Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

ПРОГРАММА «ФОТОРЕДАКТОР»

БГУИР КП 1-40 02 01 203 ПЗ

Студент: гр. 250502 Болашенко В.С.

Руководитель: Богдан Е. В.

МИНСК 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc152904501)

[1 ОБЗОР ЛИТЕРАТУРЫ 6](#_Toc152904502)

[1.1 Обзор методов и алгоритмов решения поставленной задачи 6](#_Toc152904503)

[1.2 Обзор аналогов приложения 6](#_Toc152904504)

[2 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 8](#_Toc152904505)

[2.1 Структура входных и выходных данных 8](#_Toc152904506)

[2.2 Разработка диаграммы классов 8](#_Toc152904507)

[2.3 Описание классов 8](#_Toc152904508)

# ВВЕДЕНИЕ

С++ — компилируемый язык программирования общего назначения. Он поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование (ООП), обобщенное программирование.

Язык программирования широко используется для разработки программного обеспечения: создание разнообразных прикладных программ, разработка операционных систем, драйверов устройств, а также видеоигр и многое другое.

Также язык программирование С++ позволяет подключать фреймворки, которые расширяют возможности языка: позволяют создавать оконные приложения, игры, обрабатывать фото- и видеоматериалов и др. Примерами крупных фреймворков являются Qt (для разработки ПО) и OpenCV (обработка изображений, компьютерное зрение).

ООП — подход к программированию как к моделированию информационных объектов, решающий на новом уровне основную задачу структурного программирования: структурирование информации с точки зрения управляемости, что существенно улучшает управляемость самим процессом моделирования, что, в свою очередь, особенно важно при реализации крупных проектов.

Основные принципы структурирования в случае ООП связаны с различными аспектами базового понимания предметной задачи, которое требуется для оптимального управления соответствующей моделью:

* абстракция для выделения в моделируемом предмете важного для решения конкретной задачи по предмету, в конечном счёте — контекстное понимание предмета, формализуемое в виде класса;
* инкапсуляция для быстрой и безопасной организации собственно иерархической управляемости: чтобы было достаточно простой команды «что делать», без одновременного уточнения как именно делать, так как это уже другой уровень управления;
* наследование для быстрой и безопасной организации родственных понятий: чтобы было достаточно на каждом иерархическом шаге учитывать только изменения, не дублируя всё остальное, учтённое на предыдущих шагах;
* полиморфизм для определения точки, в которой единое управление лучше распараллелить или наоборот — собрать воедино.

# ОБЗОР ЛИТЕРАТУРЫ

## 1.1 Обзор методов и алгоритмов решения поставленной задачи

Для обработки изображения был использован фреймворк OpenCV. Он позволяет представить изображение в виде матрицы пикселей. Благодаря этому, обращаясь к пикселям и меняя их значение, мы можем редактировать исходное изображение. Также OpenCV предоставляет ряд функций и методов, которые позволяют редактировать всё изображение сразу, например, изменить его яркость и контрастность, инвертировать изображение и другие возможности.

В ходе создания приложения было реализован контейнер двунаправленного кольца MyRing<T>. В нём реализованы методы для добавления и удаления элементов, методы смещения указателя на следующий или предыдущий элемент, а также перегружен оператор [] для получения доступа к произвольному элементу кольца, начиная счет от «головы» кольца. Данное кольцо используется для хранения фильтров и работы с ними.

Для реализации пользовательского интерфейса был использован фреймворк Qt. Программа Qt Creator позволяет легко и быстро создать сам интерфейс приложения, а благодаря системе сигналов и слотов, которая представленная данным фреймворком, данный интерфейс соединяется с программным кодом. Также, благодаря реализованному в Qt классу потока QThread, был реализован производный от него класс MyThread, в котором происходит обработка изображения.

## 1.2 Обзор аналогов приложения

**1.2.1 Adobe Photoshop Lightroom**

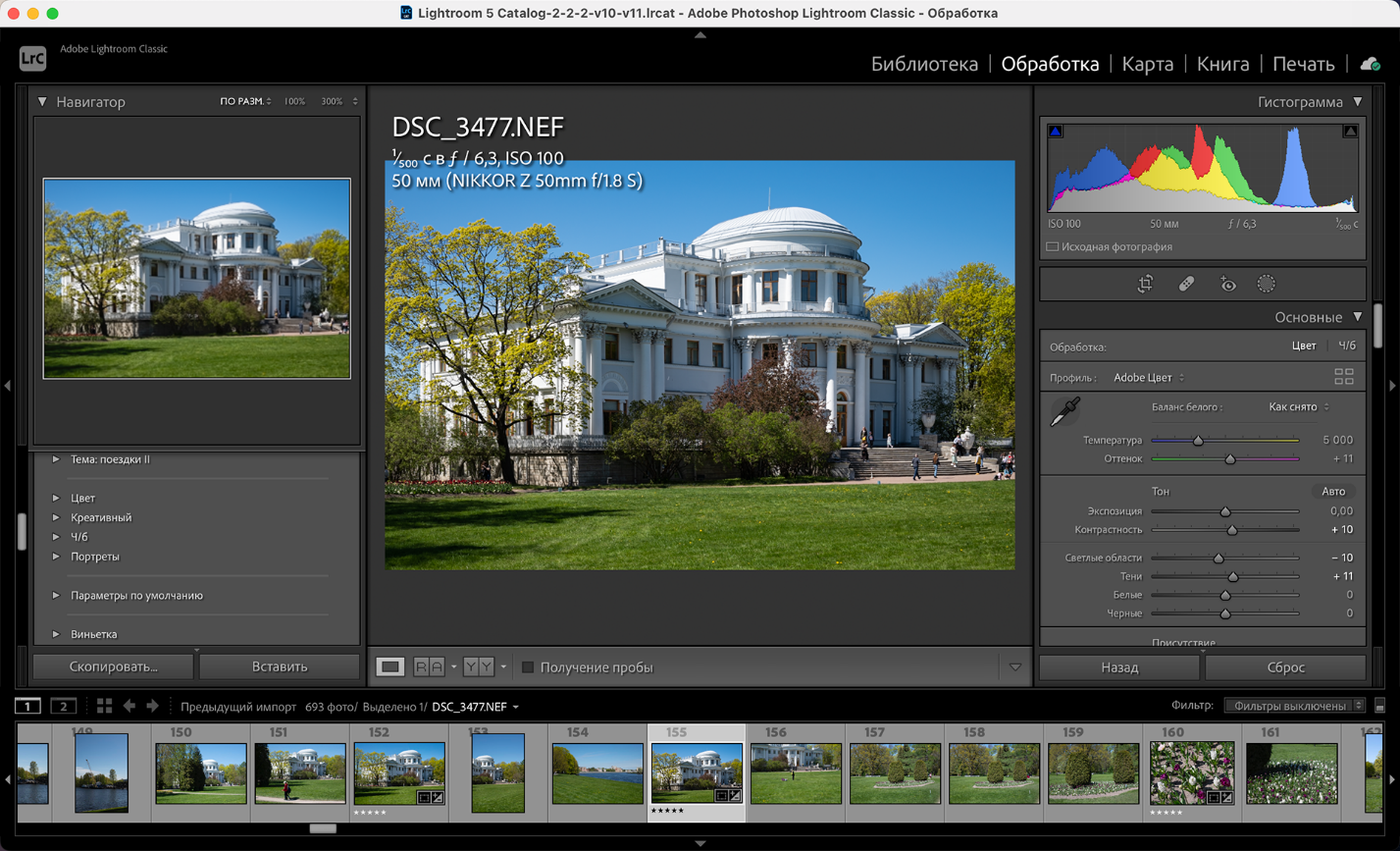
Adobe Photoshop Lightroom (рисунок 1.1) — графический редактор компании Adobe для работы с цифровыми фотографиями. Может использоваться для «проявки» «цифровых негативов» (форматы данных DNG, Raw), ретуши фотоснимков и организации их каталога. Особенностью программы является «недеструктивное редактирование», при котором исходный файл изображения остаётся неизменным, а все операции редактирования изображения осуществляются над автоматически сгенерированными из мастер-файла рабочими файлами — «версиями».

**1.2.2 Microsoft Photos**

Microsoft Photos (Фотографии) – программа для просмотра изображений, видео-редактор, сортировщик фотографий, редактор растровой графики и приложение для раздачи фотографий. «Фотографии» предоставляет базовые возможности растрового графического редактора, такие как:

Рисунок 1.1 – Adobe Photoshop Lightroom

коррекция экспозиции или цвета, изменение размера, обрезка, удаление «эффекта красных глаз», удаление «пятен», удаление «шумов» (рисунок 1.2).

****

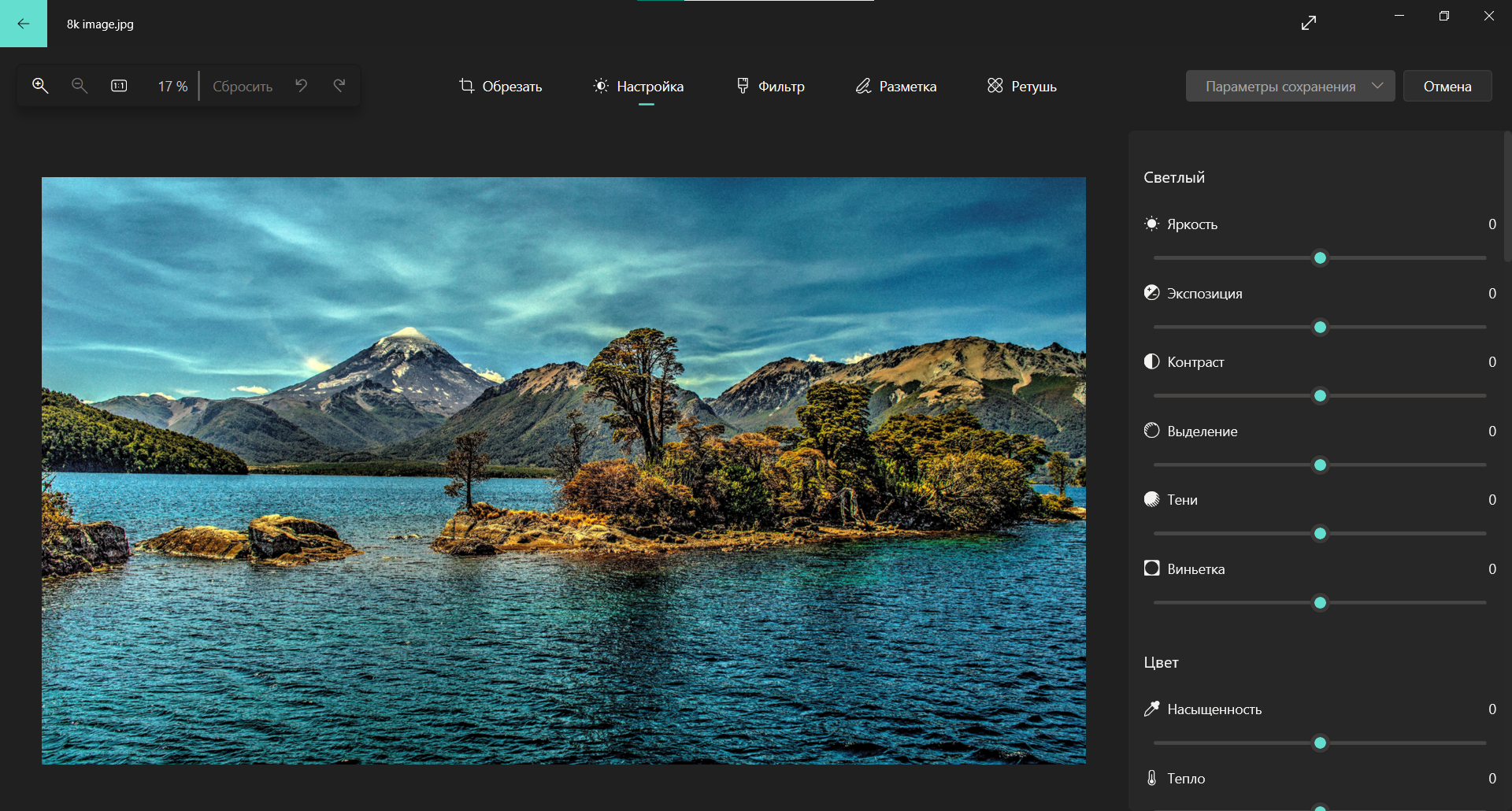


Рисунок 1.2 – Microsoft Photos (Фотографии)

# ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описываются входные и выходные данные программы, диаграмма классов, а также приводится описание используемых классов и их методов.

## 2.1 Структура входных и выходных данных

Таблица 2.1 – файл с пользовательскими фильтрами filters\_inform.json

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название | Яркость | Контрастность | Насыщенность | Четкость | Температура |
| Filter1 | 13 | 30 | 36 | 0 | -26 |

Таблица 2.2 – файл ранее открытых изображений recently\_opened.json

|  |
| --- |
| Путь к файлу |
| D:\\University\\cs\\sem3\\cursach\\test.png |

## 2.2 Разработка диаграммы классов

Диаграмма классов для данного курсового проекта представлена в **Приложении А**.

## 2.3 Описание классов

**2.3.1 Классы операций над изображением**

Класс Operation является абстрактным классом. Он описывает операцию над изображением.

Поля класса:

* int value – значение для изменения характеристики изображения.
* cv::Mat image – изображения в виде матрицы, предоставляемое OpenCV.

Метода:

* virtual cv::Mat exec() = 0 – чисто виртуальная функция, которая после переопределения в производных класса будет производить обработку изображения image и возвращать обработанное изображение.

От класса Operation наследуются классы, каждый из которых будет обрабатывать свою конкретную характеристику изображения.

Класс Oper\_brightness предназначен для изменения яркости изображения. Является производным от класса Operation.

Поля класса наследуются от класса Operation.

Метода:

* Oper\_brightness(int, cv::Mat) – конструктор класса, который устанавливает значение характеристики и изображение для обработки.
* virtual cv::Mat exec() override – виртуальная функция, которая является переопределением метода из базового класса. Она изменяется яркость изображения и возвращает измененное изображение.

Класс Oper\_contrast предназначен для изменения контрастности изображения. Является производным от класса Operation.

Поля класса наследуются от класса Operation.

Метода:

* Oper\_contrast(int, cv::Mat) – конструктор класса, который устанавливает значение характеристики и изображение для обработки.
* virtual cv::Mat exec() override – виртуальная функция, которая является переопределением метода из базового класса. Она изменяется контрастность изображения и возвращает измененное изображение.

Класс Oper\_saturation предназначен для изменения насыщенности изображения. Является производным от класса Operation.

Поля класса наследуются от класса Operation.

Метода:

* Oper\_saturation(int, cv::Mat) – конструктор класса, который устанавливает значение характеристики и изображение для обработки.
* virtual cv::Mat exec() override – виртуальная функция, которая является переопределением метода из базового класса. Она изменяется насыщенность изображения и возвращает измененное изображение.

Класс Oper\_clarity предназначен для изменения четкости изображения. Является производным от класса Operation.

Поля класса наследуются от класса Operation.

Метода:

* Oper\_clarity(int, cv::Mat) – конструктор класса, который устанавливает значение характеристики и изображение для обработки.
* virtual cv::Mat exec() override – виртуальная функция, которая является переопределением метода из базового класса. Она изменяется четкость изображения и возвращает измененное изображение.

Класс Oper\_temperature предназначен для изменения температуры изображения. Является производным от класса Operation.

Поля класса наследуются от класса Operation.

Метода:

* Oper\_temperature(int, cv::Mat) – конструктор класса, который устанавливает значение характеристики и изображение для обработки.
* virtual cv::Mat exec() override – виртуальная функция, которая является переопределением метода из базового класса. Она изменяется температуру изображения и возвращает измененное изображение.

**2.3.2 Класс потока обработки изображения**

Класс потока MyThread предназначен для обработки изображения параллельно с основным потоком программы.

Поля класса:

* std::queue<Operation \*> queue – очередь указателей на базовый класс Operation, которая хранит порядок выполнения операций над изображения.

Методы класса:

* void push(Operation \*) – метод для помещения операции над изображением в очередь.
* virtual void run() override – переопределение виртуальной функции, вызывается при запуске потока, выполняет операцию из вершины очереди и отправляет это изображение вместе с сигналом signalGUI(cv::Mat).

Сигналы класса:

* signalGUI(cv::Mat) – сигнал, который сообщает о выполнении операции над изображение и передает обработанное изображение.

Слоты класса:

* void terminateThread() – функция, которая завершает поток.

Благодаря классу MyThread пользователь может в реальном времени видеть изменения изображение, при изменении положения ползунка характеристики.

**2.3.3 Классы фильтров изображения**

Класс Filter является абстрактным классом. На его основе с помощью наследования реализованы классы Inverse, Original, Gray и CustomFilter.

Поля класса:

* cv::Mat image – обработанное изображение.
* std::string name – строка для хранения названия фильтра.
* int brightness – значение яркости изображение.
* int contrast – значение контрастности изображения.
* int saturation – значение насыщенности изображения.
* int clarity – значение четкости изображения.
* int temperature – значение температуры изображения.

Методы класса:

* cv::Mat apply() – метод для возвращения обработанного изображения.
* std::string get\_filter\_name() – метод для получения названия фильтра.
* int get\_brightness() – метод для получения значения яркости фильтра.
* int get\_contrast() – метод для получения значения контрастности фильтра.
* int get\_saturation() – метод для получения значения насыщенности фильтра.
* int get\_clarity() – метод для получения значения четкости фильтра.
* int get\_temperature() – метод для получения значения температуры фильтра.
* virtual ~Filter() = 0 – чисто виртуальный деструктор, который делает класс абстрактным.

Класс Inverse, который предназначен для инверсии изображения.

Поля класса наследуются от базового класса Filter.

Методы:

* Inverse(cv::Mat) – конструктор, которому передается изображение для обработки. Он инвертирует это изображение и сохраняет итоговый результат, задается название фильтра.

Класс Original, который предназначен для получения первоначального изображения.

Поля класса наследуются от базового класса Filter.

Методы:

* Original(cv::Mat) – конструктор, которому передается изображение для обработки, сохраняет это изображение, задает название фильтра.

Класс Gray, который меняет цветовую гамму изображения на серую.

Поля класса наследуются от базового класса Filter.

Методы:

* Gray(cv::Mat) – конструктор, которому передается изображение для обработки. Он конвертирует это изображение в оттенках серого и сохраняет итоговый результат, задается название фильтра.

Класс CustomFilter, который предназначен для применения к изображению характеристик, заданных пользователем.

Поля класса наследуются от базового класса Filter.

Методы:

* CustomFilter(std::string, cv::Mat, int, int, int, int, int) – конструктор, которому передается название фильтра, изображение для обработки, значения яркости, контрастности, насыщенности, четкости и температуры. Он обрабатывает изображение по заданным параметрам и сохраняет итоговый результат, также сохраняется название фильтра и переданные значения характеристик.

**2.3.4 Класс кнопки с автоматическим изменением размера иконки**

Класс IconautosizePushButton предназначен для автоматического изменения размера иконки кнопки во время изменения размера самой кнопки. Является производным классом от класса QPushButton

Поля класса:

* Поля, наследуемые от класса QPushButton.
* QString image\_path – строка, в которой хранится путь до изображения иконки.

Методы класса:

* void set\_image\_path(QString &) – задает строку, в которой хранится путь до изображения иконки.
* QString &get\_image\_path() – возвращает строку, в которой хранится путь до изображения иконки.

Слоты класса:

* void resizeEvent(QResizeEvent \*) override – переопределение функции изменения размера кнопки, которая меняет и размер кнопки, и размер иконки.

**2.3.5 Классы оконных интерфейсов**

Класс MainWindow является основным оконным интерфейсов, в котором происходит открытие, обработка и экспорт изображения. Наследуется от класса QMainWindow и класса Ui\_MainWindow, который сгенерирован автоматические и в котором объявлены все объекты, которые помещены на окно с помощью Qt Creator.

Поля класса:

* Поля, наследуемые от базовых классов.
* MyThread \*mythread – поток для обработки изображения.
* PROCESSES current\_process – хранит текущий процесс над изображением.
* MyRing<Filter \*>filters – двунаправленное кольцо, которое хранит существующие фильтры.
* int filter\_number – хранит номер выбранного фильтра.
* QTranslator qtlangtransl – перевод приложения на другие языки.
* QGraphicsScene \*graphicsScene – графическая сцена для отображения графических предметов на графическом виде.
* QGraphicsPixmapItem \*pixmap – графический предмет для отображения изображения типа QPixmap на графической сцене.
* struct image\_info – структура, которая хранит информацию об изображении и имеет следующие поля:
* QPixmap \*start\_image – начальное изображение.
* QPixmap \*image\_in\_proc – обработанное изображение.
* int brightness – значение яркости обработанного изображения.
* int contrast – значение контрастности обработанного изображения.
* int saturation – значение насыщенности обработанного изображения.
* int clarity – значение четкости обработанного изображения.
* int temperature – значение температуры обработанного изображения.

Методы класса:

* void set\_connections() – производит основные соединения сигналов со слотами.
* MainWindow(QWidget \*) – конструктор главного окна, в котором инициализируются переменные, выделяется память под указатели, скрываются ненужные в начальный момент объекты окна и запускается поток обработки изображения.
* void set\_curr\_proc(PROCESSES) – задает, какой процесс сейчас происходит.
* PROCESSES get\_curr\_proc() – возвращает, какой процесс сейчас происходит.
* void prepare\_image() – подготавливает стартовое изображение к дальнейшим операциям.
* void set\_filters() – инициализирует кольцо фильтров, считывая значения из файла.
* void save\_filters() – сохраняет пользовательские фильтры в файл.
* cv::Mat get\_filtered\_image(int) – получает номер фильтра, возвращает изображение с примененным к нему фильтром.
* std::string get\_filter\_name(int) – возвращает имя фильтра, номер которого передан.
* void set\_filter\_number(int) – задает значение переменной filter\_number.
* void next\_prev\_filter(int) – перемещает «голову» кольца на следующий элемент, если передаваемое число положительное, или на предыдущий элемент, если передаваемое число отрицательное.
* void resizeEvent(QEvent \*) override – переопределение обработчика событий, которое, если зафиксировано событие изменения языка приложение, запускает изменение переводимых надписей на объектах окна.

Слоты класса:

* void change\_image(cv::Mat) – изменяет отображаемое изображение, на передаваемое.
* void set\_rec\_opened\_butts() – задает кнопки ранее открытых изображений.
* void start\_proc(QString &) – открывает изображение по переданному пути, либо, если строка пустая, открывает файловое диалоговое окно, где пользователь выбирает изображение для обработки. Скрывает объекты для открытия изображения и отображает объекты для работы с изображением. Сохраняет путь до открытого изображения, если этот не был сохранен ранее.
* void main\_proc() – основной процесс работы с изображением. Получает значение с ползунка и меняет определенную характеристику изображение в зависимости от текущего процесса.
* void set\_slider\_limits() – задает границы ползунка, а также его начальное значение.
* void end\_main\_proc() – сохраняет значение характеристики изображение, с которой только что работали.
* void rotate\_left() – поворачивает изображение на 90 градусов против часовой стрелки.
* void rotate\_right() – поворачивает изображение на 90 градусов по часовой стрелки.
* void save\_image() – сохраняет изображение в выбранное пользователем место.
* void set\_new\_image() – скрывает объекты для работы с изображением и отображает объекты для открытия изображения, обнуляет процесс и характеристики.
* void set\_filters\_buttons() – задает иконку кнопки и отображение названия фильтра в зависимости от расположения фильтров в кольце.
* void set\_deleteF\_enabled(std::string) – задает активность кнопки в зависимости от имени выбранного фильтра.
* void back\_from\_filters() – возвращает пользователя от выбора фильтра, к изменению характеристик изображения.
* void apply\_filter() – применяет фильтр к изображению.
* void delete\_filter() – удаляет выбранный фильтр.
* void add\_filter() – добавляет фильтр в коллекцию, значения фильтра берутся из значений изображения, настроенных пользователем на данный момент.
* void show\_pressed\_button() – визуально помечает, какая кнопка сейчас нажата.
* void change\_language(const char\*) – изменяет язык приложения.

Класс FilterName\_window, с помощью которого задается имя для добавляемого фильтра. Он наследуется от класса QWidget и класса Ui\_Form, который сгенерирован автоматически на основании созданного в Qt Creator окна.

Поля класса:

* Поля, наследуемые от базовых классов.
* MyRing<Filters\*> \*filters – указатель на двунаправленное кольцо фильтров.
* std\_string filter\_name – имя фильтра, введенное пользователем.

Методы класса:

* FilterName\_window(QWidget \*) – конструктор класса, в котором создаются объекты окна, а также происходят соединения сигналов со слотами.
* void set\_filters(MyRing<Filter\*>\*) – инициализирует указатель на кольцо фильров.
* bool is\_name\_in\_filters(std::string) – ищет указанное имя среди всех фильтров.
* std::string get\_filter\_name() – возвращает filter\_name.
* void changeEvent(QEvent \*) override – переопределение обработчика событий, который ри сменен языка приложение запускает изменение надписей объектов окна.

Слоты класса:

* void safe\_filter\_name() – проверяет имя, вводимое пользователем, и, если имя не пустая строка и такого имени нет среди всех фильтров, то сохраняет его в filter\_name.

Сигналы класса:

* void filter\_name\_got() – сообщает о том, что имя фильтра получено.

**2.3.6 Другие классы**

Класс графического вида ViewWithoutWheel, который игнорирует события колёсика мыши. Наследуется от класса QGraphicsView.

Слоты класса:

* virtual void wheelEvent(QWheelEvent \*) override – переопределение обработчика событий колёсика мыши, которые игнорирует колёсико мыши.

Перечисление PROCESSES, которое содержит процессы, которые могут происходить с изображением: изменение яркости, контрастности, насыщенности, четкости, температуры изображение; поворот изображения на 90 градусов; применение фильтров; отсутствие процесса, которое как-то меняет изображение.

Константы перечисления:

* BRIGHTNESS
* CONTRAST
* SATURATUIN
* CLARITY
* TEMPERATURE
* ROTATION
* FILTER
* NON

Перечисление FILTER, которое содержит все возможные типы фильтров: инверсия, оригинальное изображение, оттенки серого и пользовательский фильтр.

Константы перечисления:

* INVERSE
* ORIGINAL
* GRAY
* CUSTOM