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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка BMP файла

Студент гр. 3341 _____ Моисеева А.Е.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Моисеева А.Е.

Группа 3341

Вариант 7.

Тема работы: Обработка BMP файла

Общие сведения

24 бита на цвет

без сжатия

файл может не соответствовать формату BMP, т.е. необходимо проверка на BMP формат (дополнительно стоит помнить, что версий у формата несколько). Если файл не соответствует формату BMP или его версии, то программа должна завершиться с соответствующей ошибкой.

обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями.

обратите внимание на порядок записи пикселей

все поля стандартных BMP заголовков в выходном файле должны иметь те же значения что и во входном (разумеется кроме тех, которые должны быть изменены).

Программа должна иметь следующую функции по обработке изображений:

(1) Рисование треугольника. Флаг для выполнения данной операции: `--triangle`. Треугольник определяется

Координатами его вершин. Флаг `--points`, значение задаётся в формате `x1.y1.x2.y2.x3.y3` (точки будут (x1; y1), (x2; y2) и (x3; y3)), где x1/x2/x3 – координаты по x, y1/y2/y3 – координаты по y

Толщиной линий. Флаг `--thickness`. На вход принимает число больше 0

Цветом линий. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет)
Треугольник может быть залит или нет. Флаг `--fill`. Работает как бинарное значение: флага нет – false , флаг есть – true.
цветом которым он залит, если пользователем выбран залитый. Флаг `--fill_color` (работает аналогично флагу `--color`)
(2) Находит самый большой прямоугольник заданного цвета и перекрашивает его в другой цвет. Флаг для выполнения данной операции: `--biggest_rect`. Функционал определяется:
Цветом, прямоугольник которого надо найти. Флаг `--old_color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--old_color 255.0.0` задаёт красный цвет)
Цветом, в который надо его перекрасить. Флаг `--new_color` (работает аналогично флагу `--old_color`)
(3) Создать коллаж размера N*M из одного изображения. Флаг для выполнения данной операции: `--collage`. Коллаж представляет собой это же самое изображение повторяющееся N*M раз.
Количество изображений по “оси” Y. Флаг `--number_y`. На вход принимает число больше 0
Количество изображений по “оси” X. Флаг `--number_x`. На вход принимает число больше 0

Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции.

Содержание пояснительной записки:
разделы «Аннотация», «Содержание», «Введение», «Ход работы», «Пример работы программы», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 18.03.2024

Дата сдачи реферата: 15.05.2024

Дата защиты реферата: 16.05.2024

Студент		Моисеева А.Е.
Преподаватель		Глазунов С.А,

АННОТАЦИЯ

В данной курсовой работе была реализована программа, обрабатывающая BMP изображения, не имеющие сжатия, с глубиной 24 бита. Программа проверяет тип изображения, его версию, при соответствии требованиям в дальнейшем обрабатывает его и подаёт на выход изменённую копию изображения. Взаимодействие с программой осуществляется с помощью CLI (интерфейс командной строки).

SUMMARY

In this course has been created a program that processes uncompressed BMP images with a depth of 24 bits. The program checks the type of image, its version, if it meets the requirements, it further processes it and outputs a modified copy of the image. Interaction with the program is performed using CLI (command line interface).

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является создание программы на языке Си для обработки BMP изображений.

Для достижения поставленной цели потребовалось решить ряд задач:

- изучить, как устроены BMP файлы, что они в себе содержат;

- научиться распознавать BMP файлы среди прочих и проверять их прочие характеристики;

- научиться считывать и записывать BMP изображения;

- разработать функцию рисования треугольника на изображении, его заливки;

- разработать функцию поиска наибольшего прямоугольника заданного цвета на изображении и его перекрашивания;

- разработать функцию создания коллажа из исходного изображения по заданным количествам повторений изображения по оси x и по оси y;

- изучить библиотеку *getopt.h*;

- научиться работать с аргументами командной строки, длинными и короткими флагами;

- создать *Makefile* для сборки программы;

- протестировать разработанную программу.

1. РАБОТА С ФАЙЛАМИ

1.1. Проверка файла

Перед получением информации из файла производится проверка на соответствие формату BMP и прочим требуемым параметрам. Это производится с помощью функции *int check_bmp(),* которая принимает на вход имя файла и указатели на структуры *BitmapFileHeader* и *BitmapInfoHeader*. Открывается файл в режиме чтения бинарных данных, сохраняется указатель на файл, затем с помощью *fread* структуры заполняются информацией из файла для дальнейшей проверки. Если файл не открылся, выводится сообщение об ошибке, файл закрывается, выводится сообщение об ошибке и происходит выход из программы с помощью функции *exit*. Далее происходит проверка сигнатуры (должна быть *0x4d42*, что указывает на формат BMP). Файл проверяется на наличие сжатия (должно отсутствовать), размер заголовка (должен быть равен 40 байтам), глубину цвета (должна быть 24 бита на пиксель), размер (максимально допустимый – 50000 пикселей в ширину и высоту). При провале какой-либо из проверок выводится сообщение об ошибке, файл закрывается и происходит выход из программы. В ином случае функция возвращает 1 и происходит закрытие файла.

1.2. Чтение файла

За чтение содержимого файла отвечает функция *Rgb** read_bmp(),* которая принимает на вход имя файла и указатели на структуры *BitmapFileHeader* и *BitmapInfoHeader*. Она открывает файл для чтения в бинарном режиме, считывает информацию об изображении в предоставленные структуры. Затем определяет ширину и высоту изображения из информационного заголовка изображения. Для обработки выравнивания строки в BMP файле вычисляется количество файлов заполнения, необходимое для выравнивания строк данных до размера, кратного 4 байтам. После этого функция выделяет память для двумерного массива пикселей изображения, где каждая строка может содержать

дополнительные байты заполнения в конце. Функция читает каждую строку пикселей изображения (начиная с последней, поскольку изображения BMP хранятся с последней строки), включая байты заполнения и сохраняет их в массиве. После чтения всех данных файл закрывается, и функция возвращает указатель на массив с данными изображения.

1.3 Запись информации в файл

Для записи информации об изображении в файл используется функция *void write_bmp()*, которая принимает на вход имя выходного файла, двумерный массив пикселей и указатели на структуры *BitmapFileHeader* и *BitmapInfoHeader*. Она открывает файл для записи в бинарном режиме и записывает туда информацию об изображении. По аналогии с функцией чтения, вычисляется количество байтов заполнения для выравнивания строк. Функция проходит через все строки изображения (так же начиная с последней) и записывает данные в файл, включая необходимые для выравнивания. После записи всех строк изображения файл закрывается.

1.4 Консольный интерфейс

Консольный интерфейс реализован при помощи библиотеки *getopt.h,* которая применяется в функции *int main().* Она управляет обработкой аргументов командной строки для различных операций обработки изображений. Она поддерживает множество опций, таких как вывод справки, чтение и запись файлов, добавление графических элементов (треугольник, заливка, изменение цвета и создание коллажа). Каждая опция проверяется на уникальность использования и правильность передаваемых параметров. Например, для рисования треугольника необходимо указать цвет, толщину и координаты точек. В случае, если параметры некорректны, или часть из них отсутствует, программа информирует пользователя об ошибке и завершает выполнение. При обработке каждой опции программа считывает необходимые параметры и выполняет соответствующие действия, такие как чтение данных изображения, применение графических изменений и сохранение полученных данных в файл. При этом

обеспечивается корректная обработка ошибок и управление памятью, включая освобождение выделенной памяти после завершения операций. После обработки всех параметров и выполнения запрошенных действий программа завершает работу, освобождая выделенные ресурсы.

2. ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ

2.1 Проверка цветовых компонентов

Проверка корректности цветовых компонентов красного, зелёного и синего производится с помощью функции *Rgb check_color(),* на вход которой поступают три целочисленных значения компонентов. Они должны находиться в диапазоне от 0 до 255, если этого не происходит, выводится сообщение об ошибке и программа завершает выполнение. В случае успешной проверки значения компонентов приводятся к типу *unsigned char* и используются для создания структуры *Rgb*, которая и возвращается функцией.

2.2. Изменение цвета пикселя

Цвет конкретного пикселя в изображении задаётся с помощью функции *void color_pixel(),* которая принимает на вход двумерный массив пикселей, целочисленные координаты пикселя по x и y, цвет, в который необходимо перекрасить пиксель и ширина и высота изображения. Функция проверяет, что указанные координаты находятся в пределах изображения, если это так, цвет пикселя устанавливается в заданный цвет.

2.3. Помощь пользователю

Помощь пользователю предоставляется с помощью функции *void help()*. Она предоставляет информационное сообщение, которое описывает функционал программы и её опции. Это сообщение включает в себя описания команд и параметров, которые можно использовать при запуске программы.

2.4. Справочная информация

Вывод справочной информации об изображении производится с помощью функции *void show_info(),* которая принимает на вход указатели на структуры *BitmapFileHeader* и *BitmapInfoHeader*. Она выводит на экран детальную информацию о заголовках BMP файла, включая такие данные, как размер файла, размер изображения, количество цветов в палитре и прочие.

2.5. Проверка принадлежности точки треугольнику

Проверка принадлежности точки треугольнику по координатам осуществляется с помощью функции *int check_tr(),* которая принимает на вход координаты точки и координаты трёх вершин треугольника. Она вычисляет три площади, образованных между точками треугольника и проверяемой точкой. Если все значения имеют одинаковый знак, то точка находится внутри треугольника.

2.7. Рисование линии

Для рисования линии используется функция *void draw_line(),* которая принимает на вход двумерный массив пикселей, цвет линии, координаты начала и конца линии, ширину линии, высоту и ширину изображения. Функция реализует алгоритм Брезенхэма. Этот алгоритм работает на основе сравнения и корректировки ошибки между желаемой линией и реализованной на растровой сетке. Алгоритм начинает с конечной точки линии и продвигается к начальной, корректируя направление движения с помощью сравнения текущей ошибки с дельтами по осям x и y. При этом каждая точка линии рисуется с заданной толщиной с помощью функции *draw_thick.*

2.8. Толщина линии

Толщина линии реализована через функцию *void draw_thick(),* которая принимает на вход координаты точки, массив пикселей, ширину линии, высоту и ширину изображения и цвет линии. Она рисует дополнительные точки вокруг заданной центральной линии. Толщина линии распространяется равномерно во все стороны от центральной точки (по горизонтали и вертикали) в пределах заданной половины ширины линии. Это достигается за счёт последовательного закрашивания пикселей, начиная от центральной точки и расширяясь в каждом направлении до достижения заданной ширины. Для каждой точки проверяется, находится ли она в пределах изображения, чтобы избежать выхода за границы.

3. ОСНОВНЫЕ ФУНКЦИИ

3.1. Рисование треугольника

Рисование треугольника осуществляется с помощью функции *void triangle()*, которая принимает на вход координаты вершин треугольника, ширину контура, высоту и ширину изображения, массив пикселей, цвет контура. Для рисования треугольника функция последовательно вызывает функцию *draw_line* для рисования трёх сторон фигуры по заданным точкам.

3.2. Заливка треугольника

Заливка треугольника происходит при помощи функции *void fill_tr(),* которая принимает на вход двумерный массив пикселей, цвет заливки, координаты вершин треугольника, высоту и ширину изображения. Она перебирает каждый пиксель изображения и, используя функцию *check_tr*, проверяет, находится ли пиксель внутри заданного треугольника. Если он внутри, то цвет меняется с помощью функции *color_pixel.*

3.3. Поиск самого большого прямоугольника заданного цвета и перекрашивание его в другой цвет

За эту опцию отвечает функция *void find_color_max_rect(),* которая принимает на вход массив пикселей, высоту и ширину изображения, цвет искомого прямоугольника и цвет заливки. Эта функция перебирает все пиксели изображения, и когда находит пиксель заданного цвета, пытается определить ширину и высоту прямоугольника, начинающегося с этой точки. После определения размеров прямоугольника, если он оказывается больше текущего максимального, обновляются данные о максимальном прямоугольнике. Затем, используя координаты и размеры прямоугольника, функция перекрашивает его в новый цвет.

3.4. Создание коллажа из исходного фото

За создание коллажа отвечает функция *Rgb** collage(),* которая принимает на вход количество повторений исходного изображения по осям x и y, указатели

на структуры *BitmapFileHeader* и *BitmapInfoHeader*, высота и ширина изображения и массив пикселей. Функция выделяет память под новое изображение большего размера, затем копирует пиксели из исходного изображения в новое в соответствии с заданными параметрами повторения. После копирования всех пикселей, функция обновляет заголовки файла изображения для отражения нового размера и высвобождает память, занятую исходным изображением. Функция возвращает массив пикселей созданного коллажа.

ЗАКЛЮЧЕНИЕ

Разработана программа на языке программирования Си, обрабатывающая BMP изображения и имеющая CLI. В ходе выполнения работы было изучено устройство BMP файлов; изучены методы считывание и записи файлов; получены навыки обработки изображений; разработаны функции для рисования треугольника и его заливки; поиска наибольшего прямоугольника заданного цвета и его перекрашивания; создания коллажа из исходного изображения; изучена библиотека *getopt.h;* изучена работа с аргументами командной строки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1.https://ru.wikipedia.org/wiki/BMP - структура файла BMP;

2.https://se.moevm.info/lib/exe/fetch.php/courses:programming:programming_cw_metoda_2nd_course_last_ver.pdf.pdf - методические материалы для написания курсовой работы

3.https://www.r5.org/files/books/computers/languages/c/kr/Brian_Kernighan_Dennis_Ritchie-The_C_Programming_Language-RU.pdf – язык программирования Си

4. https://habr.com/ru/articles/55665/ - принцип работы getopt_long

5. https://ru.wikibooks.org/wiki/Реализации_алгоритмов/Алгоритм_Брезенхэма - алгоритм Брезенхэма

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>

#define no_argument 0
#define required_argument 1

#pragma pack(push,1)
typedef struct{
    unsigned short signature;
    unsigned int filesize;
    unsigned short reserved1;
    unsigned short reserved2;
    unsigned int pixelArrOffset;
}BitmapFileHeader;
typedef struct{
    unsigned int headerSize;
    unsigned int width;
    unsigned int height;
    unsigned short planes;
    unsigned short bitsPerPixel;
    unsigned int compression;
    unsigned int imageSize;
    unsigned int xPixelsPerMeter;
    unsigned int yPixelsPerMeter;
    unsigned int colorsInColorTable;
    unsigned int importantColorCount;
}BitmapInfoHeader;
typedef struct {
    unsigned char b;
    unsigned char g;
    unsigned char r;
}Rgb;
#pragma pack(pop)

void help(){
    printf(
    "Course work for option 4.7, created by Anna Moiseeva."
    "\nДля запука программы ипользуются аргументы в следующем порядке:
'programm_name -flag1 -flag2 -flag3 ... input_file_name\n'"
    "Функции программы:\n"
    "help - вывод справоной информации о работе программы\n\n"
    "info - вывод информации об изображении\n\n"
    "triangle - рисование треугольника\n"
    "флаги:\n"
    "points - координаты для треугольника в формате x1.y1.x2.y2.x3.y3\n"
    "thickness - толщина контура треугольника\n"
    "color - цвет контура треугольника в формате rrr.ggg.bbb\n"
    "fill - флаг, который подается при наличии заливки\n"
    "fill_color - цвет заливки при ее наличии в формате rrr.ggg.bbb\n\n"
    "biggest_rect - перекрасить наибольший прямоугольник искомого цвета в
новый\n"
    "флаги:\n"
    "old_color - по этому цвету ищется наибольший прямугольник\n"
    "new_color - цвет заливки наибольшего прямоугольника\n\n"
```

```
    "collage - делает коллаж из исходного изображения\n"
    "флаги\n"
    "number_x - количество повторений исходного изображения по оси x\n"
    "number_y - количество повторений исходного изображения по оси y\n"
    );
}

void show_info(BitmapFileHeader *bmfh, BitmapInfoHeader *bmif) {
    printf("signature:\t%x (%hx)\n", bmfh->signature, bmfh->signature);
    printf("filesize:\t%x (%u)\n", bmfh->filesize, bmfh->filesize);
    printf("reserved1:\t%x (%hx)\n", bmfh->reserved1, bmfh->reserved1);
    printf("reserved2:\t%x (%hx)\n", bmfh->reserved2, bmfh->reserved2);
    printf("pixelArrOffset:\t%x (%u)\n", bmfh->pixelArrOffset, bmfh-
>pixelArrOffset);
    printf("headerSize:\t%x (%u)\n", bmif->headerSize, bmif->headerSize);
    printf("width:     \t%x (%u)\n", bmif->width, bmif->width);
    printf("height:    \t%x (%u)\n", bmif->height, bmif->height);
    printf("planes:    \t%x (%hx)\n", bmif->planes, bmif->planes);
    printf("bitsPerPixel:\t%x (%hx)\n", bmif->bitsPerPixel, bmif->bitsPerPixel);
    printf("compression:\t%x (%u)\n", bmif->compression, bmif->compression);
    printf("imageSize:\t%x (%u)\n", bmif->imageSize, bmif->imageSize);
    printf("xPixelsPerMeter:\t%x (%u)\n", bmif->xPixelsPerMeter, bmif-
>xPixelsPerMeter);
    printf("yPixelsPerMeter:\t%x (%u)\n", bmif->yPixelsPerMeter, bmif-
>yPixelsPerMeter);
    printf("colorsInColorTable:\t%x (%u)\n", bmif->colorsInColorTable, bmif-
>colorsInColorTable);
    printf("importantColorCount:\t%x (%u)\n", bmif->importantColorCount, bmif-
>importantColorCount);
}

int check_bmp(char file_name[], BitmapFileHeader *bmfh, BitmapInfoHeader *bmif){
    FILE *f = fopen(file_name, "rb");
    fread(bmfh, 1, sizeof(BitmapFileHeader), f);
    fread(bmif, 1, sizeof(BitmapInfoHeader), f);
    if (!f){
        fprintf(stderr, "Не удалось открыть файл\n");
        fclose(f);
        exit(45);
    }
    if (bmfh->signature != 0x4d42){
        fprintf(stderr, "Файл должен иметь формат bmp\n");
        fclose(f);
        exit(45);
    }
    if (bmif->compression != 0){
        fprintf(stderr, "Изображение должно быть без сжатия\n");
        fclose(f);
        fclose(f);
        exit(45);
    }
    if (bmif->headerSize != 40){
        fprintf(stderr, "Изображение старой версии, не поддерживается\n");
        fclose(f);
        exit(45);
    }
    if (bmif->bitsPerPixel != 24){
        fprintf(stderr, "Изображение должно иметь 24 бита на пиксель\n");
        fclose(f);
        exit(45);
    }
    if (bmif->width > 50000 || bmif->height > 50000){
```

```
        fprintf(stderr, "Изображение слишком большое\n");
        fclose(f);
        exit(45);
    }
    fclose(f);
    return 1;
}

Rgb** read_bmp(char file_name[], BitmapFileHeader *bmfh, BitmapInfoHeader
*bmif){
    FILE *f = fopen(file_name, "rb");
    fread(bmfh, 1, sizeof(BitmapFileHeader), f);
    fread(bmif, 1, sizeof(BitmapInfoHeader), f);
    unsigned int H = bmif->height;
    unsigned int W = bmif->width;
    size_t padding = (4 - (W * sizeof(Rgb)) % 4) % 4;
    Rgb **arr = malloc(H * sizeof(Rgb*));
    for (int i = H - 1; i >= 0; i--){
        arr[i] = malloc(W * sizeof(Rgb) + padding);
        fread(arr[i], 1, W * sizeof(Rgb) + padding, f);
    }
    fclose(f);
    return arr;
}

void write_bmp(char new_file[], BitmapFileHeader *bmfh, BitmapInfoHeader *bmif,
Rgb **arr){
    FILE *f = fopen(new_file, "wb");
    fwrite(bmfh, 1, sizeof(BitmapFileHeader), f);
    fwrite(bmif, 1, sizeof(BitmapInfoHeader), f);
    unsigned int H = bmif->height;
    unsigned int W = bmif->width;
    size_t padding = (4 - (W * sizeof(Rgb)) % 4) % 4;
    for (int i = H - 1; i >= 0; i--) {
        fwrite(arr[i], 1, W * sizeof(Rgb) + padding, f);
    }
    fclose(f);
}


void color_pixel(int x, int y, Rgb **arr, Rgb color, unsigned int H, unsigned
int W){
    if (x >= 0 && x < W && y >= 0 && y < H) {
        arr[y][x].b = color.b;
        arr[y][x].g = color.g;
        arr[y][x].r = color.r;
    }
}

void draw_thick(int x, int y, Rgb** arr, int width, unsigned int H, unsigned int
W, Rgb color){
    int i = 0;
    int x1 = x;
    while (i <= width / 2 && x1 < W) {
        color_pixel(x1, y, arr, color, H, W);
        i++;
        x1++;
    }
    i = 0;
    x1 = x;
    while (i <= width / 2 && x1 >= 0) {
        color_pixel(x1, y, arr, color, H, W);
```

```
        i++;
        x1--;
    }
    i = 0;
    int y1 = y;
    while (i <= width / 2 && y1 < H) {
        color_pixel(x, y1, arr, color, H, W);
        i++;
        y1++;
    }
    i = 0;
    y1 = y;
    while (i <= width / 2 && y1 >= 0) {
        color_pixel(x, y1, arr, color, H, W);
        i++;
        y1--;
    }
}

void draw_line(int x1, int y1, int x2, int y2, int width, unsigned int H,
unsigned int W, Rgb **arr, Rgb color){
    const int deltaX = abs(x2 - x1);
    const int deltaY = abs(y2 - y1);
    const int signX = x1 < x2 ? 1 : -1;
    const int signY = y1 < y2 ? 1 : -1;
    int error = deltaX - deltaY;
    draw_thick(x2, y2, arr, width, H, W, color);
    while (x1 != x2 || y1 != y2) {
        draw_thick(x1, y1, arr, width, H, W, color);
        int error2 = error * 2;
        if (error2 > -deltaY) {
            error -= deltaY;
            x1 += signX;
        }
        if (error2 < deltaX) {
            error += deltaX;
            y1 += signY;
        }
    }
}

void triangle(int x1, int y1, int x2, int y2, int x3, int y3, int width,
unsigned int H, unsigned int W, Rgb **arr, Rgb color){
    draw_line(x1, y1, x2, y2, width, H, W, arr, color);
    draw_line(x2, y2, x3, y3, width, H, W, arr, color);
    draw_line(x1, y1, x3, y3, width, H, W, arr, color);
}

int check_tr(int x, int y, int x1, int y1, int x2, int y2, int x3, int y3){
    int pr1 = (x1 - x) * (y2 - y1) - (x2 - x1) * (y1 - y);
    int pr2 = (x2 - x) * (y3 - y2) - (x3 - x2) * (y2 - y);
    int pr3 = (x3 - x) * (y1 - y3) - (x1 - x3) * (y3 - y);
    if ((pr1 >= 0 && pr2 >= 0 && pr3 >= 0) || (pr1 <= 0 && pr2 <= 0 && pr3 <=
0)) return 1;
    else return 0;
}

void fill_tr(Rgb **arr, Rgb color, int x1, int y1, int x2, int y2, int x3, int
y3, unsigned int H, unsigned int W){
    for (int i = 0; i < H; i++){
        for (int j = 0; j < W; j++){
            if (check_tr(j, i, x1, y1, x2, y2, x3, y3) == 1){
```

```
                color_pixel(j, i, arr, color, H, W);
            }
        }
    }
}

void find_color_max_rect(unsigned int H, unsigned int W, Rgb **arr, Rgb
color_find, Rgb color_fill){
    int maxrectwidth = 0, maxrectheight = 0;
    int maxrectx = 0, maxrecty = 0;
    for (int i = 0; i < H; i++) {
        for (int j = 0; j < W; j++) {
            Rgb current_color = arr[i][j];
            if (color_find.b == current_color.b && color_find.g ==
current_color.g && color_find.r == current_color.r) {
                int rectwidth = 1;
                while (j + rectwidth < W && color_find.b == arr[i][j +
rectwidth].b && color_find.g == arr[i][j + rectwidth].g && color_find.r ==
arr[i][j + rectwidth].r) {
                    rectwidth++;
                }
                int rectheight = 1;
                int validRow;
                while (i + rectheight < H) {
                    validRow = 1;
                    for (int k = 0; k < rectwidth; k++) {
                        Rgb temp_color = arr[i + rectheight][j + k];
                        if (color_find.b != temp_color.b || color_find.g !=
temp_color.g || color_find.r != temp_color.r) {
                            validRow = 0;
                            break;
                        }
                    }
                    if (!validRow) break;
                    rectheight++;
                }
                if (rectheight * rectwidth > maxrectheight * maxrectwidth) {
                    maxrectheight = rectheight;
                    maxrectwidth = rectwidth;
                    maxrectx = j;
                    maxrecty = i;
                }
            }
        }
    }
    for (int y = maxrecty; y < maxrecty + maxrectheight; y++) {
        for (int x = maxrectx; x < maxrectx + maxrectwidth; x++) {
            color_pixel(x, y, arr, color_fill, H, W);
        }
    }
}

Rgb** collage(int x, int y, unsigned int H, unsigned int W, BitmapFileHeader
*bmfh, BitmapInfoHeader *bmif, Rgb **arr){
    unsigned int collage_w = x * W;
    unsigned int collage_h = y * H;
    Rgb **temp = malloc(collage_h * sizeof(Rgb*));
    for (int i = 0; i < collage_h; i++){
        temp[i] = malloc(collage_w * sizeof(Rgb));
    }
    for (int i = 0; i < collage_h; i++){
        int row = i % H;
```

```
        for (int j = 0; j < collage_w; j++){
            int column = j % W;
            temp[i][j] = arr[row][column];
        }
    }
    for (int i = 0; i < H; i++){
        free(arr[i]);
    }
    free(arr);
    bmif->height = collage_h;
    bmif->width = collage_w;
    bmfh->filesize = sizeof(BitmapFileHeader) + sizeof(BitmapInfoHeader) +
collage_h * collage_w * sizeof(Rgb);
    bmfh->pixelArrOffset = sizeof(BitmapFileHeader) + sizeof(BitmapInfoHeader);
    return temp;
}


Rgb check_color(int r, int g, int b) {
    if (r > 255 || g > 255 || b > 255 || r < 0 || g < 0 || b < 0) {
        fprintf(stdout, "Значения цвета должны быть в диапазоне от 0 до
255.\n");
        exit (45);
    }
    Rgb color;
    color.r = (unsigned char)r;
    color.g = (unsigned char)g;
    color.b = (unsigned char)b;
    return color;
}

int main(int argc, char * argv[]){
    int opt;
    char *input_file_name = NULL;
    char *output_file_name = "out.bmp";
    int output_flag = 0, input_flag = 0, info_flag = 0, triangle_flag = 0,
color_flag = 0, points_flag = 0, thickness_flag = 0, fill_flag = 0,
fill_color_flag = 0;
    int rect_flag = 0, old_color_flag = 0, new_color_flag = 0, collage_flag = 0,
x_flag = 0, y_flag = 0;
    int x1, y1, x2, y2, x3, y3, thickness, x, y, r, g, b;
    Rgb color_triangle, fill_color, old_color, new_color;
    static struct option long_options[] = {
        {"help", no_argument, 0, 'h'},
        {"output", required_argument, 0, 'o'},
        {"input", required_argument, 0, 'i'},
        {"info", no_argument, 0, 'I'},
        {"triangle", no_argument, 0, 'T'},
        {"color", required_argument, 0, 'c'},
        {"points", required_argument, 0, 'p'},
        {"thickness", required_argument, 0, 't'},
        {"fill", no_argument, 0, 'f'},
        {"fill_color", required_argument, 0, 'F'},
        {"biggest_rect", no_argument, 0, 'b'},
        {"old_color", required_argument, 0, 'O'},
        {"new_color", required_argument, 0, 'n'},
        {"collage", no_argument, 0, 'C'},
        {"number_x", required_argument, 0, 'x'},
        {"number_y", required_argument, 0, 'y'},
        {0, 0, 0, 0}
    };
```

```
    while ((opt = getopt_long(argc, argv, "IrfhHCo:L:R:t:c:F:s:d:D:i:",
long_options, NULL)) != -1){
        switch(opt){
            case 'h':
                help();
                return 0;
            case 'o':
                output_flag++;
                output_file_name = optarg;
                break;
            case 'i':
                input_flag++;
                input_file_name = optarg;
                break;
            case 'I':
                info_flag++;
                break;
            case 'T':
                triangle_flag++;
                break;
            case 'c':
                color_flag++;
                if (sscanf(optarg, "%d.%d.%d", &r, &g, &b) != 3) {
                    fprintf(stdout, "Неверный формат цвета. Используйте формат
rrr.ggg.bbb.\n");
                    exit(45);
                }
                color_triangle = check_color(r, g, b);
                break;
            case 'p':
                points_flag++;
                sscanf(optarg, "%d.%d.%d.%d.%d.%d", &x1, &y1, &x2, &y2, &x3,
&y3);
                break;
            case 't':
                thickness_flag++;
                thickness = atoi(optarg);
                if (thickness <= 0){
                    fprintf(stdout, "Неверная толщина линии\n");
                    exit(45);
                }
                break;
            case 'f':
                fill_flag++;
                break;
            case 'F':
                fill_color_flag++;
                if (sscanf(optarg, "%d.%d.%d", &r, &g, &b) != 3) {
                    fprintf(stdout, "Неверный формат цвета. Используйте формат
rrr.ggg.bbb.\n");
                    exit(45);
                }
                fill_color = check_color(r, g, b);
                break;
            case 'b':
                rect_flag++;
                break;
            case 'O':
                old_color_flag++;
                if (sscanf(optarg, "%d.%d.%d", &r, &g, &b) != 3) {
                    fprintf(stdout, "Неверный формат цвета. Используйте формат
rrr.ggg.bbb.\n");
```

```
                    exit(45);
                }
                old_color = check_color(r, g, b);
                break;
            case 'n':
                new_color_flag++;
                if (sscanf(optarg, "%d.%d.%d", &r, &g, &b) != 3) {
                    fprintf(stdout, "Неверный формат цвета. Используйте формат
rrr.ggg.bbb.\n");
                    exit(45);
                }
                new_color = check_color(r, g, b);
                break;
            case 'C':
                collage_flag++;
                break;
             case 'x':
                x_flag++;
                x = atoi(optarg);
                if (x <= 0){
                    fprintf(stdout, "Неверное количество повторений\n");
                    exit(45);
                }
                break;
            case 'y':
                y_flag++;
                y = atoi(optarg);
                if (y <= 0){
                    fprintf(stdout, "Неверное количество повторений\n");
                    exit(45);
                }
                break;
            default:
                fprintf(stdout, "Неверный флаг.\n");
                exit(45);
        }
    }

    if ((triangle_flag > 0 && rect_flag > 0) || (rect_flag > 0 && collage_flag >
0) || (triangle_flag > 0 && collage_flag > 0) || (rect_flag > 0 && collage_flag
> 0 && triangle_flag > 0)) {
        fprintf(stdout, "Нельзя использовать несколько функций вместе.\n");
        exit(45);
    }

    if (rect_flag > 1 || triangle_flag > 1 || collage_flag > 1 || info_flag > 1
|| points_flag > 1 || color_flag > 1 || thickness_flag > 1 || x_flag > 1 ||
y_flag > 1 || fill_color_flag > 1 || fill_flag > 1 || old_color_flag > 1 ||
new_color_flag > 1 || output_flag > 1 || input_flag > 1) {
        fprintf(stdout, "Флаги могут быть использованы только один раз\n");
        exit(45);
    }

    if (optind < argc && input_file_name == NULL) {
        if (strcmp(argv[optind], output_file_name) == 0) {
            fprintf(stdout, "Имя входного файла и имя выходного файла не должны
совпадать.\n");
            exit(45);
        }
        input_file_name = argv[optind];
    }
```

```
    if (input_file_name == NULL) {
        fprintf(stdout, "Входной файл не указан.\n");
        exit(45);
    }

    BitmapFileHeader *bmfh = malloc(sizeof(BitmapFileHeader));
    BitmapInfoHeader *bmif = malloc(sizeof(BitmapInfoHeader));
    if (info_flag) {
        show_info(bmfh, bmif);
        return 0;
    }
    if (check_bmp(input_file_name, bmfh, bmif)){
        Rgb **arr = read_bmp(input_file_name, bmfh, bmif);
        unsigned int H = bmif->height;
        unsigned int W = bmif->width;
        if (triangle_flag){
            if (triangle_flag && !(color_flag && thickness_flag &&
points_flag)){
                fprintf(stdout, "Необходимо указать все параметры для флага --
triangle.\n");
                exit(45);
            }
            if (triangle_flag && (old_color_flag || new_color_flag || x_flag ||
y_flag)){
                fprintf(stdout, "Указаны лишние параметры\n");
                exit(45);
            }
            if (fill_flag && !fill_color_flag){
                fprintf(stdout, "Необходимо указать цвет заливки\n");
                exit(45);
            }
            if (fill_color_flag && !fill_flag){
            fprintf(stdout, "Отсутствует флаг для заливки, заливка будет
проигнорирована\n");
            }
            if (fill_flag && fill_color_flag){
                fill_tr(arr, fill_color, x1, y1, x2, y2, x3, y3, H, W);
            }
            triangle(x1, y1, x2, y2, x3, y3, thickness, H, W, arr,
color_triangle);
        }
        if (rect_flag){
            if (rect_flag && !(old_color_flag && new_color_flag)){
                fprintf(stdout, "Необходимо указать все параметры для флага --
biggest_rect.\n");
                exit(45);
            }
            if (rect_flag & (thickness_flag || color_flag || points_flag ||
x_flag || y_flag)){
                fprintf(stdout, "Указаны лишние параметры\n");
                exit(45);
            }
            find_color_max_rect(H, W, arr, old_color, new_color);
        }
        if (collage_flag){
            if (collage_flag && !(x_flag && y_flag)){
                fprintf(stdout, "Необходимо указать все параметры для флага --
collage.\n");
                exit(45);
            }
            if (collage_flag && (thickness_flag || color_flag || points_flag ||
old_color_flag || new_color_flag)){
```

```
                fprintf(stdout, "Указаны лишние параметры\n");
                exit(45);
            }
            arr = collage(x, y, H, W, bmfh, bmif, arr);
        }
        write_bmp(output_file_name, bmfh, bmif, arr);
        for (int i = 0; i < bmif->height; i++){
            free(arr[i]);
        }
        free(arr);
    }
    free(bmfh);
    free(bmif);
    return 0;
}
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Фото для обработки – рисование треугольника

```
./a.out --input ./image.bmp --output ./output.bmp --triangle --points
50.50.100.100.150.50 --color 255.0.0 --thickness 5 --fill --fill_color
0.0.255
```

Результат работы программы:

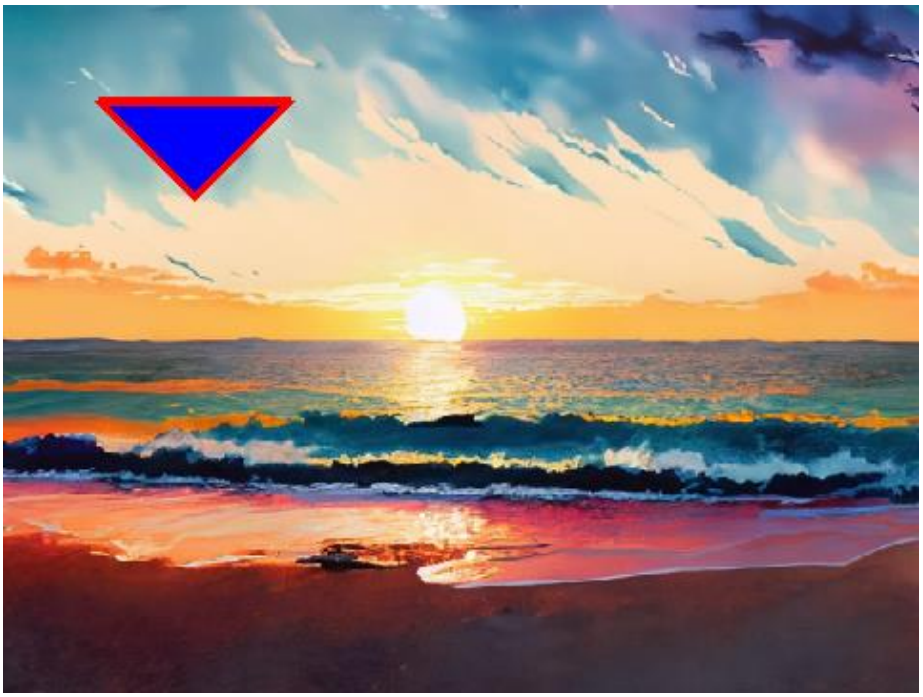

Фото для обработки – поиск наибольшего прямоугольника заданного цвета и перекрашивание его в новый

```
./a.out --input ./test.bmp --output ./output1.bmp --biggest_rect --old_color 0.255.0 --new_color 255.0.0
```

Результат работы программы:

Фото для обработки – создание коллажа из исходного изображения

```
./a.out --input ./im1.bmp --output ./out.bmp --collage --number_x 3 --
number_y 2
```

Результат работы программы: