МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка PNG файла

Студент гр. 2300 _____ Войнов А.Н.

Преподаватель _____ Гаврилов А.В.

Санкт-Петербург

2023

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Войнов А.Н.

Группа 2300

Тема работы: Обработка PNG файла

Программа должна иметь CLI или GUI.

Программа должна реализовывать весь следующий функционал по обработке png-файла

Общие сведения

- Формат картинки PNG (рекомендуем использовать библиотеку libpng)
- без сжатия
- файл всегда соответствует формату PNG
- обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями.
- все поля стандартных PNG заголовков в выходном файле должны иметь те же значения что и во входном (разумеется кроме тех, которые должны быть изменены).

Программа должна реализовывать следующий функционал по обработке PNG-файла

- (1) Рисование отрезка. Отрезок определяется:
  - координатами начала
  - координатами конца
  - цветом

- толщиной

- (2) Инвертировать цвета в заданной окружности. Окружность определяется
  - либо координатами левого верхнего и правого нижнего угла квадрата, в который она вписана, либо координатами ее центра и радиусом
- (3) Обрезка изображения. Требуется обрезать изображение по заданной области. Область определяется:
  - Координатами левого верхнего угла
  - Координатами правого нижнего угла

Содержание пояснительной записки:

Аннотация, содержание, введение, ход работы, примеры работы программы, заключение, список использованных источников.

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 22.03.2023

Дата сдачи реферата: 22.05.2023

Дата защиты реферата: 24.05.2023

Студент _____ Войнов А.Н

Преподаватель _____ Гаврилов А.В.

АННОТАЦИЯ

Курсовая работа представляет собой программу на языке C++, написанную с использованием qt Creator для реализации графического интерфейса.

Программа считывает и сохраняет на диск png файл с форматом цвета rgba. Также пользователь может выбрать один из следующих вариантов обработки изображения: нарисовать отрезок, инвертировать цвета пикселей в окружности или обрезать картинку по заданным координатам. Затем пользователь может ещё раз обработать изображение или закрыть программу.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: Разработать программу с графическим интерфейсом, которая может считывать и сохранять png файл с форматом цвета rgba на диск, а также обрабатывать его согласно условию.

Для достижения цели требуется решить следующие задачи:

1) Изучить qt Creator.
2) Изучить libpng.
3) Реализовать считывание и запись изображение на диск.
4) Реализовать функции обработки изображения.
5) Создать графический интерфейс
6) Протестировать программу

ХОД РАБОТЫ

2.1 main.cpp

В файле main.cpp находится функция main(), которая запускает графический интерфейс, а также устанавливает иконку приложения.

2.2 Класс image

Для работы с изображением был реализован класс image. Внутри экземпляра этого класса хранятся данные об изображении, а также реализованы методы для работы с ним:

1) int readPngFile() – в качестве параметра получает адрес изображения в файловой системе, считывает и записывает его данные в соответствующие поля класса. В случае успешного считывания возвращает 0, иначе возвращает 1 и выводит сообщение об ошибке.

2) void writePngFile() - в качестве параметра получает адрес файла в файловой системе, сохраняет изображение, хранящееся в памяти внутри программы в этом файле.

3) void drawLine() – получает координаты концов отрезка, его толщину и цвет. Рисует отрезок на изображении, заменяя цвета исходных пикселей на переданный цвет с помощью метода pixelSetColor(). Для изображения отрезка используется формула y = ax + b, где a – тангенс угла наклона прямой. Если одна или обе координаты находятся вне области изображения, то отрисована будет только часть отрезка, которая лежат внутри этой области.

4) void invertCircle() – получает на вход координаты центра и радиус окружности. Затем проходится по всем пикселям лежащим внутри и на её границе и инвертирует её цвета с помощью метода invertPixelColor(). Эта функция принимает любые целые координаты. Если по итогам вычислений ни одна точка не лежит в области изображения, то ничего не происходит. Иначе инвертируются точки, лежащие в его области.

5) int invertCircle() – второй вариант инвертации пикселей внутри окружности. В качестве параметров получает координаты верхнего левого и нижнего правого углов квадрата, в которой будет вписана окружность. Проверяет являются ли координаты координатами углов квадрата. Если нет, то возвращает 1. Иначе переводит их в формат, используемый первым вариантом инвертации и вызывает его и возвращает 0. Эта функция принимает любые целые координаты. Если по итогам вычислений ни одна точка не лежит в области изображения, то ничего не происходит. Иначе инвертируются точки, лежащие в его области.

6) void cutImage() - в качестве параметров получает координаты верхнего левого и нижнего правого углов квадрата, по которому будет обрезано изображение. Если один из переданных параметров выходит за границу изображения, устанавливает его на границу (то есть если пользователь выделит область, один из концов которой лежит на изображении, а второй нет, обрежется та часть изображения, которая была выделена и программа не сообщит об ошибке. Однако если обе точки будут лежать вне области изображения, то пользователь увидит сообщение об ошибке). Затем создаёт новый массив для пикселей, копирует в него значения тех, которые требуется сохранить. Очищает старый массив и ставит на его место новый.

7) void pixelSetColor() – получает координаты пикселя и цвет, который он должен получить. Меняет цвет пикселя на нужный.

8) void invertPixelColor() – получает координаты пикселя и инвертирует его цвет, то есть для каждого из трех каналов цветов значение заменяется с n на 255-n.

9) QImage toQimg() – создаёт экземпляр класса QImage из изображения, хранящегося в памяти и возвращает его. Используется для отрисовки изображения.

10) void copy() – получает на вход экземпляр класса image, устанавливает значения всех полей текущего экзмпляра равными

соответствующим в переданного. Используется для отмены всех действий, произведённых с изображением.

11) doesExist(), getWidth(), getHeight, getPixels_arr() – гетеры соответствующих полей.

2.3 Графический интерфейс

В функции main() файла main.cpp создаётся экземпляр класса MainWindow, затем это окно отображается. Оно состоит из поля для отображения изображения, кнопок соответствующих функций обработки, кнопки для отмены всех действий с изображением, а также трёх меню: файл, инструменты, дополнительно. В «файл» можно открыть или сохранить изображения. В инструментах можно вызвать функции для обработки изображения с альтернативным способом ввода данных, а также отменить все сделанные изменения. В «дополнительно» можно открыть справку и ознакомиться с работой приложения.

У класса есть следующие поля:

1) ui – хранит данные о интерфейсе и его составляющих, обеспечивает
2) взаимодействие внутренней логики с интерфейсом.
3) img – объект класса image, хранит данные о изображении, с которым работает пользователь.
4) oldImg – хранит объект класса image в исходном виде, чтобы дать
5) возможность пользователю отменить все изменения.
6) xStart и yStart – координаты точки, на которой была нажата мышь.
7) tools tool – экземпляр перечисления tools, хранит данные о том, какая кнопка была нажата последней (то есть какая функция сейчас активна).
8) thikness, color – хранят данные о изображении, которые используются в функциях по его обработке.
9) path – хранит путь к изображению в файловой системе.

Также в файле с этим классом реализован enum tools.

Для класса реализованы следующие методы:

1) void on_open_triggered() – при нажатии «открыть» вызывает окно выбора пути к файлу, затем передаёт в метод для считывания файла поля img. Также для вызова можно использовать комбинацию клавиш ctrl+o.

2) void on_save_triggered(), void on_saveAs_triggered() – методы для сохранения изображения, хранящегося в поле img. Первое сохраняет изображение в том же файле, из которого его открывали. Второе позволяет выбрать файл. Для вызова можно использовать комбинации клавиш ctrl+s и ctrl+shift+s соответственно.

3) void on_invertCircle_clicked(), void on_drawLine_clicked(), void on_invertCircleInSquare_clicked(), void on_cutImg_clicked() – методы, срабатывающие при нажатии кнопок на панели инструментов. Устанавливают нужные значения для перечисления tools.

4) void on_invertUsingCircleMenu_triggered(), void on_invertUsingSquareMenu_triggered(), void on_drawLineMenu_triggered(), void on_cutMenu_triggered() – методы вызываются при нажатии соответвующих пунктов в меню «инструменты» и открывают окна для ввода данных числами.

5) void on_info_triggered() – открывает справку.

6) void on_back_clicked() и void on_backMenu_triggered() – сбрасывают все изменения изображения до начального состояния.

7) void prepareForCut() – вспомогательной метод, принимает на вход координаты углов прямоугольника и устанавливает их следующим образом: первая – верхний левый угол, вторая – нижний правый. Это позволяет пользователю выделять любую область для вырезания изображения и не привязывает его к строгому порядку «левый верхний угол – правый нижний угол».

8) bool doesBothCoordsOutOfImg() – принимает координаты углов прямоугольника перед вырезанием. Если обе координаты находятся вне изображения возвращает true и программа оповещает пользователя о неверно введенной области, иначе false.

9) void showTmpPic() – отображает картинку, хранящуюся в поле img.

10) void mousePressEvent(QMouseEvent *event) – сохраняет координаты, в которых была нажата кнопка мыши в полях xStart и yStart.

11) void mouseReleaseEvent(QMouseEvent *event) – срабатывает, когда пользователь отпускает кнопку мыши. Проверят есть ли выбранное действие для работы с изображением. Если да, то вызывает соответствующий метод для img и отрисовывает изменения, иначе ничего не происходит.

Некоторые общие замечания по работе функций:

Пока изображение не загружено, нельзя выбрать функцию. При нажатии кнопок или полей меню будет высвечиваться сообщение об ошибке.

В работе графического интерфейса также участвуют некоторые классы, реализовывающие ввод данных для конкретных функций: DialogDrawLine, CutDialog, DrawLineMenuDialog, InvertUsingCircleDialog, InvertUsingSquareDialog и класс для справки: InfoDialog.

2.4 Справка

Для упрощения понимания работы с программой была создана справка (вызывать можно с помощью «f1»), в ней описываются методы ввода данных в программу, также говорится о формате изображения, которое может обрабатывать программа. Далее текст справки:

Это приложение редактирует изображения в формате png с цветом типа rgba.

Чтобы загрузить изображение выберите "файл -> открыть" или ctrl+o

Чтобы сохранить в том же файле, который вы открыли выберите "файл -> сохранить" или ctrl+s

Чтобы сохранить в другом файле "файл -> сохранить как" или ctr+shift+s

Путь для загрузки и сохранения может содержать в себе латинские буквы и цифры.

Приложение может обрабатывать изображения следующими способами:

1) нарисовать прямую линию заданной тощины и заданного цвета, для этого нажимте на иконку с линией на панеле инструментов слева, выберите толщину и цвет, а затем с помощью мышки нарисуйте линию. (если вам надо нарисовать отрезок с концами в четко заданных точках, то воспользуйтесь "инструменты ->нарисовать линию")

2) инверитровать цвета в окружности, для этого нажмите на иконку квадрата или круга, а затем нарисуйте с помощью мыши проведите радиус (начинается от центра и к точке на окружности) для иконки круга и или нарисуйте квадрат, в который будет вписана окружность, для иконки квадрата. Квадрат нарисовать довольно сложно:(поэтому вы можете задать координаты его углов через "инструменты->инвертировать->через квадрат", аналогично для круга можно задать его центр и радиус.

3) обрезать изображение по прямоугольнику. Для этого выберите иконку прямоугольника, а затем выделите мышкой область обрезания. Также можно задать её через "инструменты->вырезать". Задавая область вырезания через инструмменты убедитесь, что первая координата - верхний левый угол, а вторая нижний правый.

Все действия можно выполнять только после загрузки изображения. Также вы можете сбросить всё, что вы сделали с помощью иконки стрелочки или ctrl+z.

ЗАКЛЮЧЕНИЕ

В ходе работы была разработана программа для обработки изображения, она взаимодействует с пользователем с помощью графического интерфейса. Была изучена информация о работе с qt и libpng. Также готовая программа была протестирована. Примеры работы программы см в приложении А.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://doc.qt.io
2. https://stackoverflow.com
3. http://cppstudio.com
4. http://www.libpng.org/pub/png/libpng-1.2.5-manual.html

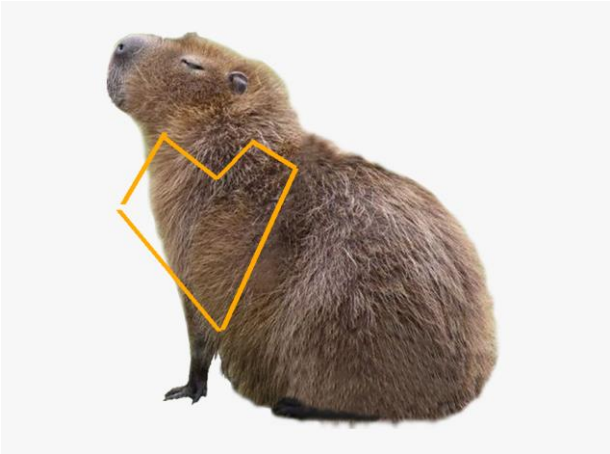
ПРИЛОЖЕНИЕ А
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

Пример 1:

Выберем инструмент для инвертации в окружности, заданной с помощью окружности

Пример 2:

Выберем инструмент для рисования прямых линий, установим значение толщины на 6, а цвет #ffaa00 и нарисуем несколько линий



Пример 3:

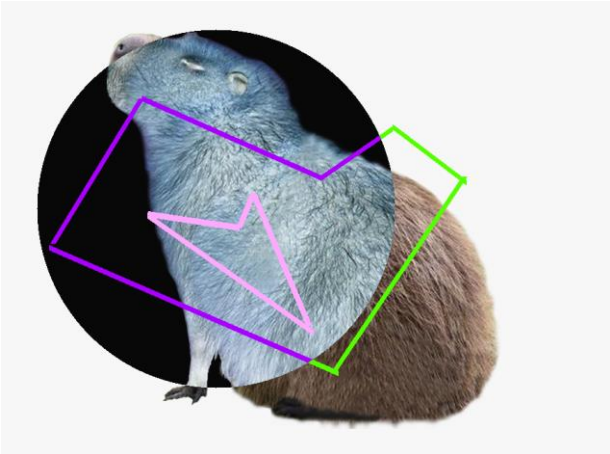
С помощью «инструменты -> инвертировать в окружности -> через квадрат» откроем окно для ввода углов квадрата, там введём две точки с координатами (200,200) и (300,300).

Пример 4:

Выберем инструмент для вырезания обрезания изображения и выделим небольшую область

Пример 5:

Применим к изображению несколько вариантов обработки подряд



ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД

Название файла main.cpp:

```
#include "mainwindow.h"
#include "image.h"


#include <QApplication>
#include <iostream>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    a.setWindowIcon(QIcon(":img/appIcon.png"));
    MainWindow w;


    w.show();
    return a.exec();
}
```

Название файла mainwindow.cpp:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"


MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    mousePressed = false;
    tool = tools::no;
    setWindowTitle("Графический редактор");

    ui->invertCircle->setIconSize(QSize(30,30));
    ui->invertCircle->setIcon(QIcon(":img/invertCircleIcon.png"));

    ui->invertCircleInSquare->setIconSize(QSize(30,30));
    ui->invertCircleInSquare-
>setIcon(QIcon(":img/invertSquareIcon.png"));

    ui->drawLine->setIconSize(QSize(30,30));
    ui->drawLine->setIcon(QIcon(":img/drawLineIcon.png"));

    ui->cutImg->setIconSize(QSize(30,30));
    ui->cutImg->setIcon(QIcon(":img/cutIcon.png"));

    ui->drawLine->setIconSize(QSize(30,30));
    ui->drawLine->setIcon(QIcon(":img/line.png"));

    ui->back->setIconSize(QSize(30,30));
    ui->back->setIcon(QIcon(":img/backArrow.png"));
```

```
    color = Qt::red;
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::showTmpPic(){
    ui->label->setFixedSize(img.getWidth(), img.getHeight());
    ui->label->setPixmap(QPixmap::fromImage(img.toQimg()));
}



void MainWindow::on_open_triggered(){
    QString fileName = QFileDialog::getOpenFileName(this, "Открыть файл",
"/home", "PNG_Images (*.png *.jpg)");

    path = fileName;
    if(!img.readPngFile(fileName)){
        oldImg.Copy(img);
        showTmpPic();
    }
}

void MainWindow::on_save_triggered(){
    if(!img.doesExists()){
        QMessageBox::critical(nullptr, "ошибка", "нечего сохранять");
        return;
    }
    img.writePngFile(path);
}

void MainWindow::on_saveAs_triggered(){
    if(!img.doesExists()){
        QMessageBox::critical(nullptr, "ошибка", "нечего сохранять");
        return;
    }
    QString fileName = QFileDialog::getSaveFileName(this, "Сохранить файл
как", "/result", "PNG_Images (*.png *.jpg)");

    if(fileName == ""){
        QMessageBox::critical(nullptr, "ошибка", "файл не существует");
        return;
    }
    img.writePngFile(fileName);
}


void MainWindow::mousePressEvent(QMouseEvent *event){
    xStart = event->x() - ui->label->x();
    yStart = event->y() - ui->label->y() - 25;
}

void MainWindow::prepareForCut(int &x1, int &y1, int &x2, int &y2){ //
довести до ума
```

```
    if(y1 < y2 && x1 < x2){
        return;
    }else if(x1 > x2 && y1 < y2){
        int left = x2;
        int right = x1;
        x1 = left;
        x2 = right;
    }else if(x1 > x2 && y1 > y2){
        int top = y2;
        int down = y1;
        int left = x2;
        int right = x1;
        x1 = left;
        x2 = right;
        y1 = top;
        y2 = down;
    }else if(x1 < x2 && y1 > y2){
        int top = y2;
        int down = y1;
        y1 = top;
        y2 = down;
    }
}

bool MainWindow::DoesBothCoordsOutOfImg(int x1, int y1, int x2, int y2){
    bool firstOut = false;
    bool secondOut = false;
    if(x1 < 0 || x1 > img.getWidth() || y1 < 0 || y1 > img.getHeight())
        firstOut = true;
    if(x2 < 0 || x2 > img.getWidth() || y2 < 0 || y2 > img.getHeight())
        secondOut = true;

    return  firstOut && secondOut;
}


void MainWindow::mouseReleaseEvent(QMouseEvent *event){
    int xFin = event->x() - ui->label->x();
    int yFin = event->y() - ui->label->y() - 25;

    if(tool == tools::invert){
                int r = sqrt(pow(xFin - xStart, 2) + pow(yFin - yStart,
2));
        if(r <= 0){
            QMessageBox::critical(nullptr, "ошибка", "Радиус должен быть
натуральным числом");
            return;
        }
        img.invertCircle(xStart, yStart, sqrt(pow(xFin - xStart, 2) +
pow(yFin - yStart, 2)));
    }else if(tool == tools::line){
        img.drawLine(xStart, yStart, xFin, yFin, color, thikness);
    }else if(tool == tools::invertSquare){
        prepareForCut(xStart, yStart, xFin, yFin);
        if(img.invertCircle(xStart, yStart, xFin, yFin))
            QMessageBox::critical(nullptr, "ошибка", "нарисуйте квадрат
или введите координаты углов через инструменты");
```

```
    }else if(tool == tools::cut){
        if(DoesBothCoordsOutOfImg(xStart, yStart, xFin, yFin)){
            QMessageBox::critical(nullptr, "ошибка", "обе точки находятся
вне изображения");
            return;
        }
        prepareForCut(xStart, yStart, xFin, yFin);
        img.cutImage(xStart, yStart, xFin, yFin);
    }

    if(tool != tools::no)
        showTmpPic();
}

void MainWindow::on_invertCircle_clicked(){
    if(!img.doesExists()){
        QMessageBox::critical(nullptr, "Ошибка", "Загрузите изображение с
помощью 'файл->открыть'");
        return;
    }
    tool = tools::invert;
}



void MainWindow::on_drawLine_clicked(){
    if(!img.doesExists()){
        QMessageBox::critical(nullptr, "Ошибка", "Загрузите изображение с
помощью 'файл->открыть'");
        return;
    }
    tool = tools::line;
    DialogDrawLine drawlinewindow(nullptr, thikness, color);
    drawlinewindow.exec();
}

void MainWindow::on_invertCircleInSquare_clicked(){
    if(!img.doesExists()){
        QMessageBox::critical(nullptr, "Ошибка", "Загрузите изображение с
помощью 'файл->открыть'");
        return;
    }
    tool = tools::invertSquare;

}

void MainWindow::on_cutImg_clicked(){
    if(!img.doesExists()){
        QMessageBox::critical(nullptr, "Ошибка", "Загрузите изображение с
помощью 'файл->открыть'");
        return;
    }
    tool = tools::cut;
}

void MainWindow::on_invertUsingCircleMenu_triggered(){// int &x, int &y,
int &r, int width, int height);
```

```
    if(!img.doesExists()){
        QMessageBox::critical(nullptr, "Ошибка", "Загрузите изображение с
помощью 'файл->открыть'");
        return;
    }
    int x, y, r;
    bool didIt = false;
    InvertUsingCircleDialog invertWindow(nullptr, x, y, r, didIt,
img.getWidth(), img.getHeight());
    invertWindow.exec();
    if(didIt){
        img.invertCircle(x, y, r);
        showTmpPic();
    }
}



void MainWindow::on_invertUsingSquareMenu_triggered(){
    if(!img.doesExists()){
        QMessageBox::critical(nullptr, "Ошибка", "Загрузите изображение с
помощью 'файл->открыть'");
        return;
    }

    int x1, y1, x2, y2;
    bool didIt = false;

    InvertUsingSquareDialog invertWindow(nullptr, x1, y1, x2, y2, didIt,
img.getWidth(), img.getHeight());
    invertWindow.exec();
    if(didIt){
        img.invertCircle(x1, y1, x2, y2);
        showTmpPic();
    }
}


void MainWindow::on_drawLineMenu_triggered()
{
    if(!img.doesExists()){
        QMessageBox::critical(nullptr, "Ошибка", "Загрузите изображение с
помощью 'файл->открыть'");
        return;
    }

    int x1, y1, x2, y2;
    bool didIt = false;

    DrawLineMenuDialog drawWindow(nullptr, x1, y1, x2, y2, thikness,
didIt, color, img.getWidth(), img.getHeight());
    drawWindow.exec();
    if(didIt){
        img.drawLine(x1, y1, x2, y2, color, thikness);
        showTmpPic();
    }
}
```

```
void MainWindow::on_cutMenu_triggered()
{
    if(!img.doesExists()){
        QMessageBox::critical(nullptr, "Ошибка", "Загрузите изображение с
помощью 'файл->открыть'");
        return;
    }

    int x1, y1, x2, y2;
    bool didIt = false;
    CutDialog cutWindow(nullptr, x1, y1, x2, y2, didIt, img.getWidth(),
img.getHeight());
    cutWindow.exec();

    if(DoesBothCoordsOutOfImg(x1, y1, x2, y2)){
        QMessageBox::critical(nullptr, "ошибка", "обе точки находятся вне
изображения");
        return;
    }
    if(didIt){
        prepareForCut(x1, y1, x2, y2);
        img.cutImage(x1, y1, x2, y2);
        showTmpPic();
    }
}

void MainWindow::on_info_triggered()
{
    InfoDialog infoWindow;
    infoWindow.exec();
}

void MainWindow::on_back_clicked()
{
    if(img.doesExists()){
        img.Copy(oldImg);
        showTmpPic();
    }
}

void MainWindow::on_backMenu_triggered()
{
    if(img.doesExists()){
        img.Copy(oldImg);
        showTmpPic();
    }
}
```

Название файла invertusingsquaredialog.cpp:

```
#include "invertusingsquaredialog.h"
#include "ui_invertusingsquaredialog.h"


InvertUsingSquareDialog::InvertUsingSquareDialog(QWidget *parent, int
&x1, int &y1, int &x2, int &y2, bool &did, int width, int height) :
```

```
    QDialog(parent),
    ui(new Ui::InvertUsingSquareDialog), x1In(x1), x2In(x2), y1In(y1),
y2In(y2), didIt(did)
{
    setWindowFlags(windowFlags() & ~Qt::WindowContextHelpButtonHint);

    ui->setupUi(this);
    setWindowTitle("Инвертировать");
    ui->info->setText(QString("Размер строки изображения %1 x
%2").arg(width).arg(height));
}

InvertUsingSquareDialog::~InvertUsingSquareDialog()
{
    delete ui;
}

void InvertUsingSquareDialog::on_buttonBox_accepted(){
    QString x1String = ui->x1Text->text();
    QString x2String = ui->x2Text->text();
    QString y1String = ui->y1Text->text();
    QString y2String = ui->y2Text->text();
    bool doesx1Int, doesx2Int, doesy1Int, doesy2Int;

    if(x1String == "" || x2String == "" || y1String == "" || y2String ==
""){
        QMessageBox::critical(nullptr, "Ошибка", "Заполните все поля");
        didIt = false;
        return;
    }

    x1In = x1String.toInt(&doesx1Int);
    x2In = x2String.toInt(&doesx2Int);
    y1In = y1String.toInt(&doesy1Int);
    y2In = y2String.toInt(&doesy2Int);

    if(!doesx1Int || !doesx2Int || !doesy1Int || !doesy2Int){
        QMessageBox::critical(nullptr, "Ошибка", "Координаты должны быть
целыми числами");
        didIt = false;
        return;
    }

    if(x2In - x1In != y2In - y1In){
        QMessageBox::critical(nullptr, "Ошибка", "Введенные координаты не
являются углами квадрата");
        didIt = false;
        return;
    }

    if(x2In < x1In || y2In < y1In){
        QMessageBox::critical(nullptr, "Ошибка", "Неверное расположение
углов");
        didIt = false;
        return;
    }
```

```
    didIt = true;
}
```

Название файла invertusingcircledialog.cpp:

```
#include "invertusingcircledialog.h"
#include "ui_invertusingcircledialog.h"


InvertUsingCircleDialog::InvertUsingCircleDialog(QWidget *parent, int &x,
int &y, int &r, bool &did, int width, int height) :
    QDialog(parent),
    ui(new Ui::InvertUsingCircleDialog), xIn(x), yIn(y), rIn(r),
didIt(did)
{
    setWindowFlags(windowFlags() & ~Qt::WindowContextHelpButtonHint);
    ui->setupUi(this);
    setWindowTitle("Инвертировать");

    ui->info->setText(QString("Размер строки изображения %1 x
%2").arg(width).arg(height));
}

InvertUsingCircleDialog::~InvertUsingCircleDialog()
{
    delete ui;
}

void InvertUsingCircleDialog::on_buttonBox_accepted(){
    bool doesXint, doesYint, doesRint;
    QString xString = ui->xCenter->text();
    QString yString = ui->yCenter->text();
    QString rString = ui->radius->text();
    if(rString == "" || xString == "" || yString == ""){
        QMessageBox::critical(nullptr, "Ошибка", "Заполните все поля");
        didIt = false;
        return;
    }

    xIn = xString.toInt(&doesXint);
    yIn = yString.toInt(&doesYint);
    rIn = rString.toInt(&doesRint);

    if(!doesRint || !doesXint || !doesYint || rIn <= 0){
        QMessageBox::critical(nullptr, "Ошибка", "Радиус должен быть
натуральным числом, а координаты целыми");
        didIt = false;
        return;
    }

    didIt = true;
}

Название файла infodialog.cpp:
#include "infodialog.h"
#include "ui_infodialog.h"
```

```
InfoDialog::InfoDialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::InfoDialog)
{
    ui->setupUi(this);
    setWindowFlags(windowFlags() & ~Qt::WindowContextHelpButtonHint);
    setWindowTitle("Справка");
}

InfoDialog::~InfoDialog()
{
    delete ui;
}
```

Название файла image.cpp:

```
#include "image.h"

image::image() {}

image::~image() {}

int image::readPngFile(QString path){
    std::string tmp_str = path.toStdString();
    const char *fileName = tmp_str.c_str();

    int headerLen = 8;
    char header[headerLen];

    FILE *file = fopen(fileName, "rb");
    if(!file){
        QMessageBox::critical(nullptr, "Ошбика", "Файл с таким именем не
найден или путь к файлу содержит недопустимые символы");
        return 1;
    }
    fread(header, sizeof(char), headerLen, file);

    if(png_sig_cmp((png_const_bytep)header, 0, headerLen)){
        QMessageBox::critical(nullptr, "Ошибка", "Файл не является png");
        return 1;
    }

    png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL, NULL,
NULL);
    if(!png_ptr){
        QMessageBox::critical(nullptr, "Ошибка", "ошибка в создании
png_ptr");
        return 1;
    }

    info_ptr = png_create_info_struct(png_ptr);
    if(!info_ptr){
        png_destroy_read_struct(&png_ptr, (png_infopp)NULL,
(png_infopp)NULL);
        fclose(file);
        QMessageBox::critical(nullptr, "Ошибка", "Ошибка в создании
info_ptr");
```

```
        return 1;
    }

    if(setjmp(png_jmpbuf(png_ptr))){
        png_destroy_read_struct(&png_ptr, &info_ptr, (png_infopp)NULL);
        fclose(file);
        QMessageBox::critical(nullptr, "Ошибка", "Не удалось получить
информацию об изображении");
        return 1;
    }

    png_init_io(png_ptr, file); // Initialize the default input/output
functions
    png_set_sig_bytes(png_ptr, headerLen);

    png_read_info(png_ptr, info_ptr);

    //проверяем что файл png RGBA
    if (png_get_color_type(png_ptr, info_ptr) !=
PNG_COLOR_TYPE_RGB_ALPHA){
        png_destroy_read_struct(&png_ptr, &info_ptr, NULL);//очистка
структуры
        fclose(file);

        QMessageBox::critical(nullptr,"Ошибка","У этого png файла тип
цвета не RGBA\n\nДанная программа работает только с файлами png  , с
типом цвета  RGBA!!!");
        return 1;
    }


    width = png_get_image_width(png_ptr, info_ptr);
    height = png_get_image_height(png_ptr, info_ptr);
    color_type = png_get_color_type(png_ptr, info_ptr);
    bit_depth = png_get_bit_depth(png_ptr, info_ptr);

    number_of_passes = png_set_interlace_handling(png_ptr);
    png_read_update_info(png_ptr, info_ptr);

    if (setjmp(png_jmpbuf(png_ptr))){
        png_destroy_read_struct(&png_ptr, &info_ptr, (png_infopp)NULL);
        fclose(file);
        QMessageBox::critical(nullptr, "Ошибка", "Не удалось считать
изображение");
        return 1;
    }

    pixels_arr = (png_bytep*) malloc(sizeof(png_bytep) * height);
    for(int y = 0; y < height; y++)
        pixels_arr[y] = (png_byte *) malloc(png_get_rowbytes(png_ptr,
info_ptr));

    png_read_image(png_ptr, pixels_arr);

    fclose(file);
    exist = true;
    return 0;
```

```
}

bool image::doesExists(){
    return exist;
}

void image::writePngFile(QString path){
    std::string tmp_str =   path.toStdString();
    const char *fileName = tmp_str.c_str();

    //открываем файл
    FILE *file = fopen(fileName, "wb");
    if(!file){
        QMessageBox::critical(nullptr, "Ошбика", "Файл с таким именем не
найден или путь к файлу содержит недопустимые символы");
        return;
    }

    //выделяем память
    png_ptr = png_create_write_struct(PNG_LIBPNG_VER_STRING, NULL, NULL,
NULL);
    if(!png_ptr){
        QMessageBox::critical(nullptr, "Ошибка", "ошибка в создании
png_ptr");
        return;
    }

    info_ptr = png_create_info_struct(png_ptr);
    if(!info_ptr){
        png_destroy_write_struct(&png_ptr, (png_infopp)NULL);
        fclose(file);
        QMessageBox::critical(nullptr, "Ошибка", "Ошибка в создании
info_ptr");
        return;
    }


    if (setjmp(png_jmpbuf(png_ptr))){
        png_destroy_write_struct(&png_ptr, &info_ptr);
        fclose(file);
        QMessageBox::critical(nullptr, "Ошибка", "Не удалось записать
данные о файле");
        return;
    }

    png_init_io(png_ptr, file);

    if (setjmp(png_jmpbuf(png_ptr))){
        png_destroy_write_struct(&png_ptr, &info_ptr);
        fclose(file);
        QMessageBox::critical(nullptr, "Ошибка", "Не удалось записать
header");
        return;
    }

    png_set_IHDR(png_ptr, info_ptr, width, height,
                 bit_depth, color_type, PNG_INTERLACE_NONE,
```

```
                PNG_COMPRESSION_TYPE_BASE, PNG_FILTER_TYPE_BASE);
    png_write_info(png_ptr, info_ptr);

    if (setjmp(png_jmpbuf(png_ptr))){
        png_destroy_write_struct(&png_ptr, &info_ptr);
        fclose(file);
        QMessageBox::critical(nullptr, "Ошибка", "Не удалось записать
изображение");
        //return;
    }

    png_write_image(png_ptr, pixels_arr);

    if (setjmp(png_jmpbuf(png_ptr))){
        png_destroy_write_struct(&png_ptr, &info_ptr);
        fclose(file);
        QMessageBox::critical(nullptr, "Ошибка", "Не удалось записать
конец файла");
    }

    png_write_end(png_ptr, NULL);

    for(int y = 0; y < height; y++){
        free(pixels_arr[y]);
    }

    fclose(file);
}

void image::pixelSetColor(int x, int y, QColor color){ // возвращает 0
если всё хорошо, 1 если введены неверные координаты пикселя

    if (x >= 0 && x < width && y >= 0 && y < height){ // проверяем
правильно ли введены данные о пикселе
        pixels_arr[y][x*4] = color.red();
        pixels_arr[y][x*4 + 1] = color.green();
        pixels_arr[y][x*4 + 2] = color.blue();
        pixels_arr[y][x*4 + 3] = color.alpha();
    }
}


QImage image::toQimg(){
    QImage imgForShow(width, height, QImage::Format_RGBA8888);
    for(int i = 0; i < width; i++){
        for(int j = 0; j < height; j++){
            imgForShow.setPixel(i, j, qRgba(pixels_arr[j][i * 4],
pixels_arr[j][i * 4 + 1], pixels_arr[j][i * 4 + 2], pixels_arr[j][i * 4 +
3]));
        }
    }

    return imgForShow;
}

void image::drawLine(int x1, int y1, int x2, int y2, QColor color, int
thikness){
```

```
    int topThik = trunc((float)(thikness - 1) / 2);
    int botThick = round((float)(thikness - 1) / 2);


    if(x1 == x2){ // вертикальная линия
        if(y2 > y1){
            for(int y = y1; y < y2; y++)
                for(int x = x1 - botThick; x <= x1 + topThik; x++)
                    pixelSetColor(x, y, color);

        }else{
            for(int y = y2; y < y1; y++)
                for(int x = x1 - botThick; x <= x1 + topThik; x++)
                    pixelSetColor(x, y , color);

        }
    }else{ // иначе находим tg
        float tg = ((float)(y2 - y1)) / (x2 - x1); // a = tg
        float b = y1 - tg * x1;
        if(abs(tg) <= 1){
            if(x2 > x1){//y = a*x
                for(int x = x1; x <= x2; x++){
                    int yBase = round(tg * x + b); // находим значение по
"графику"
                    for(int y = yBase - botThick; y <= yBase + topThik;
y++) // создаем толщину
                        pixelSetColor(x, y, color);
                }
            }else{
                for(int x = x2; x <= x1; x++){
                    int yBase = round(tg * x + b); // находим значение по
"графику"
                    for(int y = yBase - botThick; y <= yBase + topThik;
y++) // создаем толщину
                        pixelSetColor(x, y, color);
                }
            }
        }else{
            if(y2 > y1){
                for(int y = y1; y <= y2; y++){
                    int xBase = round((y - b) / tg);
                    for(int x = xBase - botThick; x <= xBase + topThik;
x++)
                        pixelSetColor(x, y, color);
                }
            }else{
                for(int y = y2; y <= y1; y++){
                    int xBase = round((y - b) / tg);
                    for(int x = xBase - botThick; x <= xBase + topThik;
x++)
                        pixelSetColor(x, y, color);
                }
            }
        }
    }
}
```

```
void image::invertPixelColor(int x, int y){

    if (x >= 0 && x < width && y >= 0 && y < height){ // проверяем
правильно ли введены данные о пикселе
        pixels_arr[y][x*4] = 255 - pixels_arr[y][x*4];
        pixels_arr[y][x*4 + 1] = 255 - pixels_arr[y][x*4 + 1];
        pixels_arr[y][x*4 + 2] = 255 - pixels_arr[y][x*4 + 2];
    }
}

void image::invertCircle(int x0, int y0, int r){

    for(int x = x0 - r; x <= x0 + r; x++){
        int y = y0;
        while(pow(x - x0, 2) + pow(y - y0, 2) <= pow(r, 2)){
            invertPixelColor(x, y);
            y++;
        }
        y = y0 - 1;
        while(pow(x - x0, 2) + pow(y - y0, 2) <= pow(r, 2)){
            invertPixelColor(x, y);
            y--;
        }
    }

}

int image::getWidth(){
    return width;
}

int image::getHeight(){
    return height;
}

png_bytep* image::getPixels_arr(){
    return pixels_arr;
}

int image::invertCircle(int x1, int y1, int x2, int y2){
    if(abs(x1 - x2) == abs(y1 - y2)){
        int x0 = ceil(float(x2 + x1) / 2);
        int y0 = ceil(float(y2 + y1) / 2);
        int r = floor(float(abs(x2 - x1)) / 2);
        invertCircle(x0, y0, r);
        return 0;
    }

    return 1;
}

void image::cutImage(int x1, int y1, int x2, int y2){

    if(x1 < 0)
        x1 = 0;
    if(y1 < 0)
        y1 = 0;
```

```
    if(x2 > width)
        x2 = width - 1;
    if(y2 > height)
        y2 = height - 1;

    png_bytep *tmp = (png_bytep*)malloc(sizeof(png_bytep) * (y2 - y1 +
1));
    for(int i = 0; i < y2 - y1 + 1; i++){
        tmp[i] = (png_byte*)malloc(sizeof(png_byte) * (x2 - x1 + 1) * 4);
    }



    for(int y = 0; y < y2 - y1 + 1; y++){
        for(int x = 0; x < (x2 - x1 + 1) * 4; x++){
            tmp[y][x] = pixels_arr[y + y1][x + x1 * 4];
        }
    }

    for(int y = 0; y < height; y++)
        free(pixels_arr[y]);
    free(pixels_arr);

    height = y2 - y1 + 1;
    width = x2 - x1 + 1;

    pixels_arr = tmp;
}

void image::Copy(image other){
    width = other.width;
    height = other.height;
    color_type = other.color_type;
    bit_depth = other.bit_depth;
    png_ptr = other.png_ptr;
    info_ptr = other.info_ptr;
    number_of_passes = other.number_of_passes;
    exist = other.exist;

    pixels_arr = (png_bytep *) malloc(sizeof(png_bytep) * height);
    for (int i = 0; i < height; i++)
        pixels_arr[i] = (png_byte *) malloc(sizeof(png_byte) * width *
4);// так как канолов 4


    for (int y = 0; y < height; y++) {
        for (int x = 0; x < (width*4); x++) {//4 - количество каналов (в
RGBA их 4)
            pixels_arr[y][x] = other.pixels_arr[y][x];
        }
    }
}
```

Название файла drawlinemenudialog.cpp:

```
#include "drawlinemenudialog.h"
#include "ui_drawlinemenudialog.h"
```

```
DrawLineMenuDialog::DrawLineMenuDialog(QWidget *parent, int &x1, int &y1,
int &x2, int &y2, int &thick, bool &did, QColor &color, int width, int
height) :
    QDialog(parent),
    ui(new Ui::DrawLineMenuDialog), x1In(x1), x2In(x2), y1In(y1),
y2In(y2), didIt(did), thickIn(thick), colorIn(color)
{
    setWindowFlags(windowFlags() & ~Qt::WindowContextHelpButtonHint);
    ui->setupUi(this);
    setWindowTitle("Нарисовать линию");

    ui->info->setText(QString("Размер строки изображения %1 x
%2").arg(width).arg(height));
}

DrawLineMenuDialog::~DrawLineMenuDialog()
{
    delete ui;
}

void DrawLineMenuDialog::on_buttonBox_accepted()
{
    QString x1String = ui->x1Text->text();
    QString x2String = ui->x2Text->text();
    QString y1String = ui->y1Text->text();
    QString y2String = ui->y2Text->text();
    QString thickString = ui->thickText->text();
    bool changeColor = ui->ChangeColor->isChecked();
    bool doesx1Int, doesx2Int, doesy1Int, doesy2Int, doesThickInt;

    if(x1String == "" || x2String == "" || y1String == "" || y2String ==
"" || thickString == ""){
        QMessageBox::critical(nullptr, "Ошибка", "Заполните все поля");
        didIt = false;
        return;
    }

    x1In = x1String.toInt(&doesx1Int);
    x2In = x2String.toInt(&doesx2Int);
    y1In = y1String.toInt(&doesy1Int);
    y2In = y2String.toInt(&doesy2Int);
    thickIn = thickString.toInt(&doesThickInt);

    if(!doesx1Int || !doesx2Int || !doesy1Int || !doesy2Int){
        QMessageBox::critical(nullptr, "Ошибка", "Координаты должны быть
целыми числами");
        didIt = false;
        return;
    }

    if(!doesThickInt || thickIn <= 0){
        QMessageBox::critical(nullptr, "Ошибка", "Толщина должна быть
натуральным числом");
        didIt = false;
        return;
```

```
    }

    if(changeColor){
        colorIn = QColorDialog::getColor(Qt::red,nullptr,"Выберите цвет
линии",QColorDialog::ShowAlphaChannel);
    }

    didIt = true;
}
```

Название файла dialogdrawline.cpp:

```
#include "dialogdrawline.h"
#include "ui_dialogdrawline.h"

DialogDrawLine::DialogDrawLine(QWidget *parent, int &thikness, QColor
&color) :
    QDialog(parent),
    ui(new Ui::DialogDrawLine), insideThikness(thikness), colorIn(color)
{
    ui->setupUi(this);
    setWindowFlags(windowFlags() & ~Qt::WindowContextHelpButtonHint);
    setWindowTitle("Настройки линии");
}

DialogDrawLine::~DialogDrawLine(){
    delete ui;
}



void DialogDrawLine::on_buttonBox_accepted(){
    bool doesInt;
    QString thiknessStr = ui->thikness->text();
    insideThikness = thiknessStr.toInt(&doesInt);
    if(!doesInt || insideThikness <= 0){
        QMessageBox::critical(nullptr, "Ошибка", "Введите натуральное
число");
        return;
    }
    colorIn = QColorDialog::getColor(Qt::red,nullptr,"Выберите цвет
линии",QColorDialog::ShowAlphaChannel);
}
```

Название файла cutdialog.cpp:

```
#include "cutdialog.h"
#include "ui_cutdialog.h"

CutDialog::CutDialog(QWidget *parent, int &x1, int &y1, int &x2, int &y2,
bool &did, int width, int height) :
    QDialog(parent),
    ui(new Ui::CutDialog),x1In(x1), x2In(x2), y1In(y1), y2In(y2),
didIt(did)
{
    setWindowFlags(windowFlags() & ~Qt::WindowContextHelpButtonHint);
```

```
    ui->setupUi(this);
    setWindowTitle("Вырезать");

    ui->info->setText(QString("Размер строки изображения %1 x
%2").arg(width).arg(height));

}

CutDialog::~CutDialog()
{
    delete ui;
}

void CutDialog::on_buttonBox_accepted()
{
    QString x1String = ui->x1Text->text();
    QString x2String = ui->x2Text->text();
    QString y1String = ui->y1Text->text();
    QString y2String = ui->y2Text->text();
    bool doesx1Int, doesx2Int, doesy1Int, doesy2Int;

    if(x1String == "" || x2String == "" || y1String == "" || y2String ==
""){
        QMessageBox::critical(nullptr, "Ошибка", "Заполните все поля");
        didIt = false;
        return;
    }

    x1In = x1String.toInt(&doesx1Int);
    x2In = x2String.toInt(&doesx2Int);
    y1In = y1String.toInt(&doesy1Int);
    y2In = y2String.toInt(&doesy2Int);

    if(!doesx1Int || !doesx2Int || !doesy1Int || !doesy2Int){
        QMessageBox::critical(nullptr, "Ошибка", "Координаты должны быть
целыми числами");
        didIt = false;
        return;
    }

    didIt = true;
}
```

Название файла mainwindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QFileDialog>
#include <QMouseEvent>
#include "image.h"
#include <QLabel>
#include <QColor>
#include "ui_mainwindow.h"
#include <QMessageBox>
#include <iostream>
```

```
#include <QPixmap>
#include "label.h"
#include <string>
#include <cstdlib>
#include <cmath>
#include "invertusingsquaredialog.h"
#include "cutdialog.h"
#include <windows.h>
#include <QSize>
#include <QIcon>
#include "dialogdrawline.h"
#include <QColorDialog>
#include "invertusingcircledialog.h"
#include "drawlinemenudialog.h"
#include "infodialog.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE


enum tools{
    no,
    invert,
    invertSquare,
    cut,
    line
};

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_open_triggered();
    void on_save_triggered();
    void on_saveAs_triggered();
    void mousePressEvent(QMouseEvent *event);
    void mouseReleaseEvent(QMouseEvent *event);
    void showTmpPic();

    void on_invertCircle_clicked();

    void on_drawLine_clicked();

    void on_invertCircleInSquare_clicked();

    void on_cutImg_clicked();

    void on_invertUsingCircleMenu_triggered();

    void on_invertUsingSquareMenu_triggered();
```

```
    void on_drawLineMenu_triggered();

    void on_cutMenu_triggered();

    void on_info_triggered();

    void on_back_clicked();

    void on_backMenu_triggered();

private:
    void prepareForCut(int &x1, int &y1, int &x2, int &y2);
    bool DoesBothCoordsOutOfImg(int x1, int y1, int x2, int y2);
    Ui::MainWindow *ui;
    image img;
    QString path;
    image oldImg;

    //x and y position of mouse;
    int xStart;
    int yStart;

    tools tool;
    bool mousePressed;
    int thikness;
    QColor color;
};



#endif // MAINWINDOW_H
```

Название файла invertusingsquaredialog.h:

```
#ifndef INVERTUSINGSQUAREDIALOG_H
#define INVERTUSINGSQUAREDIALOG_H

#include <QDialog>
#include <QString>
#include <QMessageBox>


namespace Ui {
class InvertUsingSquareDialog;
}

class InvertUsingSquareDialog : public QDialog
{
    Q_OBJECT

public:
    explicit InvertUsingSquareDialog(QWidget *parent, int &x1, int &y1,
int &x2, int &y2, bool &did, int width, int height);
    ~InvertUsingSquareDialog();
```

```
private slots:
    void on_buttonBox_accepted();

private:
    Ui::InvertUsingSquareDialog *ui;
    int &x1In;
    int &x2In;
    int &y1In;
    int &y2In;
    bool &didIt;
};

#endif // INVERTUSINGSQUAREDIALOG_H
```

Название файла invertusingcircledialog.h:

```
#ifndef INVERTUSINGCIRCLEDIALOG_H
#define INVERTUSINGCIRCLEDIALOG_H

#include <QDialog>
#include <QString>
#include <QMessageBox>

namespace Ui {
class InvertUsingCircleDialog;
}

class InvertUsingCircleDialog : public QDialog
{
    Q_OBJECT

public:
    explicit InvertUsingCircleDialog(QWidget *parent, int &x, int &y, int
&r, bool &did, int width, int height);
    ~InvertUsingCircleDialog();

private slots:
    void on_buttonBox_accepted();

private:
    Ui::InvertUsingCircleDialog *ui;
    int &xIn;
    int &yIn;
    int &rIn;
    bool &didIt;
};

#endif // INVERTUSINGCIRCLEDIALOG_H
```

Название файла infodialog.h:

```
#ifndef INFODIALOG_H
#define INFODIALOG_H

#include <QDialog>
```

```
namespace Ui {
class InfoDialog;
}

class InfoDialog : public QDialog
{
    Q_OBJECT

public:
    explicit InfoDialog(QWidget *parent = nullptr);
    ~InfoDialog();

private:
    Ui::InfoDialog *ui;
};

#endif // INFODIALOG_H
```

Название файла image.h:

```
#ifndef IMAGE_H
#define IMAGE_H
#include <png.h>
#include <math.h>
#include <QMessageBox>
#include <QString>
#include <stdlib.h>
#include <QImage>
#include <QColor>

class image
{
private:
    int width, height;
    png_byte color_type;
    png_byte bit_depth;

    png_structp png_ptr;
    png_infop info_ptr;
    int number_of_passes;
    png_bytep *pixels_arr;
    bool exist = false;
    void pixelSetColor(int x, int y, QColor color);
    void invertPixelColor(int x, int y);
public:
    void Copy(image other);
    bool doesExists();
    image();
    ~image();
    QImage toQimg();
    png_bytep* getPixels_arr();
    int getWidth();
    int getHeight();
    int readPngFile(QString path);
    void writePngFile(QString path);
```

```
    void drawLine(int x1, int y1, int x2, int y2, QColor color, int
thikness);
    void invertCircle(int x0, int y0, int r);
    int invertCircle(int x1, int y1, int x2, int y2);
    void cutImage(int x1, int y1, int x2, int y2);

};

#endif // IMAGE_H
```

Название файла drawlinemenudialog.h:

```
#ifndef DRAWLINEMENUDIALOG_H
#define DRAWLINEMENUDIALOG_H

#include <QDialog>
#include <QMessageBox>
#include <QColor>
#include <QColorDialog>

namespace Ui {
class DrawLineMenuDialog;
}

class DrawLineMenuDialog : public QDialog
{
    Q_OBJECT

public:
    explicit DrawLineMenuDialog(QWidget *parent, int &x1, int &y1, int
&x2, int &y2, int &thick, bool &did, QColor &color, int width, int
height);
    ~DrawLineMenuDialog();

private slots:
    void on_buttonBox_accepted();

private:
    Ui::DrawLineMenuDialog *ui;
    int &x1In;
    int &x2In;
    int &y1In;
    int &y2In;
    bool &didIt;
    int &thickIn;
    QColor &colorIn;
};

#endif // DRAWLINEMENUDIALOG_H
```

Название файла dialogdrawline.h:

```
#ifndef DIALOGDRAWLINE_H
#define DIALOGDRAWLINE_H

#include <QDialog>
```

```
#include <QString>
#include <QMessageBox>
#include <QColor>
#include <QColorDialog>

namespace Ui {
class DialogDrawLine;
}

class DialogDrawLine : public QDialog
{
    Q_OBJECT

public:
    explicit DialogDrawLine(QWidget *parent, int &thikness, QColor
&color) ;
    ~DialogDrawLine();

private slots:
    void on_buttonBox_accepted();

private:
    Ui::DialogDrawLine *ui;
    int &insideThikness;
    QColor &colorIn;
};

#endif // DIALOGDRAWLINE_H
```

Название файла cutdialog.h:

```
#ifndef CUTDIALOG_H
#define CUTDIALOG_H

#include <QDialog>
#include <QMessageBox>

namespace Ui {
class CutDialog;
}

class CutDialog : public QDialog
{
    Q_OBJECT

public:
    explicit CutDialog(QWidget *parent, int &x1, int &y1, int &x2, int
&y2, bool &did, int width, int height);
    ~CutDialog();

private slots:
    void on_buttonBox_accepted();

private:
    Ui::CutDialog *ui;
    int &x1In;
    int &x2In;
```

```
    int &y1In;
    int &y2In;
    bool &didIt;
};

#endif // CUTDIALOG_H
```