpoints = image.infoHeader->infEndpoints;

infoHeader->infGammaRed = image.infoHeader->infGammaRed;

infoHeader->infGammaGreen = image.infoHeader->infGammaGreen;

infoHeader->infGammaBlue = image.infoHeader->infGammaBlue;

infoHeader->infIntent = image.infoHeader->infIntent;

infoHeader->infProfileData = image.infoHeader->infProfileData;

infoHeader->infProfileSize = image.infoHeader->infProfileSize;

infoHeader->infReserved = image.infoHeader->infReserved;

rgb = new RGB\*[infoHeader->infHeight];

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

rgb[i] = new RGB[infoHeader->infWidth];

}

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

for (unsigned int j = 0; j < infoHeader->infWidth; j++) {

rgb[i][j].Blue = image.rgb[i][j].Blue;

rgb[i][j].Green= image.rgb[i][j].Green;

rgb[i][j].Red = image.rgb[i][j].Red;

}

}

}

bmpImage::~bmpImage()

{

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

delete[] rgb[i];

}

delete[] rgb;

delete fileHeader;

delete infoHeader;

}

bool bmpImage::loadImage(ifstream &stream){

qDebug() << "At reading file.\n";

//считываем заголовок изображения

read(stream, fileHeader->fType, sizeof(fileHeader->fType));

read(stream, fileHeader->fSize, sizeof(fileHeader->fSize));

read(stream, fileHeader->fReserved1, sizeof(fileHeader->fReserved1));

read(stream, fileHeader->fReserved2, sizeof(fileHeader->fReserved2));

read(stream, fileHeader->fOffBits, sizeof(fileHeader->fOffBits));

if(fileHeader->fType != 0x4D42){

qDebug() << "Error: It's not BMP image.\n";

return 0;

}

//информация изображения

read(stream, infoHeader->infSize, sizeof(infoHeader->infSize));

// bmp core

if (infoHeader->infSize >= 12) {

read(stream, infoHeader->infWidth, sizeof(infoHeader->infWidth));

read(stream, infoHeader->infHeight, sizeof(infoHeader->infHeight));

read(stream, infoHeader->infPlanes, sizeof(infoHeader->infPlanes));

read(stream, infoHeader->infBitCount, sizeof(infoHeader->infBitCount));

}

// получаем информацию о битности

int colorsCount = infoHeader->infBitCount >> 3;

if (colorsCount < 3) {

colorsCount = 3;

}

int bitsOnColor = infoHeader->infBitCount / colorsCount;

int maskValue = (1 << bitsOnColor) - 1;

// bmp v1

if (infoHeader->infSize >= 40) {

read(stream, infoHeader->infCompression, sizeof(infoHeader->infCompression));

read(stream, infoHeader->infSizeImage, sizeof(infoHeader->infSizeImage));

read(stream, infoHeader->infXPelsPerMeter, sizeof(infoHeader->infXPelsPerMeter));

read(stream, infoHeader->infYPelsPerMeter, sizeof(infoHeader->infYPelsPerMeter));

read(stream, infoHeader->infClrUsed, sizeof(infoHeader->infClrUsed));

read(stream, infoHeader->infClrImportant, sizeof(infoHeader->infClrImportant));

}

// bmp v2

infoHeader->infRedMask = 0;

infoHeader->infGreenMask = 0;

infoHeader->infBlueMask = 0;

if (infoHeader->infSize >= 52) {

read(stream, infoHeader->infRedMask, sizeof(infoHeader->infRedMask));

read(stream, infoHeader->infGreenMask, sizeof(infoHeader->infGreenMask));

read(stream, infoHeader->infBlueMask, sizeof(infoHeader->infBlueMask));

}

// если маска не задана, то ставим маску по умолчанию

if (infoHeader->infRedMask == 0 || infoHeader->infGreenMask == 0 || infoHeader->infBlueMask == 0) {

infoHeader->infRedMask = maskValue << (bitsOnColor \* 2);

infoHeader->infGreenMask = maskValue << bitsOnColor;

infoHeader->infBlueMask = maskValue;

}

// bmp v3

if (infoHeader->infSize >= 56) {

read(stream, infoHeader->infAlphaMask, sizeof(infoHeader->infAlphaMask));

} else {

infoHeader->infAlphaMask = maskValue << (bitsOnColor \* 3);

}

// bmp v4

if (infoHeader->infSize >= 108) {

read(stream, infoHeader->infCSType, sizeof(infoHeader->infCSType));

read(stream, infoHeader->infEndpoints, sizeof(infoHeader->infEndpoints));

read(stream, infoHeader->infGammaRed, sizeof(infoHeader->infGammaRed));

read(stream, infoHeader->infGammaGreen, sizeof(infoHeader->infGammaGreen));

read(stream, infoHeader->infGammaBlue, sizeof(infoHeader->infGammaBlue));

}

// bmp v5

if (infoHeader->infSize >= 124) {

read(stream, infoHeader->infIntent, sizeof(infoHeader->infIntent));

read(stream, infoHeader->infProfileData, sizeof(infoHeader->infProfileData));

read(stream, infoHeader->infProfileSize, sizeof(infoHeader->infProfileSize));

read(stream, infoHeader->infReserved, sizeof(infoHeader->infReserved));

}

// проверка на поддерку этой версии формата

if (infoHeader->infSize != 12 && infoHeader->infSize != 40 && infoHeader->infSize != 52 &&

infoHeader->infSize != 56 && infoHeader->infSize != 108 && infoHeader->infSize != 124) {

qDebug() << "Error: Unsupported BMP format.\n";

return 0;

}

if (infoHeader->infBitCount != 16 && infoHeader->infBitCount != 24 && infoHeader->infBitCount != 32) {

qDebug() << "Error: Unsupported BMP bit count.\n";

return 0;

}

if (infoHeader->infCompression != 0 && infoHeader->infCompression != 3) {

qDebug() << "Error: Unsupported BMP compression.\n";

return 0;

}

qDebug() << "Width:" << infoHeader->infWidth << "Height" << infoHeader->infHeight;

rgb = new RGB\*[infoHeader->infHeight];

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

rgb[i] = new RGB[infoHeader->infWidth];

}

int linePadding = ((infoHeader->infWidth \* (infoHeader->infBitCount / 8)) % 4) & 3; //отсуп в конце каждой строки

// чтение цвета

unsigned int bufer;

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

for (unsigned int j = 0; j < infoHeader->infWidth; j++) {

read(stream, bufer, infoHeader->infBitCount / 8);

rgb[i][j].Red = bitextract(bufer, infoHeader->infRedMask);

rgb[i][j].Green = bitextract(bufer, infoHeader->infGreenMask);

rgb[i][j].Blue = bitextract(bufer, infoHeader->infBlueMask);

rgb[i][j].Reserved = bitextract(bufer, infoHeader->infAlphaMask);

}

stream.seekg(linePadding, std::ios\_base::cur);

}

qDebug() << "Image Done!\n";

return 1;

}

bool bmpImage::saveImage(ofstream &stream){

qDebug() << "At writing file.\n";

// записываем заголовок файла

write(stream, fileHeader->fType, sizeof(fileHeader->fType));

write(stream, fileHeader->fSize, sizeof(fileHeader->fSize));

write(stream, fileHeader->fReserved1, sizeof(fileHeader->fReserved1));

write(stream, fileHeader->fReserved2, sizeof(fileHeader->fReserved2));

write(stream, fileHeader->fOffBits, sizeof(fileHeader->fOffBits));

// записываем заголовок изображения

write(stream, infoHeader->infSize, sizeof(infoHeader->infSize));

// bmp core

if (infoHeader->infSize >= 12) {

write(stream, infoHeader->infWidth, sizeof(infoHeader->infWidth));

write(stream, infoHeader->infHeight, sizeof(infoHeader->infHeight));

write(stream, infoHeader->infPlanes, sizeof(infoHeader->infPlanes));

write(stream, infoHeader->infBitCount, sizeof(infoHeader->infBitCount));

}

// bmp v1

if (infoHeader->infSize >= 40) {

write(stream, infoHeader->infCompression, sizeof(infoHeader->infCompression));

write(stream, infoHeader->infSizeImage, sizeof(infoHeader->infSizeImage));

write(stream, infoHeader->infXPelsPerMeter, sizeof(infoHeader->infXPelsPerMeter));

write(stream, infoHeader->infYPelsPerMeter, sizeof(infoHeader->infYPelsPerMeter));

write(stream, infoHeader->infClrUsed, sizeof(infoHeader->infClrUsed));

write(stream, infoHeader->infClrImportant, sizeof(infoHeader->infClrImportant));

}

// bmp v2

if (infoHeader->infSize >= 52) {

write(stream, infoHeader->infRedMask, sizeof(infoHeader->infRedMask));

write(stream, infoHeader->infGreenMask, sizeof(infoHeader->infGreenMask));

write(stream, infoHeader->infBlueMask, sizeof(infoHeader->infBlueMask));

}

// bmp v3

if (infoHeader->infSize >= 56) {

write(stream, infoHeader->infAlphaMask, sizeof(infoHeader->infAlphaMask));

}

// bmp v4

if (infoHeader->infSize >= 108) {

write(stream, infoHeader->infCSType, sizeof(infoHeader->infCSType));

write(stream, infoHeader->infEndpoints, sizeof(infoHeader->infEndpoints));

write(stream, infoHeader->infGammaRed, sizeof(infoHeader->infGammaRed));

write(stream, infoHeader->infGammaGreen, sizeof(infoHeader->infGammaGreen));

write(stream, infoHeader->infGammaBlue, sizeof(infoHeader->infGammaBlue));

}

// bmp v5

if (infoHeader->infSize >= 124) {

write(stream, infoHeader->infIntent, sizeof(infoHeader->infIntent));

write(stream, infoHeader->infProfileData, sizeof(infoHeader->infProfileData));

write(stream, infoHeader->infProfileSize, sizeof(infoHeader->infProfileSize));

write(stream, infoHeader->infReserved, sizeof(infoHeader->infReserved));

}

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

for (unsigned int j = 0; j < infoHeader->infWidth; j++) {

write(stream, rgb[i][j].Blue, sizeof(unsigned char));

write(stream, rgb[i][j].Green, sizeof(unsigned char));

write(stream, rgb[i][j].Red, sizeof(unsigned char));

}

}

qDebug() << "Image Done!\n";

return 1;

}

int bmpImage::getHeight()

{

return infoHeader->infHeight;

}

int bmpImage::getWidth()

{

return infoHeader->infWidth;

}

int bmpImage::getGreen(int x, int y)

{

return rgb[y][x].Green;

}

int bmpImage::getRed(int x, int y)

{

return rgb[y][x].Red;

}

int bmpImage::getBlue(int x, int y)

{

return rgb[y][x].Blue;

}

int bmpImage::getType()

{

return fileHeader->fType;

}

void bmpImage::setPixel(unsigned int x,unsigned int y,int red, int green, int blue){

if(x < infoHeader->infWidth && y < infoHeader->infHeight && int(infoHeader->infHeight-y) < infoHeader->infHeight){

y = abs(int(infoHeader->infHeight-y));

rgb[y][x].Red = Truncate(red);

rgb[y][x].Green = Truncate(green);

rgb[y][x].Blue = Truncate(blue);

}

}

void bmpImage::editBright(int factor){

if(fileHeader->fType != 1 && infoHeader->infHeight != NULL){

for(unsigned int y = 0; y < infoHeader->infHeight; y++){

for(unsigned int x = 0; x < infoHeader->infWidth; x++){

rgb[y][x].Red = Truncate(factor + rgb[y][x].Red);

rgb[y][x].Green = Truncate(factor + rgb[y][x].Green);

rgb[y][x].Blue = Truncate(factor + rgb[y][x].Blue);

}

}

}else{

qDebug() << "Any problem with image!";

}

}

void bmpImage::editContrast(float factor){

if(fileHeader->fType != 1 && infoHeader->infHeight != NULL){

float midBright = 0, midBright1 = 0, midBright2 = 0;

for(unsigned int y = 0; y < infoHeader->infHeight; y++){

for(unsigned int x = 0; x < infoHeader->infWidth; x++){

midBright += rgb[y][x].Red;

midBright1 += rgb[y][x].Green;

midBright2 += rgb[y][x].Blue;

}

}

midBright = (midBright2 \* 0.114 + midBright1 \* 0.587 + midBright \* 0.299)/(infoHeader->infHeight\*infoHeader->infWidth);

for(unsigned int y = 0; y < infoHeader->infHeight; y++){

for(unsigned int x = 0; x < infoHeader->infWidth; x++){

double newRed = Truncate(midBright + floor(factor \* (rgb[y][x].Red-midBright)));

double newGreen = Truncate(midBright + floor(factor \* (rgb[y][x].Green-midBright)));

double newBlue = Truncate(midBright + floor(factor \* (rgb[y][x].Blue-midBright)));

rgb[y][x].Red = newRed;

rgb[y][x].Green = newGreen;

rgb[y][x].Blue = newBlue;

}

}

}else{

qDebug() << "Any problem with image!";

}

}

void bmpImage::extension(){

//создаем массив ргб цветов

RGB \*\*tempRgb = new RGB\*[infoHeader->infHeight];

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

tempRgb[i] = new RGB[infoHeader->infWidth];

}

//заполняем массив значением -1, как флаг того, что цвет пикселей в этом квадрате 3х3 не был изменен

for(unsigned int y = 0; y < infoHeader->infHeight; y++){

for(unsigned int x = 0; x < infoHeader->infWidth; x++){

tempRgb[y][x].Red = -1;

tempRgb[y][x].Green = -1;

tempRgb[y][x].Blue = -1;

}

}

//средние цвета

int middleR = 0, middleG = 0, middleB = 0;

//заполняем массив средними значениями по квадрату, если какой-то стороны не хватает для обработки 3х3, заполняем сколько можем

unsigned int maxHeight = infoHeader->infHeight - ((infoHeader->infHeight - 1) % 3);

unsigned int maxWidth = infoHeader->infWidth - ((infoHeader->infWidth - 1) % 3);

for(unsigned int y = 1; y < maxHeight - maxHeight % 3; y++){

for(unsigned int x = 1; x < maxWidth - maxHeight % 3; x++){

if(tempRgb[y][x].Red == -1){

middleR = (rgb[y-1][x-1].Red+rgb[y-1][x].Red+rgb[y-1][x+1].Red

+rgb[y][x-1].Red+rgb[y][x].Red+rgb[y][x+1].Red

+rgb[y+1][x-1].Red+rgb[y+1][x].Red+rgb[y+1][x+1].Red)/9;

middleG = (rgb[y-1][x-1].Green+rgb[y-1][x].Green+rgb[y-1][x+1].Green

+rgb[y][x-1].Green+rgb[y][x].Green+rgb[y][x+1].Green

+rgb[y+1][x-1].Green+rgb[y+1][x].Green+rgb[y+1][x+1].Green)/9;

middleB = (rgb[y-1][x-1].Blue+rgb[y-1][x].Blue+rgb[y-1][x+1].Blue

+rgb[y][x-1].Blue+rgb[y][x].Blue+rgb[y][x+1].Blue

+rgb[y+1][x-1].Blue+rgb[y+1][x].Blue+rgb[y+1][x+1].Blue)/9;

tempRgb[y-1][x-1].Red = middleR; tempRgb[y-1][x].Red = middleR; tempRgb[y-1][x+1].Red = middleR;

tempRgb[y][x-1].Red = middleR; tempRgb[y][x].Red = middleR; tempRgb[y][x+1].Red = middleR;

tempRgb[y+1][x-1].Red = middleR; tempRgb[y+1][x].Red = middleR; tempRgb[y+1][x+1].Red = middleR;

tempRgb[y-1][x-1].Green = middleG; tempRgb[y-1][x].Green = middleG; tempRgb[y-1][x+1].Green = middleG;

tempRgb[y][x-1].Green = middleG; tempRgb[y][x].Green = middleG; tempRgb[y][x+1].Green = middleG;

tempRgb[y+1][x-1].Green = middleG; tempRgb[y+1][x].Green = middleG; tempRgb[y+1][x+1].Green = middleG;

tempRgb[y-1][x-1].Blue = middleB; tempRgb[y-1][x].Blue = middleB; tempRgb[y-1][x+1].Blue = middleB;

tempRgb[y][x-1].Blue = middleB; tempRgb[y][x].Blue = middleB; tempRgb[y][x+1].Blue = middleB;

tempRgb[y+1][x-1].Blue = middleB; tempRgb[y+1][x].Blue = middleB; tempRgb[y+1][x+1].Blue = middleB;

}

}

}

//переносим массив в основное изображение

for(unsigned int y = 0; y < maxHeight - maxHeight%3; y++){

for(unsigned int x = 0; x < maxWidth- maxHeight%3; x++){

rgb[y][x].Red = Truncate(tempRgb[y][x].Red);

rgb[y][x].Green = Truncate(tempRgb[y][x].Green);

rgb[y][x].Blue = Truncate(tempRgb[y][x].Blue);

}

}

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

delete[] tempRgb[i];

}

delete[] tempRgb;

}

void bmpImage::extension2(){

//создаем массив ргб цветов

RGB \*\*tempRgb = new RGB\*[infoHeader->infHeight];

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

tempRgb[i] = new RGB[infoHeader->infWidth];

}

//заполняем массив значением -1, как флаг того, что цвет пикселей в этом квадрате 3х3 не был изменен

for(unsigned int y = 0; y < infoHeader->infHeight; y++){

for(unsigned int x = 0; x < infoHeader->infWidth; x++){

tempRgb[y][x].Red = -1;

tempRgb[y][x].Green = -1;

tempRgb[y][x].Blue = -1;

}

}

//средние цвета

int middleR = 0, middleG = 0, middleB = 0;

//заполняем массив средними значениями по квадрату, если какой-то стороны не хватает для обработки 3х3, заполняем сколько можем

unsigned int maxHeight = infoHeader->infHeight - ((infoHeader->infHeight - 1) %2);

unsigned int maxWidth = infoHeader->infWidth - ((infoHeader->infWidth - 1) % 2);

for(unsigned int y = 0; y < maxHeight - maxHeight % 2; y++){

for(unsigned int x = 0; x < maxWidth - maxHeight % 2; x++){

if(tempRgb[y][x].Red == -1){

middleR = (rgb[y][x].Red+rgb[y][x+1].Red

+rgb[y+1][x].Red+rgb[y+1][x+1].Red)/4;

middleG = (rgb[y][x].Green+rgb[y][x+1].Green

+rgb[y+1][x].Green+rgb[y+1][x+1].Green)/4;

middleB = (rgb[y][x].Blue+rgb[y][x+1].Blue

+rgb[y+1][x].Blue+rgb[y+1][x+1].Blue)/4;

tempRgb[y][x].Red = middleR; tempRgb[y][x+1].Red = middleR;

tempRgb[y+1][x].Red = middleR; tempRgb[y+1][x+1].Red = middleR;

tempRgb[y][x].Green = middleG; tempRgb[y][x+1].Green = middleG;

tempRgb[y+1][x].Green = middleG; tempRgb[y+1][x+1].Green = middleG;

tempRgb[y][x].Blue = middleB; tempRgb[y][x+1].Blue = middleB;

tempRgb[y+1][x].Blue = middleB; tempRgb[y+1][x+1].Blue = middleB;

}

}

}

//переносим массив в основное изображение

for(unsigned int y = 0; y < maxHeight - maxHeight%2; y++){

for(unsigned int x = 0; x < maxWidth- maxHeight%2; x++){

rgb[y][x].Red = Truncate(tempRgb[y][x].Red);

rgb[y][x].Green = Truncate(tempRgb[y][x].Green);

rgb[y][x].Blue = Truncate(tempRgb[y][x].Blue);

}

}

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

delete[] tempRgb[i];

}

delete[] tempRgb;

}

void bmpImage::lens(){

int x = infoHeader->infWidth/2;

int y = infoHeader->infHeight/2;

int LensDiam = 0;

if(infoHeader->infHeight > infoHeader->infWidth){

LensDiam = infoHeader->infWidth/3.14;

}else{

LensDiam = infoHeader->infHeight/3.14;

}

int k = LensDiam/10; //коэф искажения

RGB \*\*tempRgb = new RGB\*[infoHeader->infHeight];

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

tempRgb[i] = new RGB[infoHeader->infWidth];

}

for(unsigned int y = 0; y < infoHeader->infHeight; y++){

for(unsigned int x = 0; x < infoHeader->infWidth; x++){

tempRgb[y][x].Red = rgb[y][x].Red;

tempRgb[y][x].Green = rgb[y][x].Green;

tempRgb[y][x].Blue = rgb[y][x].Blue;

}

}

for(int i = -LensDiam; i <= LensDiam; i++)

{

int len = sqrt((float)(LensDiam\*LensDiam - i\*i));

for(int j = -len; j <= len; j++)

{

int dy = -k\*sin(3.14\*i/(2\*LensDiam));

int dx = -k\*sin(3.14\*j/(2\*abs(len)));

if(abs(len) == abs(j)){

dx = -k\*sin(3.14\*i);

dy = -k\*sin(3.14\*j);

}

if(abs(i + dy) < LensDiam && abs(j + dx) < abs(len))

{

rgb[y + i + dy][x + j + dx].Red = Truncate(tempRgb[y+i][x+j].Red);

rgb[y + i + dy][x + j + dx].Green = Truncate(tempRgb[y+i][x+j].Green);

rgb[y + i + dy][x + j + dx].Blue = Truncate(tempRgb[y+i][x+j].Blue);

}

}

}

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

delete[] tempRgb[i];

}

delete[] tempRgb;

}

void bmpImage::sphere(){

int halfWidth = infoHeader->infWidth/2;

int halfHeight = infoHeader->infHeight/2;

RGB \*\*tempRgb = new RGB\*[infoHeader->infHeight];

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

tempRgb[i] = new RGB[infoHeader->infWidth];

}

for(unsigned int y = 0; y < infoHeader->infHeight; y++){

for(unsigned int x = 0; x < infoHeader->infWidth; x++){

tempRgb[y][x].Red = 0;

tempRgb[y][x].Green = 0;

tempRgb[y][x].Blue = 0;

}

}

int radius = 0;

if(infoHeader->infHeight > infoHeader->infWidth){

radius = infoHeader->infWidth/3.1415;

}else{

radius = infoHeader->infHeight/3.1415;

}

for(unsigned int y = 0; y < infoHeader->infHeight; y++){

for(unsigned int x = 0; x < infoHeader->infWidth; x++){

int distX = x-halfWidth;

int distY = y-halfHeight;

float distanceSource = hypot(distX,distY);

float distanceNew = 2\*radius\*asin(distanceSource/radius)/2;

float factor = distanceNew/distanceSource;

if(distanceSource<=radius){

unsigned int newX = abs(distX \* factor + halfWidth);

unsigned int newY = abs(distY \* factor + halfHeight);

if(distanceNew == 0){

newX = distX;

newY = distY;

}

if(newX >= infoHeader->infWidth) newX = infoHeader->infWidth - 1;

if(newY >= infoHeader->infHeight) newY = infoHeader->infHeight - 1;

tempRgb[y][x].Red = rgb[newY][newX].Red;

tempRgb[y][x].Green = rgb[newY][newX].Green;

tempRgb[y][x].Blue = rgb[newY][newX].Blue;

}

}

}

//переносим массив в основное изображение

for(unsigned int y = 0; y < infoHeader->infHeight; y++){

for(unsigned int x = 0; x < infoHeader->infWidth; x++){

rgb[y][x].Red = Truncate(tempRgb[y][x].Red);

rgb[y][x].Green = Truncate(tempRgb[y][x].Green);

rgb[y][x].Blue = Truncate(tempRgb[y][x].Blue);

}

}

for (unsigned int i = 0; i < infoHeader->infHeight; i++) {

delete[] tempRgb[i];

}

delete[] tempRgb;

}

## Файл bmpimage.h

#ifndef BMPIMAGE\_H

#define BMPIMAGE\_H

#include <fstream>

#include <QDebug>

using namespace std;

class bmpImage

{

// CIEXYZTRIPLE stuff

typedef int FXPT2DOT30;

typedef struct {

FXPT2DOT30 ciexyzX;

FXPT2DOT30 ciexyzY;

FXPT2DOT30 ciexyzZ;

} CIEXYZ;

typedef struct {

CIEXYZ ciexyzRed;

CIEXYZ ciexyzGreen;

CIEXYZ ciexyzBlue;

} CIEXYZTRIPLE;

// bitmap file header

typedef struct

{

unsigned short fType;

unsigned int fSize;

unsigned short fReserved1;

unsigned short fReserved2;

unsigned int fOffBits;

}FILEHEADER;

FILEHEADER \*fileHeader;

// bitmap info header

typedef struct

{

unsigned int infSize;

unsigned int infWidth;

unsigned int infHeight;

unsigned short infPlanes;

unsigned short infBitCount;

unsigned int infCompression;

unsigned int infSizeImage;

unsigned int infXPelsPerMeter;

unsigned int infYPelsPerMeter;

unsigned int infClrUsed;

unsigned int infClrImportant;

unsigned int infRedMask;

unsigned int infGreenMask;

unsigned int infBlueMask;

unsigned int infAlphaMask;

unsigned int infCSType;

CIEXYZTRIPLE infEndpoints;

unsigned int infGammaRed;

unsigned int infGammaGreen;

unsigned int infGammaBlue;

unsigned int infIntent;

unsigned int infProfileData;

unsigned int infProfileSize;

unsigned int infReserved;

}INFOHEADER;

INFOHEADER \*infoHeader;

typedef struct

{

short Blue;

short Green;

short Red;

char Reserved;

}RGB;

RGB \*\*rgb;

public:

bmpImage();

bmpImage(bmpImage &image);

~bmpImage();

bool loadImage(ifstream &stream);

bool saveImage(ofstream &stream);

//геттеры

int getWidth();

int getHeight();

int getRed(int x, int y);

int getGreen(int x, int y);

int getBlue(int x, int y);

int getType();

//сеттеры

void setPixel(unsigned int x,unsigned int y,int red, int green, int blue);

//прочее

void editContrast(float n);

void editBright(int n);

void extension();

void extension2();

void lens();

void sphere();

};

#endif // BMPIMAGE\_H

## Файл drawalgorithms.cpp

#include "mainwindow.h"

#include <QtDebug>

int sign(float a){

if(a > 0.) return 1;

if(a < 0.) return -1;

return 0;

}

//пикселы окружности по часовой стрелке для алгортима Минчера

void pixelCircleM(bmpImage \*image, int centerX, int centerY, int x, int y, int r, int g, int b){

image->setPixel(centerX+x, centerY+y, r, g, b);

image->setPixel(centerX+y, centerY+x, r, g, b);

image->setPixel(centerX+y, centerY-x, r, g, b);

image->setPixel(centerX+x, centerY-y, r, g, b);

image->setPixel(centerX-x, centerY-y, r, g, b);

image->setPixel(centerX-y, centerY-x, r, g, b);

image->setPixel(centerX-y, centerY+x, r, g, b);

image->setPixel(centerX-x, centerY+y, r, g, b);

}

void BresenhamAlgorithm(bmpImage \*image, int x1, int y1, int x2, int y2, int r, int g, int b){

//Генерация отрезка алгоритмом Брезенхейма

int x = x1;

int y = y1;

int Dx = abs(x2-x1);

int Dy = abs(y2-y1);

int s1 = sign(x2-x1);

int s2 = sign(y2-y1);

int swaped = false;

if(Dy>Dx){

int t =Dx;

Dx = Dy;

Dy = t;

swaped = true;

} else{

swaped = false;

}

int e = Dy\*2 - Dx;

image->setPixel(x, y, r, g, b);

for(int i = 0; i<Dx;++i){

while(e>=0){

if(swaped){

x+=s1;

} else {

y+=s2;

}

e-=2\*Dx;

}

if(!swaped){

x+=s1;

} else {

y+=s2;

}

e+=2\*Dy;

image->setPixel(x, y, r, g, b);

}

}

void MichinerCircle (bmpImage \*image, int centerX, int centerY, int radius, int r, int g, int b){

//Генерация окружности алгоритмом Минчера

int x = 0;

int y = radius;

int d = 3 - 2 \* radius;

while (x < y) {

pixelCircleM(image, centerX, centerY, x, y, r, g, b);

if (d < 0){

d = d + 4\*x + 6;

}

else {

d = d + 4\*(x-y) + 10;

--y;

}

++x;

}

if (x == y) pixelCircleM(image, centerX, centerY, x, y, r, g, b);

}

## Файл drawbmpimage.cpp

#include "drawbmpimage.h"

drawBMPImage::drawBMPImage(bmpImage \*image,QWidget \*parent) : QWidget(parent)

{

this->image = image;

}

drawBMPImage::drawBMPImage(QWidget \*parent) : QWidget(parent)

{

this->image = NULL;

}

drawBMPImage::~*drawBMPImage*()

{

}

void drawBMPImage::*paintEvent*(QPaintEvent \*)

{

if(image!=NULL && image->getType()!= 1){

QPainter painter(this);

QImage img(image->getWidth(),image->getHeight(),QImage::Format\_RGB32);

for(int x = image->getWidth()-1; ; x--){

for(int y = image->getHeight()-1, j = 0; j < image->getHeight(); y--,j++){

img.setPixel(x,j,qRgb(image->getRed(x,y),image->getGreen(x,y),image->getBlue(x,y)));

}

if(x == 0) break;

}

painter.drawImage(0,0,img);

}

}

## Файл drawbmpimage.h

#ifndef DRAWBMPIMAGE\_H

#define DRAWBMPIMAGE\_H

#include <QWidget>

#include <QPainter>

#include "bmpimage.h"

class drawBMPImage : public QWidget

{

Q\_OBJECT

bmpImage \*image;

public:

explicit drawBMPImage(QWidget \*parent = 0);

drawBMPImage(bmpImage \*image,QWidget \*parent = 0);

~*drawBMPImage*();

signals:

protected:

void *paintEvent*(QPaintEvent \*);

public slots:

};

#endif // DRAWBMPIMAGE\_H

## Файл editcontrastbrightwidget.cpp

#include "editContrastBrightWidget.h"

editContrastBrightWidget::editContrastBrightWidget(bmpImage \*image, QWidget \*parent) : QWidget(parent)

{

this->image = image;

//параметры окна

setWindowFlags(Qt::Window);

setWindowTitle("Редактирование контраста и яркости");

setMinimumSize(400,50);

QPalette p(palette());

p.setColor(QPalette::Background, Qt::white);

setPalette(p);

QHBoxLayout \*verticalLayout = new QHBoxLayout(this);

//параметры контрастности

contrastSlider= new QSlider(Qt::Horizontal,this);

contrastSlider->setRange(-100,100);

contrastSlider->setPageStep(1);

contrastSlider->setValue(0);

contrastSlider->setToolTip("Контраст");

contrastSlider->setTickInterval(1);

contrastLabel = new QLabel("Контраст:",this);

QObject::connect(contrastSlider, SIGNAL(valueChanged(int)), contrastLabel, SLOT(setNum(int)));

contrastEnableButton= new QPushButton("Применить",this);

verticalLayout->addWidget(contrastLabel);

verticalLayout->addWidget(contrastSlider);

verticalLayout->addWidget(contrastEnableButton);

//параметры яркости

brightLabel = new QLabel("Яркость:",this);

brightSlider= new QSlider(Qt::Horizontal,this);

brightSlider->setRange(-255,255);

brightSlider->setPageStep(1);

brightSlider->setValue(0);

brightSlider->setToolTip("Яркость");

brightSlider->setTickInterval(1);

QObject::connect(brightSlider, SIGNAL(valueChanged(int)), brightLabel, SLOT(setNum(int)));

brightEnableButton= new QPushButton("Применить",this);

verticalLayout->addWidget(brightLabel);

verticalLayout->addWidget(brightSlider);

verticalLayout->addWidget(brightEnableButton);

QObject::connect(brightEnableButton, SIGNAL(clicked()), this, SLOT(editBright()));

QObject::connect(brightEnableButton, SIGNAL(clicked()), parent, SLOT(reDrawImage()));

QObject::connect(contrastEnableButton, SIGNAL(clicked()), this, SLOT(editContrast()));

QObject::connect(contrastEnableButton, SIGNAL(clicked()), parent, SLOT(reDrawImage()));

}

editContrastBrightWidget::~*editContrastBrightWidget*()

{

delete contrastLabel;

delete brightLabel;

delete contrastSlider;

delete brightSlider;

delete contrastEnableButton;

delete brightEnableButton;

}

void editContrastBrightWidget::editBright(){

if(image!=NULL && image->getType()!=-1){

image->editBright(brightSlider->value());

brightSlider->setValue(0);

}else{

qDebug() << "Sthm with your image!";

}

}

void editContrastBrightWidget::editContrast(){

if(image!=NULL && image->getType()!= 1){

image->editContrast((1.00 + contrastSlider->value()\*0.01));

contrastSlider->setValue(0);

}else{

qDebug() << "Sthm with your image!";

}

}

## Файл editcontrastbrightwidget.h

#ifndef EDITCONTRASTANDBRIGHTWIDGET\_H

#define EDITCONTRASTANDBRIGHTWIDGET\_H

#include <QWidget>

#include <QLabel>

#include <QSlider>

#include <QPushButton>

#include <QLayout>

#include "bmpimage.h"

class editContrastBrightWidget : public QWidget

{

Q\_OBJECT

QLabel \*contrastLabel;

QLabel \*brightLabel;

QSlider \*contrastSlider;

QSlider \*brightSlider;

QPushButton \*contrastEnableButton;

QPushButton \*brightEnableButton;

bmpImage \*image;

public:

explicit editContrastBrightWidget(bmpImage \*image, QWidget \*parent = 0);

~*editContrastBrightWidget*();

signals:

public slots:

void editContrast();

void editBright();

};

#endif // EDITCONTRASTANDBRIGHTWIDGET\_H

## Файл main.cpp

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

## Файл mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

QPalette p(palette());

p.setColor(QPalette::Background, Qt::white);

setPalette(p);

ui->setupUi(this);

image = new bmpImage();

drawer = new drawBMPImage();

ui->gridImageLayout->addWidget(drawer);

sucsess = 0;

mousePosX = 0;

mousePosY = 0;

drawMode = 0;

}

MainWindow::~MainWindow()

{

}

void MainWindow::on\_openFileAction\_triggered()

{

QString str = QFileDialog::getOpenFileName(this, QString::fromUtf8("Открыть bmp файл"),

QDir::currentPath(), "Images (\*.bmp \*.dib \*.rle)");

std::string name = str.toStdString();

bmpFile.open(name.c\_str(),std::ifstream::binary);

if (!bmpFile.is\_open())

qDebug() << "Cannot to open file!\n";

if(bmpFile.is\_open() && image!=NULL && image->getType()!= 1){

delete image;

}

if(bmpFile.is\_open()){

image = new bmpImage();

sucsess = image->loadImage(bmpFile);

}

if(sucsess){

reDrawImage();

//под размер экрана

QDesktopWidget \*mApp = QApplication::desktop();

if(mApp->width() > image->getWidth()

&& mApp->height() > image->getHeight()){

setMinimumSize(image->getWidth(),image->getHeight());

}

else if(mApp->width() < image->getWidth()

&& mApp->height() > image->getHeight()){

setMinimumSize(mApp->width(),image->getHeight());

}

else if(mApp->height() < image->getHeight()

&& mApp->width() > image->getWidth()){

setMinimumSize(image->getWidth(),mApp->height());

}

else if(mApp->height() < image->getHeight()

&& mApp->width() < image->getWidth()){

setMinimumSize(mApp->width(),mApp->height());

}

}else{

QMessageBox::critical(this,QObject::tr("Ошибка"),tr("Ошибка при загрузке изображения"));

qDebug() << "Any problem with image!";

}

bmpFile.close();

}

void MainWindow::on\_saveAsAction\_triggered()

{

if(sucsess){

QString fileName = QFileDialog::getSaveFileName(this,

QString::fromUtf8("Сохранить файл"),

"untitled.bmp",

"Images (\*.bmp \*.dib \*.rle)");

std::string name = fileName.toStdString();

saveFile.open(name.c\_str(),std::ofstream::binary);

image->saveImage(saveFile);

saveFile.close();

}else{

QMessageBox::warning(this,QObject::tr("Внимание!"),tr("Сначала загрузите изображение!"));

qDebug() << "Any problem with image!";

}

}

void MainWindow::on\_contrastAction\_triggered()

{

if(sucsess){

editContrastBrightWidget \*editContrastBright = new editContrastBrightWidget(image, this);

editContrastBright->show();

}else{

QMessageBox::warning(this,QObject::tr("Внимание!"),tr("Сначала загрузите изображение!"));

qDebug() << "Any problem with image!";

}

}

void MainWindow::on\_extensionAction\_triggered()

{

if(sucsess){

image->extension();

reDrawImage();

}else{

QMessageBox::warning(this,QObject::tr("Внимание!"),tr("Сначала загрузите изображение!"));

qDebug() << "Any problem with image!";

}

}

void MainWindow::on\_extension2Action\_triggered()

{

if(sucsess){

image->extension2();

reDrawImage();

}else{

QMessageBox::warning(this,QObject::tr("Внимание!"),tr("Сначала загрузите изображение!"));

qDebug() << "Any problem with image!";

}

}

void MainWindow::on\_lensAction\_triggered()

{

if(sucsess){

image->lens();

reDrawImage();

}else{

QMessageBox::warning(this,QObject::tr("Внимание!"),tr("Сначала загрузите изображение!"));

qDebug() << "Any problem with image!";

}

}

void MainWindow::on\_sphereAction\_triggered()

{

if(sucsess){

image->sphere();

reDrawImage();

}else{

QMessageBox::warning(this,QObject::tr("Внимание!"),tr("Сначала загрузите изображение!"));

qDebug() << "Any problem with image!";

}

}

void MainWindow::on\_lineAction\_triggered()

{

if(sucsess){

color = QColorDialog::getColor(Qt::black, this, "Выберите цвет");

drawPointX.clear();

drawPointY.clear();

drawMode = 2;

}else{

QMessageBox::warning(this,QObject::tr("Внимание!"),tr("Сначала загрузите изображение!"));

qDebug() << "Any problem with image!";

}

}

void MainWindow::on\_circleAction\_triggered()

{

if(sucsess){

color = QColorDialog::getColor(Qt::black, this, "Выберите цвет");

drawPointX.clear();

drawPointY.clear();

drawMode = 3;

}else{

QMessageBox::warning(this,QObject::tr("Внимание!"),tr("Сначала загрузите изображение!"));

qDebug() << "Any problem with image!";

}

}

void MainWindow::on\_dotAction\_triggered()

{

if(sucsess){

color = QColorDialog::getColor(Qt::black, this, "Выберите цвет");

drawPointX.clear();

drawPointY.clear();

drawMode = 1;

}else{

QMessageBox::warning(this,QObject::tr("Внимание!"),tr("Сначала загрузите изображение!"));

qDebug() << "Any problem with image!";

}

}

void MainWindow::reDrawImage()

{

ui->gridImageLayout->removeWidget(drawer);

delete drawer;

drawer = new drawBMPImage(image);

ui->gridImageLayout->addWidget(drawer);

}

void MainWindow::mousePressEvent(QMouseEvent \*event){

if(drawMode){

if(event->button() == Qt::LeftButton){

if(event->x() < image->getWidth() && event->y() < image->getHeight()){

mousePosX = event->x();

mousePosY = event->y()-20;

int R = color.red();

int G = color.green();

int B = color.blue();

if(drawMode == 1){

image->setPixel(mousePosX,mousePosY,R,G,B);

reDrawImage();

}

if(drawMode == 2){

if(drawPointX.length()<=2){

drawPointX.append(mousePosX);

drawPointY.append(mousePosY);

}

if(drawPointX.length()>=2){

BresenhamAlgorithm(image,drawPointX[0],drawPointY[0],drawPointX[1],drawPointY[1],R,G,B);

drawPointX.clear();

drawPointY.clear();

reDrawImage();

}

}

if(drawMode == 3){

if(drawPointX.length()<=2){

drawPointX.append(mousePosX);

drawPointY.append(mousePosY);

}

if(drawPointX.length()>=2){

int radius = sqrt(abs(drawPointY[0]-drawPointY[1])\*abs(drawPointY[0]-drawPointY[1])

+abs(drawPointX[0]-drawPointX[1])\*abs(drawPointX[0]-drawPointX[1]));

MichinerCircle(image,drawPointX[0],drawPointY[0],radius,R,G,B);

drawPointX.clear();

drawPointY.clear();

reDrawImage();

}

}

}

}

if(event->button() == Qt::RightButton){

drawMode = 0;

reDrawImage();

}

}

}

## Файл mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QFileDialog>

#include <QGraphicsScene>

#include <QGraphicsItemGroup>

#include <QMessageBox>

#include <QDesktopWidget>

#include <QColorDialog>

#include <QMouseEvent>

#include <fstream>

#include "bmpimage.h"

#include "drawbmpimage.h"

#include "editcontrastbrightwidget.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~MainWindow();

private slots:

void on\_openFileAction\_triggered();

void on\_contrastAction\_triggered();

void on\_extensionAction\_triggered();

void on\_lensAction\_triggered();

void on\_sphereAction\_triggered();

void on\_lineAction\_triggered();

void on\_circleAction\_triggered();

void on\_dotAction\_triggered();

void on\_saveAsAction\_triggered();

void on\_extension2Action\_triggered();

void mousePressEvent(QMouseEvent \*event);

public slots:

void reDrawImage();

private:

Ui::MainWindow \*ui;

bmpImage \*image;

std::ifstream bmpFile;

std::ofstream saveFile;

drawBMPImage \*drawer;

int mousePosX;

int mousePosY;

bool sucsess;

int drawMode; // 0 - нет, 1 - точка, 2 - линия, 3 - окружность

QVector<int> drawPointX;

QVector<int> drawPointY;

QColor color;

};

void MichinerCircle (bmpImage \*image, int centerX, int centerY, int radius, int r, int g, int b);

void BresenhamAlgorithm(bmpImage \*image, int x1, int y1, int x2, int y2, int r, int g, int b);

#endif // MAINWINDOW\_H

## Файл mainwindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>380</width>

<height>150</height>

</rect>

</property>

<property name="sizePolicy">

<sizepolicy hsizetype="Minimum" vsizetype="Minimum">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="contextMenuPolicy">

<enum>Qt::NoContextMenu</enum>

</property>

<property name="windowTitle">

<string>BMP editor Кузенков Василий АВТ-615</string>

</property>

<widget class="QWidget" name="centralWidget">

<property name="sizePolicy">

<sizepolicy hsizetype="Preferred" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="toolTipDuration">

<number>0</number>

</property>

<layout class="QGridLayout" name="gridLayout">

<property name="leftMargin">

<number>0</number>

</property>

<property name="topMargin">

<number>0</number>

</property>

<property name="rightMargin">

<number>0</number>

</property>

<property name="bottomMargin">

<number>0</number>

</property>

<property name="spacing">

<number>0</number>

</property>

<item row="0" column="0">

<layout class="QGridLayout" name="gridImageLayout">

<property name="horizontalSpacing">

<number>5</number>

</property>

<property name="verticalSpacing">

<number>0</number>

</property>

</layout>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menuBar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>380</width>

<height>21</height>

</rect>

</property>

<property name="defaultUp">

<bool>false</bool>

</property>

<property name="nativeMenuBar">

<bool>true</bool>

</property>

<widget class="QMenu" name="fileMenu">

<property name="title">

<string>Файл</string>

</property>

<addaction name="openFileAction"/>

<addaction name="separator"/>

<addaction name="saveAsAction"/>

<addaction name="separator"/>

</widget>

<widget class="QMenu" name="editMenu">

<property name="title">

<string>Редактирование</string>

</property>

<addaction name="contrastAction"/>

<addaction name="separator"/>

<addaction name="extensionAction"/>

<addaction name="extension2Action"/>

<addaction name="separator"/>

<addaction name="lensAction"/>

<addaction name="sphereAction"/>

</widget>

<widget class="QMenu" name="drawMenu">

<property name="title">

<string>Рисование</string>

</property>

<addaction name="lineAction"/>

<addaction name="circleAction"/>

<addaction name="dotAction"/>

</widget>

<addaction name="fileMenu"/>

<addaction name="editMenu"/>

<addaction name="drawMenu"/>

</widget>

<action name="openFileAction">

<property name="text">

<string>Открыть bmp...</string>

</property>

<property name="shortcut">

<string>Ctrl+O</string>

</property>

</action>

<action name="saveAsAction">

<property name="text">

<string>Сохранить как...</string>

</property>

<property name="shortcut">

<string>Ctrl+Shift+S</string>

</property>

</action>

<action name="contrastAction">

<property name="text">

<string>Контраст\Яркость</string>

</property>

</action>

<action name="brightAction">

<property name="text">

<string>Яркость</string>

</property>

</action>

<action name="extensionAction">

<property name="text">

<string>Разрешение (3х3)</string>

</property>

</action>

<action name="lensAction">

<property name="text">

<string>Линза</string>

</property>

</action>

<action name="sphereAction">

<property name="text">

<string>Сфера (fisheye)</string>

</property>

</action>

<action name="lineAction">

<property name="text">

<string>Линии</string>

</property>

<property name="toolTip">

<string>Рисование линии алгоритмом Брезенхема. Выбираем две точки, и между ними сразу проводится линия.</string>

</property>

</action>

<action name="circleAction">

<property name="text">

<string>Окружности</string>

</property>

<property name="toolTip">

<string>Рисование окружности алгоритмом Минчера. Выбираем две точки, первая послужит центром, а вторая основанием окружности.</string>

</property>

</action>

<action name="dotAction">

<property name="text">

<string>Точка</string>

</property>

</action>

<action name="extension2Action">

<property name="text">

<string>Разрешение (2х2)</string>

</property>

</action>

<action name="action\_WAV">

<property name="text">

<string>Открыть окно редактора WAV</string>

</property>

<property name="shortcut">

<string>Ctrl+Shift+W</string>

</property>

</action>

</widget>

<layoutdefault spacing="6" margin="11"/>

<resources/>

<connections/>

</ui>