

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

КУРСОВАЯ РАБОТА
по дисциплине «Программирование»
Тема: Обработка PNG изображений

Студент гр. 3388

Снигирев А.А.

Преподаватель

Заславский М.М.

Санкт-Петербург

2024

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Снигирев А.А.

Группа 3388

Тема работы: Обработка *PNG* изображений

Программа должна иметь *CLI* или *GUI*. Более подробно тут:

http://se.moevm.info/doku.php/courses:programming:rules_extra_kurs

Общие сведения

- Формат картинки *PNG* (рекомендуем использовать библиотеку *libpng*)
- 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 треугольника. Треугольник определяется
 - Координатами его вершин
 - Толщиной линий
 - Цветом линий
 - Треугольник может быть залит или нет
 - цветом, которым он залит, если пользователем выбран залитый
2. Находит самый большой прямоугольник заданного цвета и перекрашивает его в другой цвет. Функционал определяется:
 - Цветом, прямоугольник которого надо найти
 - Цветом, в который надо его перекрасить

3. Создать коллаж размера $N*M$ из одного изображения. Коллаж определяется:

- Количество изображений по “оси” Y
- Количество изображений по “оси” X
- Перечень изображений (если выбрана усложненная версия задания)

Содержание пояснительной записи:

Разделы пояснительной записи: Содержание, Введение, Заключение, Список использованных источников.

Предполагаемый объем пояснительной записи:

Не менее 50 страниц.

Дата выдачи задания: 22.03.2024

Дата сдачи реферата: 04.05.2024

Дата защиты реферата: 04.05.2024

Студент		Снигирев А.А.
Преподаватель		Заславский М.М.

АННОТАЦИЯ

Курсовая работа представляет собой программу на языке Си, которая обрабатывает *PNG*-изображение. Программа имеет *CLI* (интерфейс командной строки) для ввода параметров обработки *PNG*-файла пользователем.

Для чтения и записи изображения была использована библиотека *libpng*; для обработки изображения использовались функции стандартных библиотек; для анализов аргументов командной строки использовалась библиотека *getopt*.

SUMMARY

The coursework is a C program that processes a PNG image. The program has a CLI (command line interface) for entering PNG file processing parameters by the user.

The libpng library was used to read and write the image; the functions of standard libraries were used to process the image; the getopt library was used to analyze command-line arguments.

СОДЕРЖАНИЕ

Введение	4
1. Подключаемые библиотеки, макроопределения, структуры	7
2. Функции	8
2.1 Функции чтения и записи <i>PNG</i> -файла	8
2.2 Дополнительные, вспомогательные функции	9

2.3	Основные функции	15
2.4	Функция <i>main</i>	18
	Заключение	19
	Список использованных источников	20
	Приложение А. Результаты тестирования	21
	Приложение Б. Исходный код программы	24

ВВЕДЕНИЕ

Целью данной работы является разработка программы, обрабатывающей *PNG*-изображение, на языке Си.

Для достижения поставленной цели требуется решить следующие задачи:

- разработать функции чтения и записи *PNG*-файла, реализовать записи в структуру *Png*;
- разработать функцию рисования треугольника на изображении;
- разработать функцию поиска наибольшего прямоугольника заданного цвета и перекрашивания в другой цвет;
- разработать функцию, создающую коллаж размера $N*M$ из изображения;
- разработать функцию рисования линии;
- написать *Makefile* для сборки программы;
- протестировать разработанную программу.

1. ПОДКЛЮЧАЕМЫЕ БИБЛИОТЕКИ, МАКРООПРЕДЕЛЕНИЯ, СТРУКТУРЫ

Для корректной работы программы подключены стандартные библиотеки языка Си: *stdlib.h*, *stdio.h*, *math.h*, *cctype.h*, *string.h*.

Также подключена библиотека *png.h* для чтения и записи *PNG*-файла и библиотека *getopt.h* для анализа аргументов командной строки.

Определены две структуры:

- Структура *Png*, хранящая параметры изображения: высоту *height* и ширину *width*, цветовой тип *color_type*, битовую глубину *bit_depth*, количество каналов *channels*, указатель на *png_struct*, указатель на *png_info*, указатель на сетку пикселей;
- Структура *triangle*, содержащая все аргументы, необходимые для рисования треугольника.
- Структура *rect* хранящая в себе параметры прямоугольника

2. ФУНКЦИИ

2.1. Функции чтения и записи PNG-файла

Функция `read_png_file()` принимает на вход указатель на строку `file_name` – имя *PNG*-файла, который нужно считать, а также указатель на структуру `Png image`; с помощью функций из библиотеки *libpng* данные из *IHDR*читываются и записываются в структуру `image`; также происходит динамическое выделение памяти для сетки пикселей с последующей записью в структуру `image`; если на каком-либо этапе считывания *PNG*-файла возникает ошибка, то выводится сообщение о том, какую именно часть файла не удалось считать, и программа завершается.

Функция `write_png_file()` принимает на вход указатель на строку `file_name` – имя *PNG*-файла, куда требуется записать изображение, а также указатель на структуру `Png image`; с помощью функций из библиотеки *libpng* информация о изображении, а также сетка пикселей записывается в *PNG*-файл; если на этапе записи *PNG*-файла возникает ошибка, то она корректно обрабатывается: выводится сообщение о том, что именно не удалось записать, и программа завершается.

2.2. Дополнительные, вспомогательные функции

Функция *free_image_data()* принимает на вход указатель *img* на структуру *Png* и очищает динамическую память, выделенную для хранения сетки пикселей (принцип работы: с помощью цикла *for* проходит по каждой строке пикселей и освобождает динамически выделенную ранее память; далее освобождает память, выделенную под хранение указателей на строки пикселей).

Функция *print_help()* обращается выводит справку о курсовой работе.

Функция *print_info()* принимает на вход указатель *img* на структуру *Png* и печатает в поток вывода основную информацию о *PNG*-файле (принцип работы: получает данные из структуры *Png img* и печатает их в поток вывода).

Функция *check_position()* принимает на вход указатель *img* на структуру *Png*, координаты пикселя *x* и *y*; проверяет, является ли данный пиксель частью изображения, и если является, то возвращает значение 1, иначе – 0 (принцип работы: с помощью условного оператора *if* путём сравнения с высотой и шириной изображения проверяет нахождение пикселя внутри изображения; результат сравнения возвращается).

Функция *check_color()* принимает на вход указатель *img* на структуру *Png*, координаты пикселя *x* и *y*, указатель на массив с цветом *color_arr*; проверяет, совпадает ли фактический цвет пикселя с переданным в функцию цветом *color_arr*, и если совпадает, то возвращает значение 1, иначе – 0 (принцип работы: переходит в сетке пикселей к заданному координатами пикселя и сравнивает по трем каналам соответствие фактического цвета пикселя с переданным в функцию значением *color_arr*).

Функция *set_pixel()* принимает на вход указатель *img* на структуру *Png*,

координаты пикселя x и y , указатель на массив с цветом $color_arr$; изменяет цвет пикселя на переданный в функцию цвет (принцип работы: переходит в сетке пикселей к заданному координатами пикселю и изменяет значения трёх каналов в соответствии с переданным цветом $color_arr$).

Функция $draw_circle()$ принимает на вход указатель img на структуру Png , координаты центра окружности xc и yc , радиус r и указатель на массив с цветом $color_arr$; рисует залитый цветом круг с заданными параметрами (принцип работы: для рисования окружности используется алгоритм Брезенхема: на каждом шаге алгоритма рассматриваются три пикселя, из них выбирается наиболее подходящий путём сравнения расстояний от центра до выбранного пикселя с радиусом окружности, затем координаты наиболее подходящего пикселя передаются в вспомогательную функцию $set_pixel()$, отвечающую за перекрашивание пикселя; таким образом рисуется окружность.

Функция $draw_line()$ принимает на вход указатель img на структуру Png , координаты двух точек $x1$ и $y1$, $x2$ и $y2$, толщину линии $thick$ указатель на массив с цветом $color_arr$; рисует толстую линию по заданным параметрам (назначение: функция для рисования толстых линий; принцип работы: для рисования линии используется алгоритм Брезенхема: сначала происходит вычисление приращений, абсолютных значений и направлений рисования по двум осям: X и Y ; далее с помощью цикла $while$ происходит движение от одной точки к другой путём обновлением значения ошибки, последующим сравнением этого значения с приращением по осям и прибавлением к координатам точки значений направления; на каждом шаге функции $while$ вычисляются координаты пикселей, образующих линии, которые передаются в функцию $draw_circle()$ для рисования линии непосредственно на изображении).

Функция $isinside1()$ принимает на вход координаты трёх вершин

треугольника ax и ay , bx и by , cx и cy , координаты точки x и y ; проверяет, лежит ли точка внутри треугольника, и если лежит, то возвращает значение 1, иначе – 0 (принцип работы: считает значения трёх проекций векторных произведений на ось, перпендикулярную координатной плоскости; для того чтобы точка находилась внутри треугольника должно быть соблюдено условие: все три проекции должны быть одного знака; если условие выполняется, то возвращается 1, иначе – 0).

Функция *fill_triangle()* принимает на вход координаты вершин треугольника, указатель на структуру изображения и массив, отвечающий за цвет. С помощью функции *isinside1()* проходит по всем пикселям картинки и если пиксель лежит внутри треугольника – закрашивает его.

Функция *check_area()* принимает на вход указатель *img* на структуру *Png*, координаты двух точек $x1$ и $y1$, $x2$ и $y2$, указатель на массив с цветом *init_color_arr*; проверяет, состоит ли прямоугольник, заданный координатами левой верхней и правой нижней вершины, полностью из пикселей передаваемого в функцию цвета, и если состоит, то возвращает значение 1, иначе – 0 (принцип работы: с помощью цикла *for* в цикле *for* проходит по каждому пикслю в прямоугольной области, заданной координатами левой верхней и правой нижней вершины, и с помощью функции *check_pixel_color* проверяет каждый пиксль из области на соответствие переданному цвету; если встречает пиксель другого цвета, то возвращается значение 0, иначе – 1).

Функция *make_empty_image()* принимает на вход 2 указателя *img* и *new_img* на структуры *Png*, высоту *height* и ширину *width* нового изображения, флаг *is_have_background*; на основе информации о *img* создаёт новое изображение *new_img* с заданным размером высоты и ширины, выделяя динамически память для сетки пикселей; если флаг *is_have_background* равен 0, то значение цветов пикселей не устанавливается, в противном случае новое изображение будет иметь белый фон (принцип работы: пустому изображению *new_img* присваивается переданная в функцию высота и ширина изображения;

new_img также присваивается цветовой тип, битовая глубина и количество цветовых каналов изображения *img*; далее пустому изображению присваивается указатель на *png_ptr*, а также указатель на *info_ptr*; затем цикл *for* проходит по каждой строке пустого изображения и динамически выделяет память для каждой строки пикселей).

Функция *insert_part_of_image()* принимает на вход 2 указателя *img* и *new_img* на структуры *Png*, координаты прямоугольной области *x1* и *y1*, *x2* и *y2*, координаты левой верхней точки *i* и *j*, флаг *insert_to_new_image_fl*; копирует прямоугольную область и вставляет её в заданное место; если флаг *insert_to_new_image_fl* равен 0, то прямоугольная область вставляется в это же изображение, в противном случае заданная часть изображения *img* вставится в изображение *new_img* (принцип работы: с помощью цикла *for* в цикле *for* проходит по прямоугольной области, записывает текущее значение цвета в *color_arr*, затем, в зависимости от флага *insert_to_new_image_fl*, с помощью функции *set_pixel* изменяет цвет пикселя в изображении, в которое требуется вставить область, на *color_arr*).

2.3. Основные функции

Функция *draw_triangle()* принимает на вход указатель *img* на структуру *Png*, координаты трёх вершин треугольника *x1* и *y1*, *x2* и *y2*, *x3* и *y3*, толщину рёбер *thickness*, указатель на массив с цветом рёбер *line_color*, флаг *is_fill*, указатель на массив с цветом заливки *fill_color*; рисует треугольник по заданным параметрам; если флаг *is_fill* равен 0, то треугольник не заливается цветом, в противном случае рисуется залитый цветом треугольник (принцип работы: если значение флага *is_fill* не равно 0 (т.е. требуется залить треугольник цветом), то функция с помощью цикла *for* проходит по координатам каждой из трёх вершин треугольника, с помощью условного оператора *if* сравнивает минимальные/максимальные координаты точек со значениями *minX*, *maxX*, *minY*, *maxY* и если среди координат вершин находит меньшее/большее значение, то присваивает в соответствующую переменную; далее с помощью цикла *for* в цикле *for* проходит по каждому пикслю в прямоугольной области, внутри которой расположен треугольник, с помощью функции *is_point_in_triangle* проверяет каждую точку на то, находится ли она внутри треугольника, и если находится, то перекрашивает пиксель с помощью *set_pixel*; после заливки треугольника (если треугольник не требуется заливать, то функция сразу переходит к выполнению действий, описанных далее, минуя предыдущие действия) с помощью трёх функций рисования толстых линий *draw_line* на изображении рисуются рёбра треугольника;

Функция *recolor_max_rectangle()* принимает на вход указатель *img* на структуру *Png*, указатель на массив с исходным цветом прямоугольника *init_color_arr*, указатель на массив с цветом *final_color_arr*, в который нужно перекрасить прямоугольник; находит и перекрашивает максимальный по площади прямоугольник заданного цвета в другой цвет (принцип работы: с помощью цикла *for* в цикле *for* проходит по каждому пикслю изображения;

если находит потенциальную левую верхнюю вершину прямоугольника (проверка на равенство фактического цвета пикселя с переданным в функцию *init_color_arr* осуществляется с помощью функции *check_pixel_color*), то с помощью двух циклов *while* идёт от найденной левой верхней вершины к правой верхней вершине и левой нижней вершине, сравнивая цвет каждого пройденного пикселя на соответствие цвету *init_color_arr* (таким образом находится потенциальный прямоугольник заданного цвета); далее считается площадь прямоугольника и сравнивается со значением найденной ранее максимальной площади; если площадь текущего прямоугольника больше, то с помощью функции *check_filled_area* прямоугольник проверяется: состоит ли он полностью из пикселей исходного цвета *init_color_arr*; если состоит, то обновляется значение максимальной площади *max_area*, а также координаты точек максимального прямоугольника *max_x1*, *max_y1*, *max_x2*, *max_y2*; далее, если максимальный по площади прямоугольник был найден, то с помощью цикла *for* в цикле *for* функция проходит по каждому пикселию прямоугольника и перекрашивает в заданный цвет (перекрашивание пикселя происходит с помощью *set_pixel*)).

Функция *make_collage()* принимает на вход 2 указателя *img* и *new_img* на структуры *Png*, количество изображений по оси «Y» и «X» *n* и *m*; создаёт коллаж в *new_img* размером *N*M* из изображения *img* (принцип работы: с помощью *make_empty_image* создаёт новое изображение с высотой и шириной изображения *height * n* и *width * m* соответственно; далее с помощью цикла *for* и функции *insert_part_of_image* исходное изображение копируется *m* раз, тем самым заполняя *new_img*; после работы цикла *new_img* состоит из изображений *img*, расположенных в «строку»; далее с помощью цикла *for* и функции *insert_part_of_image* «строка» изображений копируется *n* раз, формируя коллаж).

2.4. Функция *main*

В функции *main()* реализовано управление программой с помощью аргументов командной строки. С помощью функций из библиотеки *getopt* происходит считывание параметров обработки *PNG*-изображения после соответствующих флагов, проверка на корректность передаваемых значений (в случае ввода некорректного параметра, неверного флага или недостаточного количества аргументов для обработки программа напишет в поток вывода соответствующую информацию об ошибке и завершит работу), и последующая запись параметров в структуру *args params* (справку по работе с программой, доступным флагам и передаваемым значениям можно узнать, если не передавать в командную строку аргументов или если передать флаг *-help* (*-h*)). Далее происходит чтение *PNG*-файла и обработка *PNG*-изображения в соответствии с передаваемыми флагами и параметрами. Далее обработанное изображение записывается в *PNG*-файл, и программа завершается.

Результаты тестирования см. в приложении А.

Разработанный программный код см. в приложении Б.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана программа, управляемая с помощью аргументов командной строки, считывающая *PNG*-изображение, обрабатывающая изображение (виды обработки изображения: рисование каркаса (или залитого) треугольника; перекрашивание максимального по площади прямоугольника заданного цвета в другой цвет; создание коллажа из изображения) в соответствии с передаваемыми аргументами и записывающая обработанное изображение в *PNG*-файл.

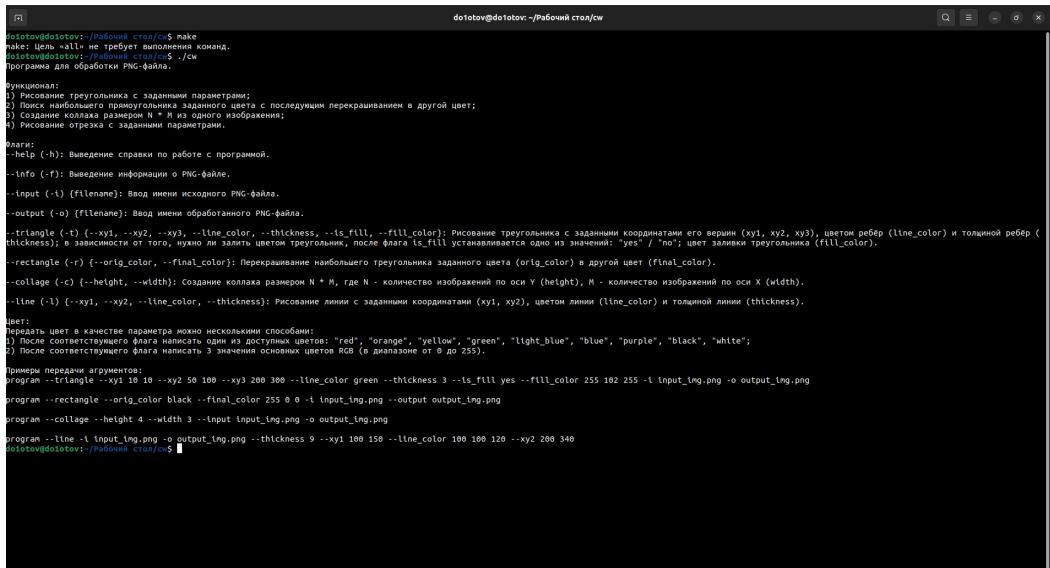
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Керниган Б., Ритчи Д., Язык программирования Си.: Издательство Москва, Вильямс, 2015 г. 304 с.
2. Онлайн-библиотека // Википедия. URL:
[https://ru.wikipedia.org/wiki/Алгоритм_Брезенхэма#:~:text=Алгоритм%20Брезенхёма%20\(англ.,разработан%20Джеком%20Элтоном%20Брезенхёмом%20\(англ.](https://ru.wikipedia.org/wiki/Алгоритм_Брезенхэма#:~:text=Алгоритм%20Брезенхёма%20(англ.,разработан%20Джеком%20Элтоном%20Брезенхёмом%20(англ.) (дата обращения 17.05.2023).
3. Веб-сайт системы вопросов и ответов // stackoverflow. URL:
<https://en.cppreference.com> (дата обращения 16.05.2023).
4. Мануал по работе с библиотекой libpng // libpng.org. URL:
<http://www.libpng.org/pub/png/libpng-1.2.5-manual.html> (дата обращения 15.05.2023).
5. Электронный учебник по программированию на языках Си и С++ // cppstudio. URL: <http://cppstudio.com/> (дата обращения 17.05.2023).

ПРИЛОЖЕНИЕ А

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

1. Вывод справки (./cw):



```
dototov@dototov:~/Рабочий стол/cw$ make
make: Цель «all» не требует выполнения команд.
dototov@dototov:~/Рабочий стол/cw$ ./cw
программа для обработки PNG-файла.

Функционал:
1) Рисование треугольника с заданными параметрами;
2) Поиск наибольшего прямоугольника заданного цвета с последующим перекрашиванием в другой цвет;
3) Создание коллажа размером N * M из одного изображения;
4) Рисование спрэка с заданными параметрами.

Плаги:
--help (-h): Выведение справки по работе с программой.
--info (-i): Выведение информации о PNG-файле.
--input (-i) {filename}: Ввод имени исходного PNG-файла.
--output (-o) {filename}: Ввод имени обработанного PNG-файла.

--triangle (-t) {-xxy1, -xxy2, -xxy3, --line_color, --thickness, --is_fill, --fill_color}: Рисование треугольника с заданными координатами его вершин (x1, x2, x3), цветом ребер (line_color) и толщиной ребер (thickness); в зависимости от того, нужно ли заливать цветом треугольник, после флага is_fill устанавливается одно из значений: "yes" / "no"; цвет заливки треугольника (fill_color).

--rectangle (-r) {-orig_color, --final_color}: Перекрашивание наибольшего треугольника заданного цвета (orig_color) в другой цвет (final_color).

--collage (-c) {-height, --width}: Создание коллажа размером N * M, где N - количество изображений по оси Y (height), M - количество изображений по оси X (width).

--line (-l) {-xxy1, -xxy2, --line_color, --thickness}: Рисование линии с заданными координатами (x1, x2), цветом линии (line_color) и толщиной линии (thickness).

Нет:
Передать цвет в качестве параметра можно несколькими способами:
1) После соответствующего флага написать один из доступных цветов: "red", "orange", "yellow", "green", "light_blue", "blue", "purple", "black", "white";
2) После соответствующего флага написать 3 значения основных цветов RGB (в диапазоне от 0 до 255).

Примеры передачи аргументов:
program --triangle -xxy1 10 10 --xxy2 200 300 --xxy3 255 50 100 --line_color green --thickness 3 --is_fill yes --fill_color 255 102 255 -i input_img.png -o output_img.png
program --rectangle -orig_color black --final_color 255 0 0 -i input_img.png -o output_img.png
program --collage -height 4 --width 3 -i input_img.png -o output_img.png
program --line -i input_img.png -o output_img.png -thickness 9 -xxy1 100 100 150 --line_color 100 100 120 --xxy2 200 340
dototov@dototov:~/Рабочий стол/cw$
```

Рисунок 1.1

2. Обработка изображения: создание коллажа из изображений

Команда: `gcc curse_w.c -lpng && ./a.out --collage --number_x 2 --number_y 2 -i arthas.png -o collage.png`



Рис. 1.1 Изображение до обработки

Рис. 1.2 После обработки

3. Обработка изображения: рисование залитого треугольника

Команда: `gcc curse_w.c -lpng && ./a.out --triangle --points 10.10.50.500.700.-50 --thickness 15 --color 255.0.255 --fill true --fill_color 0.255.255 --input volk.png:`



Рис. 2.1 До обработки



Рис. 2.2 После обработки

4. Обработка изображения: перекрашивание максимального по площади прямоугольника

Команда: `gcc curse_w.c -lpng && ./a.out --biggest_rect --old_color 255.0.0 --new_color 255.0.255 --output new.png LoTR.png:`

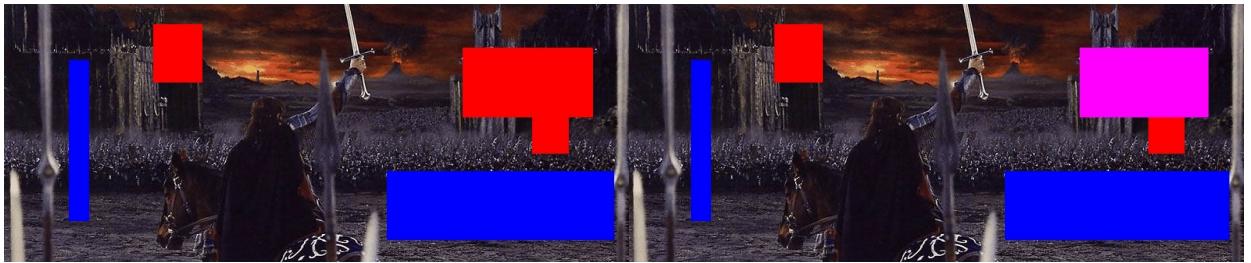


Рис. 3.1 До обработки

Рис. 3.2 После обработки

Обработка исключений

```
root@M14-I3W302:/mnt/c/Users/snigi/OneDrive/Рабочий стол/VS CODE/OOP# gcc curse_w.c -lpng && ./a.out --triangle --points 10.10.50.500.70
0.-50 --thickness 15 --color -255.0.255 --fill true --fill_color 0.255.255 --input volk.png
Неправильное значение цвета
root@M14-I3W302:/mnt/c/Users/snigi/OneDrive/Рабочий стол/VS CODE/OOP# gcc curse_w.c -lpng && ./a.out --triangle --points 10.10.50.500.70
0.-50 --thickness -15 --color -255.0.255 --fill true --fill_color 0.255.255 --input volk.png
Некорректное значение thickness
root@M14-I3W302:/mnt/c/Users/snigi/OneDrive/Рабочий стол/VS CODE/OOP# gcc curse_w.c -lpng && ./a.out --triangle --points 10.10.50.500.70
0.-50 --thickness -15 --color -255.0.255 --fill true --fill_color 0.255.255 --input vlk.png
Некорректное значение thickness
root@M14-I3W302:/mnt/c/Users/snigi/OneDrive/Рабочий стол/VS CODE/OOP# gcc curse_w.c -lpng && ./a.out --triangle --points 10.10.50.500.70
0.-50 --thickness 15 --color 255.0.255 --fill true --fill_color 0.255.255 --input vlk.png
Error in read_png_file function: file could not be opened.
root@M14-I3W302:/mnt/c/Users/snigi/OneDrive/Рабочий стол/VS CODE/OOP# gcc curse_w.c -lpng && ./a.out --triangle --points 10.10.50.500.70
0.-50 --thickness 15 --color 255.0.255 --fill true --fill_color 0.255.255 --input vlk.png
./a.out: unrecognized option '--inpt'
Arguments entered incorrectly.
root@M14-I3W302:/mnt/c/Users/snigi/OneDrive/Рабочий стол/VS CODE/OOP# gcc curse_w.c -lpng && ./a.out --collage --number_x -2 --number_y
2 -i arthas.png -o collage.png
Некорректное значение коллажа
root@M14-I3W302:/mnt/c/Users/snigi/OneDrive/Рабочий стол/VS CODE/OOP# gcc curse_w.c -lpng && ./a.out --triangle --points 10.10.50.500.70
0.-50 --thickness 15 --color 255.0.255 --fill tru --fill_color 0.255.255 --input volk.png
root@M14-I3W302:/mnt/c/Users/snigi/OneDrive/Рабочий стол/VS CODE/OOP#
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: cw.c

```
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <png.h>
#include <getopt.h>
#include <unistd.h>
```

```
typedef struct {
```

```

        unsigned int width, height;
        png_byte color_type;
        png_byte bit_depth;
        png_byte channels;

        png_structp png_ptr;
        png_infop info_ptr;
        png_bytеп *row_pointers;
} Png;
typedef struct {
    int x1;
    int y1;
    int x2;
    int y2;
    int x3;
    int y3;
    int thickness;
    int line_color[3];
    int is_fill;
    int fill_color[3];
} triangle;

typedef struct{
    int old_color[3];
    int new_color[3];
}rect;

void read_png_file(char *file_name, Png *image) {
    unsigned char header[8];
    FILE *fp = fopen(file_name, "rb");
    if (!fp) {
        printf("Error in read_png_file function: file could not be
opened.\n");
        exit(1);
    }
    fread(header, 1, 8, fp);
    if (png_sig_cmp(header, 0, 8)) {
        printf("Error in read_png_file function: file is not
recognized as a PNG.\n");
        exit(2);
    }

    image->png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING,
NULL, NULL, NULL);
    if (!image->png_ptr) {
        printf("Error in read_png_file function:
png_create_read_struct failed.\n");
        exit(3);
    }
    image->info_ptr = png_create_info_struct(image->png_ptr);
    if (!image->info_ptr) {

```

```

        printf("Error in read_png_file function:
png_create_info_struct failed.\n");
        exit(4);
    }
    if (setjmp(png_jmpbuf(image->png_ptr))) {
        fclose(fp);
        printf("Error in read_png_file function: error during
init_io.\n");
        exit(5);
    }
    png_init_io(image->png_ptr, fp);
    png_set_sig_bytes(image->png_ptr, 8);
    png_read_info(image->png_ptr, image->info_ptr);
    image->width = png_get_image_width(image->png_ptr, image-
>info_ptr);
    image->height = png_get_image_height(image->png_ptr, image-
>info_ptr);
    image->bit_depth = png_get_bit_depth(image->png_ptr, image-
>info_ptr);
    image->color_type = png_get_color_type(image->png_ptr, image-
>info_ptr);
    image->channels = png_get_channels(image->png_ptr, image-
>info_ptr);
    if (image->color_type == PNG_COLOR_TYPE_GRAY) {
        printf("The program does not support working with the
PNG_COLOR_TYPE_GRAY color type.\n");
        exit(6);
    } else if (image->color_type == PNG_COLOR_TYPE_GRAY_ALPHA) {
        printf("The program does not support working with the
PNG_COLOR_TYPE_GRAY_ALPHA color type.\n");
        exit(7);
    } else if (image->color_type == PNG_COLOR_TYPE_PALETTE) {
        printf("The program does not support working with the
PNG_COLOR_TYPE_PALETTE color type.\n");
        exit(8);
    }
    png_read_update_info(image->png_ptr, image->info_ptr);

    if (setjmp(png_jmpbuf(image->png_ptr))) {
        printf("Error in read_png_file function: error during
read_image.\n");
        exit(9);
    }
    image->row_pointers = (png_bytеп *) malloc(sizeof(png_bytеп) *
image->height);
    for (int y = 0; y < image->height; y++) {
        image->row_pointers[y] = (png_byte *)
malloc(png_get_rowbytes(image->png_ptr, image->info_ptr));
    }
    png_read_image(image->png_ptr, image->row_pointers);
    fclose(fp);
}
void free_image_data(Png *img) {

```

```

        for (int y = 0; y < img->height; y++) {
            free(img->row_pointers[y]);
        }
        free(img->row_pointers);
    }

void write_png_file(char *file_name, Png *image) {
    FILE *fp = fopen(file_name, "wb");
    if (!fp) {
        printf("Error in write_png_file function: file could not
be opened.\n");
        exit(10);
    }

    image->png_ptr =
png_create_write_struct(PNG_LIBPNG_VER_STRING, NULL, NULL, NULL);
    if (!image->png_ptr) {
        printf("Error in write_png_file function:
png_create_write_struct failed.\n");
        exit(11);
    }
    image->info_ptr = png_create_info_struct(image->png_ptr);
    if (!image->info_ptr) {
        printf("Error in write_png_file function:
png_create_info_struct failed.\n");
        exit(12);
    }
    if (setjmp(png_jmpbuf(image->png_ptr))) {
        printf("Error in write_png_file function: error during
init_io.\n");
        exit(13);
    }
    png_init_io(image->png_ptr, fp);

    if (setjmp(png_jmpbuf(image->png_ptr))) {
        fclose(fp);
        printf("Error in write_png_file function: error during
writing header.\n");
        exit(14);
    }
    png_set_IHDR(image->png_ptr, image->info_ptr, image->width,
image->height, image->bit_depth, image->color_type,
                PNG_INTERLACE_NONE, PNG_COMPRESSION_TYPE_BASE,
PNG_FILTER_TYPE_BASE);
    png_write_info(image->png_ptr, image->info_ptr);

    if (setjmp(png_jmpbuf(image->png_ptr))) {
        fclose(fp);
        printf("Error in write_png_file function: error during
writing bytes.\n");
        exit(15);
    }
    png_write_image(image->png_ptr, image->row_pointers);
}

```

```

    if (setjmp(png_jmpbuf(image->png_ptr))) {
        fclose(fp);
        printf("Error in write_png_file function: error during
writing end of file.\n");
        exit(16);
    }
    png_write_end(image->png_ptr, NULL);
    fclose(fp);
}
int check_position(Png* img, const long long int x, const long
long int y)
{
    if (x >= 0 && x < img->width && y >= 0 && y < img->height) {
        return 1;
    }
    return 0;
}
int check_color(Png* img, const int x, const int y, const int*
palit)
{
    if(img->color_type==PNG_COLOR_TYPE_RGB)
    {
        if(img->row_pointers[y][x*img->channels+0]==palit[0] &&
           img->row_pointers[y][x*img->channels+1]==palit[1] &&
           img->row_pointers[y][x*img->channels+2]==palit[2])
            return 1;
    }
    else if(img->color_type==PNG_COLOR_TYPE_RGBA)
    {
        if(img->row_pointers[y][x*img->channels+0]==palit[0] &&
           img->row_pointers[y][x*img->channels+1]==palit[1] &&
           img->row_pointers[y][x*img->channels+2]==palit[2] &&
           img->row_pointers[y][x*img->channels+3]==palit[3])
            return 1;
    }
    return 0;
}

void set_pixel(Png* img, const int x, const int y, const int* palit)
{
    if(palit[0]>255 || palit[1]>255 || palit[2]>255 || palit[0]<0
    || palit[1]<0 || palit[2]<0){
        //printf("Неправильное значение цвета\n");
        exit(40);
    }

    if(check_position(img, x,y)){
        if(check_color(img, x, y, palit))
        return;
        img->row_pointers[y][x * img->channels+0]=palit[0];
        img->row_pointers[y][x * img->channels+1]=palit[1];

```

```

        img->row_pointers[y][x * img->channels+2]=palit[2];
        //img->row_pointers[y][x * img->channels+3]=255;
    }
}

void print_png_info(Png *image) {
    printf("Высота изображения: %d\n", image->height);
    printf("Ширина изображения: %d\n", image->width);
}
void drawCircle(Png* img, const int x0, const int y0, const int radius, const int* color) {

    if(radius==1 || radius==2){
        set_pixel(img,x0,y0, color);
        set_pixel(img,x0+1,y0, color);
        set_pixel(img,x0,y0+1, color);
        set_pixel(img,x0-1,y0, color);
        set_pixel(img,x0,y0-1, color);
        if(radius==2){
            set_pixel(img,x0+1,y0+1, color);
            set_pixel(img,x0+1,y0-1, color);
            set_pixel(img,x0-1,y0+1, color);
            set_pixel(img,x0-1,y0-1, color);
            set_pixel(img,x0+2,y0, color);
            set_pixel(img,x0,y0+2, color);
            set_pixel(img,x0-2,y0, color);
            set_pixel(img,x0,y0-2, color);
        }
        return;
    }

    int x = 0;
    int y = radius;
    int delta = 1 - 2 * radius;
    int error = 0;
    while(y >= 0) {
        set_pixel(img,x0 + x, y0 + y, color);
        set_pixel(img,x0 + x, y0 - y, color);
        set_pixel(img,x0 - x, y0 + y, color);
        set_pixel(img, x0 - x, y0 - y, color);
        error = 2 * (delta + y) - 1;
        if(delta < 0 && error <= 0) {
            ++x;
            delta += 2 * x + 1;
            continue;
        }
        if(delta > 0 && error > 0) {
            --y;
            delta += 1 - 2 * y;
            continue;
        }
    }
}

```

```

        ++x;
        delta += 2 * (x - y);
        --y;
    }

}

void draw_line(Png *img, int x1, int y1, int x2, int y2, int thick, const int *line_color) {
    const long long int deltaX = abs(x2 - x1);
    const long long int deltaY = abs(y2 - y1);
    const int signX = x1 < x2 ? 1 : -1;
    const int signY = y1 < y2 ? 1 : -1;
    long long int error = deltaX - deltaY;
    drawCircle(img, x2, y2, thick, line_color);
    while (x1 != x2 || y1 != y2) {
        drawCircle(img, x1, y1, thick, line_color);
        if(thick!=0){
            drawCircle(img, x1, y1, thick-1, line_color);
        }
        int error2 = error * 2;
        if (error2 > -deltaY) {
            error -= deltaY;
            x1 += signX;
        }
        if (error2 < deltaX) {
            error += deltaX;
            y1 += signY;
        }
    }
}

int isInside1(int x1, int y1, int x2, int y2, int x3, int y3, int x0, int y0) {
    int prod_1 = (x1 - x0) * (y2 - y1) - (x2 - x1) * (y1 - y0);
    int prod_2 = (x2 - x0) * (y3 - y2) - (x3 - x2) * (y2 - y0);
    int prod_3 = (x3 - x0) * (y1 - y3) - (x1 - x3) * (y3 - y0);
    if ((prod_1 > 0 && prod_2 > 0 && prod_3 > 0) || (prod_1 < 0 && prod_2 < 0 && prod_3 < 0)) {
        return 1;
    }
    return 0;
}

void fill_triangle(Png* img, int X1, int Y1, int X2, int Y2, int X3, int Y3, int* palit)
{
    for(int y=0;y

```

```

check_position(img, x,y))
{
    set_pixel(img, x, y, palit);
}
}

void draw_triangle(Png* img, int X1, int Y1, int X2, int Y2, int
X3, int Y3, int thickness, int* palit, int is_fill, int*
palit_fill){
    if((Y1==Y2 && Y2==Y3) || (X1==X2 && X2==X3)){
        printf("Error: incorrect values of triangle\n");
        exit(18);
    }
    if(is_fill==1){
        fill_triangle(img, X1, Y1, X2, Y2, X3, Y3, palit_fill);
    }

    int a=0;
    if(thickness==1){
        draw_line(img,X1, Y1, X2, Y2,thickness-1, palit);
        draw_line(img,X2, Y2, X3, Y3,thickness-1, palit);
        draw_line(img,X3, Y3, X1, Y1,thickness-1, palit);
        return;
    }
    if(thickness%2)
    a=1;
    draw_line(img,X1, Y1, X2, Y2,thickness/2+a, palit);
    draw_line(img,X2, Y2, X3, Y3,thickness/2+a, palit);
    draw_line(img,X1, Y1, X3, Y3,thickness/2+a, palit);

}

int row_counter(Png* img,int x,int y,int* palit){
    int count=0;
    while(check_color(img,x,y,palit)){
        count++;
    }
    return count;
}
int check_color_on_area(Png* img, int x1, int y1, int x2, int y2,
int* palit)
{
    for(int y=y1;y<y2;y++)
    {
        for(int x=x1;x<x2;x++)
        {
            if(check_color(img, x, y, palit)==0)
            {
                return 0;
            }
        }
    }
}

```

```

        }
    }
    return 1;
}
void recolor_biggest_rectangle(Png* img, int* old_color, int*
new_color) {
    unsigned long long int max_area=0;
    int x_rh, y_rh, x_lw, y_lw;
    for(int y=0;y

```

```

    }
else{
    printf("We are not found rectangle with this color\n");
}

}

void make_empty_image(Png *img, Png *new_img, int height, int width, int is_have_background) {
    new_img->width = width;
    new_img->height = height;
    new_img->color_type = img->color_type;
    new_img->bit_depth = img->bit_depth;
    new_img->channels = img->channels;
    new_img->png_ptr =
png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL, NULL, NULL);
    if (!new_img->png_ptr) {
        printf("Error in make_empty_image function:
png_create_read_struct failed.\n");
        exit(40);
    }
    new_img->info_ptr = png_create_info_struct(new_img->png_ptr);
    if (!new_img->info_ptr) {
        png_destroy_read_struct(&new_img-
>png_ptr, (png_infopp)NULL, (png_infopp)NULL);
        printf("Error in make_empty_image function:
png_create_info_struct failed.\n");
        exit(40);
    }
    png_read_update_info(new_img->png_ptr, new_img->info_ptr);
    new_img->row_pointers = (png_bytep *) malloc(sizeof(png_bytep)
* new_img->height);
    for (int y = 0; y < new_img->height; y++) {
        new_img->row_pointers[y] = (png_byte *)
malloc((png_get_rowbytes(img->png_ptr, img->info_ptr) / img-
>width) * new_img->width);
    }
    if (is_have_background) {
        for (int y = 0; y < new_img->height; y++) {
            for (int x = 0; x < new_img->width; x++) {
                int white_pixel[4] = {255, 255, 255, 255};
                set_pixel(new_img, x, y, white_pixel);
            }
        }
    }
}
void insert_part_of_image(Png *img, Png *new_img, int x1, int y1,
int x2, int y2, int i, int j, int insert_to_new_image_fl) {
    for (int y = y1; y <= y2; y++) {
        for (int x = x1; x <= x2; x++) \
{
            int color_arr[4];

```

```

        color_arr[0] = img->row_pointers[y][x * img->channels
+ 0];
        color_arr[1] = img->row_pointers[y][x * img->channels
+ 1];
        color_arr[2] = img->row_pointers[y][x * img->channels
+ 2];
        if (img->color_type == PNG_COLOR_TYPE_RGBA) {
            color_arr[3] = img->row_pointers[y][x * img-
>channels + 3];
        }
        if (insert_to_new_image_f1) {
            set_pixel(new_img, x + i, y + j, color_arr);
        } else {
            set_pixel(img, x + i, y + j, color_arr);
        }
    }
}
void make_collage(Png *img, Png *new_img, int number_x, int
number_y)
{
    make_empty_image(img, new_img, (int) img->height * number_y,
(int) img->width * number_x, 0);
    for (int i = 0; i < new_img->width; i+=(int)img->width) {

        insert_part_of_image(img, new_img, 0, 0, (int)img->width -
1, (int)img->height - 1, i, 0, 1);
    }
    for (int j = (int) img->height; j < new_img->height; j +=
(int) img->height) {
        insert_part_of_image(new_img, NULL, 0, 0, (int)new_img-
>width - 1, (int) img->height - 1, 0, j, 0);
    }
}

void print_help() {
    printf("Course work for option 4.19, created by Aleksandr
Snigirev.\n");
    /*FILE *file = fopen("help", "rb");
    if (!file) {
        printf("Help_info_file is not found.\n");
        exit(22);
    }
    char ch = (char) fgetc(file);
    while (ch != EOF) {
        printf("%c", ch);
        ch = (char) fgetc(file);
    }
    fclose(file);*/
}

void take_coords(const char* inputString, triangle* points) {
    char* token;

```

```

int i = 0;

char str[100];
strcpy(str, inputString);

token = strtok(str, ".");
while(token != NULL && i < 6) {
    switch(i) {
        case 0:
            points->x1 = atoi(token);
            break;
        case 1:
            points->y1 = atoi(token);
            break;
        case 2:
            points->x2 = atoi(token);
            break;
        case 3:
            points->y2 = atoi(token);
            break;
        case 4:
            points->x3 = atoi(token);
            break;
        case 5:
            points->y3 = atoi(token);
            break;
    }
    token = strtok(NULL, ".");
    i++;
}
}

void take_color(const char* inputString, int* color_arr){
    char* token;
    int i = 0;

    char str[100];
    strcpy(str, inputString);

    token = strtok(str, ".");
    while(token != NULL && i < 3) {
        switch(i) {
            case 0:
                color_arr[0] = atoi(token);
                break;
            case 1:
                color_arr[1] = atoi(token);
                break;
            case 2:
                color_arr[2] = atoi(token);
                break;
        }
        token = strtok(NULL, ".");
        i++;
    }
}

```

```

    }
}

int check_str(char* string){
    int num=0;
    for(int i=0;i<strlen(string);i++)
    {
        if(string[i]=='.')
            num++;
        if(isdigit(string[i])==0 && string[i]!='.' &&
string[i]!='-')
            return 0;
        if(string[i]== '.' && string[i+1]=='.')
            return 0;
    }
    if(num!=5)
        return 0;
    return 1;
}
int check_rgb(char* str){
    int num=0;
    for(int i=0;i<strlen(str);i++)
    {
        if(str[i]=='.')
            num++;
        if(isdigit(str[i]) || str[i]=='.' || str[i]=='-'){
            continue;
        }
        else{
            return 0;
        }
        if(str[i]== '.' && str[i+1]=='.')
            return 0;
    }
    if(num!=2)
        return 0;
    return 1;
}

int main(int argc, char **argw) {
    if(argc<=1){
        print_help();
        return 0;
    }
    Png image, new_image;
    int opt=0;
    triangle trgl;
    rect rct;
    char input_file[255];
    char output_file[255];

```

```

int triangle_param_c = 0;
int rectangle_param_c = 0;
int collage_param_c = 0;
int png_info=0;
int num_x=0;
int num_y=0;
char key;
char* flags="htRCp:T:c:f:w:IO:o:i:n:y:x:";
int idx=0;
struct option long_flags[]={
    {"help", no_argument, NULL, 'h'},//
    {"triangle", no_argument, NULL, 't'},//
    {"collage", no_argument, NULL, 'C'},//
    {"biggest_rect", no_argument, NULL,'R'},//
    {"info", no_argument, NULL,'I'},//
    {"points", required_argument, NULL, 'p'},//
    {"thickness", required_argument, NULL, 'T'},//
    {"color", required_argument, NULL, 'c'},//
    {"fill", required_argument, NULL, 'f'},//
    {"fill_color", required_argument, NULL, 'w'},,
    {"output", required_argument, NULL, 'o'},//
    {"input", required_argument, NULL, 'i'},,
    {"old_color", required_argument, NULL, 'O'},,
    {"new_color", required_argument, NULL, 'n'},,
    {"number_y", required_argument, NULL, 'y'},,
    {"number_x", required_argument, NULL, 'x'},,
    {0,0,0,0}
};

opt=getopt_long(argc, argv, flags,long_flags, &idx);

while(opt!=-1) {

    switch(opt){
        case 'h':
            print_help();
            return 0;
        case 't'://draw triangle
            key='t';
            break;
        case 'R'://recolor rectangle
            key='R';
            break;
        case 'C'://Collage
            key='C';
            break;
        case 'o':
            strcpy(output_file, optarg);
            break;
        case 'i':
            strcpy(input_file, optarg);
            break;
        case 'I':

```

```

        png_info++;
        break;
    case 'p':
        if(check_str(optarg)==0) {
            //printf("Некорректное значение координат
вершин треугольника\n");
            exit(40);
        }
        take_coords(optarg, &trgl);
        triangle_param_c++;
        break;
    case 'T':
        if(atoi(optarg)<=0) {
            //printf("Некорректное значение thickness\n");
            exit(40);
        }
        trgl.thickness=atoi(optarg);
        triangle_param_c++;

        break;
    case 'c':
        if(check_rgb(optarg)==0) {
            //printf("Некорректное значение цвета
линии\n");
            exit(40);
        }
        take_color(optarg, trgl.line_color);
        triangle_param_c++;
        break;
    case 'f':
        if(strcmp(optarg, "true")==0) {
            trgl.is_fill=1;
        }
        else if(strcmp(optarg, "false")==0) {
            trgl.is_fill=0;
        }
        else{
            //printf("Некорректное значение заливки\n");
            //exit(40);
        }
        triangle_param_c++;
        break;
    case 'w':
        if(check_rgb(optarg)==0) {
            //printf("Некорректное значение цвета\n");
            exit(40);
        }
        take_color(optarg, trgl.fill_color);
        triangle_param_c++;
        trgl.is_fill=1;
        break;
    case 'O':

```

```

        if(check_rgb(optarg)==0) {
            //printf("Некорректное значение цвета\n");
            exit(40);
        }
        take_color(optarg, rct.old_color);
        rectangle_param_c++;
        break;
    case 'n':
        if(check_rgb(optarg)==0) {
            //printf("Некорректное значение цвета\n");
            exit(40);
        }
        take_color(optarg, rct.new_color);
        rectangle_param_c++;
        break;
    case 'y':
        if(atoi(optarg)>0) {
            num_y=atoi(optarg);
            collage_param_c++;
        }
        else{
            //printf("Некорректное значение коллажа\n");
            exit(40);
        }
        break;
    case 'x':
        if(atoi(optarg)>0) {
            num_x=atoi(optarg);
            collage_param_c++;
        }
        else{
            //printf("Некорректное значение коллажа\n");
            exit(40);
        }
        break;
    case '?':
        printf("Arguments entered incorrectly.\n");
        return 0;
    }
    opt=getopt_long(argc, argv, flags, long_flags, &idx);
}
if(strlen(input_file)==0)
    strcpy(input_file, argv[argc-1]);
if(strlen(output_file)==0){
    strcpy(output_file, "out.png");
}
if(strcmp(input_file, output_file)==0){
    //printf("Имена входного и выходного файлов совпадают\n");
    exit(40);
}

read_png_file(input_file, &image);
if(png_info) {

```

```

        printf("Тип файла %s: PNG\n", input_file);
        printf("Высота изображения: %d\n", image.height);
        printf("Ширина изображения: %d\n", image.width);
        exit(41);
    }
    switch(key) {
        case 't':
            draw_triangle(&image, trgl.x1, trgl.y1, trgl.x2,
trgl.y2, trgl.x3, trgl.y3, trgl.thickness, trgl.line_color,
trgl.is_fill, trgl.fill_color);
            write_png_file(output_file, &image);
            break;
        case 'R':
            recolor_biggest_rectangle(&image, rct.old_color,
rct.new_color);
            write_png_file(output_file, &image);
            break;
        case 'C':
            make_collage(&image, &new_image, num_x, num_y);
            write_png_file(output_file, &new_image);
            free_image_data(&new_image);
    }
    free_image_data(&image);
}

```

Файл: Makefile

```

all: cw

cw:
    gcc cw.c -lpng -lm -o cw

```