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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

ТЕМА: РАБОТА С ИЗОБРАЖЕНИЯМИ НА ЯЗЫКЕ СИ

Студент гр. 3342 _____ Роднов И.С.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Роднов И.С.

Группа 3342

Тема работы: Работа с изображениями

Исходные данные:

Вариант 4.8

Задание

Программа должна реализовывать весь следующий функционал по обработке 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 отрезка. Флаг для выполнения данной операции: `--line`. Отрезок определяется:

- координатами начала. Флаг `--start`, значение задаётся в формате `x.y`, где x – координата по x, y – координата по y
- координатами конца. Флаг `--end` (аналогично флагу `--start`)
- цветом. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет)
- толщиной. Флаг `--thickness`. На вход принимает число больше 0

(2) Отражение заданной области. Флаг для выполнения данной операции: `--mirror`. Этот функционал определяется:

- выбором оси относительно которой отражать (горизонтальная или вертикальная). Флаг `--axis`, возможные значения `x` и `y`
- Координатами левого верхнего угла области. Флаг `--left_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y
- Координатами правого нижнего угла области. Флаг `--right_down`, значение задаётся в формате `right.down`, где right – координата по x, down – координата по y

(3) Рисование пентаграммы в круге. Флаг для выполнения данной операции: `--pentagram`. Пентаграмма определяется:

- координатами ее центра и радиусом. Флаги `--center` и `--radius`. Значение флаг `--center` задаётся в формате `x.y`, где x – координата по оси x, y – координата по оси y. Флаг `--radius` На вход принимает число больше 0
- толщиной линий и окружности. Флаг `--thickness`. На вход принимает число больше 0

- цветом линий и окружности. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет)

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

Содержание пояснительной записки:

Разделы пояснительный записки: «Содержание», «Введение», «Структуры», «Функции», «Сборка программы», «Заключение», «Список использованных источников», «Приложение А. Примеры работы программы», «Приложение Б. Исходный код программы».

Предполагаемый объем пояснительной записки: не менее 30 страниц.

Дата выдачи задания: 18.03.2024

Дата сдачи реферата: 22.05.2024

Дата защиты реферата: 22.05.2024

Студент _____ Роднов И.С.

Преподаватель _____ Глазунов С.А.

АННОТАЦИЯ

Цель курсовой работы – создать программу для работы с изображениями bmp формата на языке Си. В программе реализовано взаимодействие с пользователем при помощи CLI (Command Line Interface). На вход программе подаются флаги и аргументы, после обработки которых программа изменяет изображение в соответствии с заданием.

SUMMARY

The goal of the coursework is to create a program for working with BMP image files using the C language. The program interacts with the user through a Command Line Interface (CLI). The program takes flags and arguments as input, processes them, and then modifies the image according to the task specified.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: написать программу на языке С, которая считывает изображение и обрабатывает его требуемым пользователем образом. Для этого требуется реализовать:

- Загрузку, хранение изображений из файла и запись изображения в файл;
- Ввод аргументов из командной строки;
- Функции для рисования на загруженном изображении.

1. СТРУКТУРЫ

1.1. Структура BitmapFileHeader

Структура BitmapFileHeader состоит из таких полей как:

- unsigned short signature - поле заголовка, используемое для идентификации файла BMP и DIB, имеет шестнадцатеричное значение, равное BM в ASCII.
- unsigned int filesize - размер файла BMP в байтах.
- unsigned short reserved1 - зарезервировано; фактическое значение зависит от приложения, создающего изображение.
- unsigned short reserved2 - зарезервировано; фактическое значение зависит от приложения, создающего изображение.
- unsigned int pixelArrOffset - смещение, т. е. начальный адрес байта, в котором находятся данные изображения (массив пикселей).

1.2. Структура BitmapInfoHeader

Структура BitmapInfoHeader состоит из таких полей как:

- unsigned int headerSize – размер этого заголовка в байтах.
- unsigned int width – ширина изображения в пикселях.
- unsigned int height – длина изображения в пикселях.
- unsigned short planes – количество цветовых плоскостей.
- unsigned short bitsPerPixel – глубина цвета изображения.
- unsigned int compression – используемый метод сжатия.
- unsigned int imageSize – размер изображения.
- unsigned int xPixelsPerMeter – горизонтальное разрешение изображения.
- unsigned int yPixelsPerMeter – вертикальное разрешение изображения.
- unsigned int colorsInColorTable – количество цветов в цветовой палитре.
- unsigned int importantColorCount – количество используемых важных цветов.

1.3. Структура RGB

Структура RGB состоит из таких полей как:

- unsigned char b – синяя компонента цвета.
- unsigned char g – зелёная компонента цвета.
- unsigned char r – красная компонента цвета.

2. ФУНКЦИИ

2.1. Функции рисования линии

Функция draw_line (void draw_line(Rgb **arr, int H, int W, int x0, int x1, int y0, int y1, Rgb color, int thickness)) принимает на вход двумерный массив пикселей, высоту и ширину изображения, координаты начала и конца линии, цвет линии, а так же ее толщину. Для рисования линии используется алгоритм Брезенхема. При условии, что толщина линии более 1 пикселя, вызывается дополнительная функция рисования закрашенной окружности. При помощи уравнения окружности в каждом пикселе прямой рисуется закрашенная окружность с радиусом равным половине толщины линии, за счет чего толщина линии реализуется верно. Во время рисования всегда проверяются координаты на предмет выхода за границы, чтобы не затронуть что-то вне изображения.

2.2 Функции отражения изображения

Функция mirror_picture (void mirror_picture(Rgb **arr, int H, int W, char* axis, int xlup, int ylup, int xrd, int yrd)) принимает на вход двумерный массив пикселей, высоту и ширину изображения, ось, относительно которой нужно отразить область, координаты левого верхнего и правого нижнего углов области. Первым делом вызвается функция check_borders_mirror(), которая проверяет координаты области, и если координаты находятся вне области переносит их в памяти на край изображения, так как условие задачи требует обрабатывать картинку до максимально возможного случая. Затем в зависимости от оси – x или y вызываются функции x_mirror или y_mirror соответственно. Если ось указана неверно, программа завершается с сообщением об ошибке. В функциях отражения логика следующая – в зависимости от оси область делится на 2 равные части и при помощи дополнительной переменной Rgb tmp пиксели меняются местами в этих частях.

2.3 Функции рисования пентаграммы в круге

Функция draw_pentagram(void draw_pentagram(Rgb **arr, int H, int W, int xc, int yc, int radius, Rgb color, int thickness)) принимает на вход двумерный массив пикселей, высоту и ширину изображения, координаты центра окружности, ее радиус, цвет и толщину окружности и звезды. Первым делом вызывается функция рисования окружности с нужной толщиной. При помощи уравнения окружности вычисляются координаты наименьшей и наибольшей окружностей а, затем заливается все пространство между ними. После в функции draw_pentagram() вызывается draw_star() для рисования звезды внутри окружности. В ней вызывается функция для определения вершин звезды. При помощи математических формулы находятся вершины, а затем между ними рисуются линии и получается звезда.

2.4 Функции работы с файлами

Функция read_bitmap(Rgb ** read_bitmap(const char * input_filename, BitmapFileHeader * bmfh, BitmapInfoHeader * bmif, Rgb*** arr)) считывает переданный файл с структуры BitmapFileHeader и BitmapInfoHeader, а также в двумерный массив пикселей.

Функция check_bmp (int check_bmp(BitmapFileHeader bmfh, BitmapInfoHeader bmif)) проверяет соответствие ожидаемому формату файла.

Функция write_bitmap (void write_bitmap(const char * output_filename, BitmapFileHeader bmfh, BitmapInfoHeader bmif, Rgb ***arr, int H, int W)) записывает в новый файл все данные, полученные после обработки изображения.

2.5 Прочие функции

Функция printFileHeader (void printFileHeader(BitmapFileHeader header)) печатает всю информацию о заголовке файла.

Функция printInfoHeader (void printInfoHeader(BitmapInfoHeader header)) печатает всю информацию о заголовке изображения.

Функция help (void p_help()) выводит справку.

2.6 Функция main

В функции int main() реализуется CLI при помощи getopt_long. Обрабатываются все возможные флаги, которые могут быть заданы пользователем, так же вызывается функция чтения файла, после выполнения задачи, вызывается функция записи файла. Вспомогательные функции ддля обработки флагов – pars_args() и pars_color(), они приводят аргументы из строчного вида в котором они подаются на вход, в удобный для использования функциях. А так же get_number(), которая правильно обрабатывает числа и числовые последовательности и проверяет аргументы на валидность.5

Примеры работы программы см. в приложении А.

Разработанный программный код см. в приложении Б.

3 СБОРКА ПРОГРАММЫ

Для сборки программы использовался Makefile, в котором программа собирается в файл “cw”. Целью использования Makefile было добавление флага -lm в компиляцию, для использования библиотеки math.h

ЗАКЛЮЧЕНИЕ

Была написана программу на языке C, которая считывает изображение и обрабатывает его требуемым пользователем образом. Для этого были реализованы:

- Загрузка, хранение изображений из файла и запись изображения в файл;
- Ввод аргументов из командной строки;
- Функции для рисования на загруженном изображении.

Для загрузки изображений использованная функция read_bitmap, читающая bmp-файл. Ввод аргументов из командной строки был осуществлен с помощью функции getopt_long из стандартной библиотеки Си. Программа при любом желании пользователя не упадёт с ошибкой, а корректно завершит работу.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Курс “Программирование на Си. Практические задания. Второй семестр”. URL https://e.moevm.info/course/view.php?id=8

2. Язык программирования С / Керниган Брайан, Ритчи Деннис. СПб.: "Финансы и статистика", 2003.

3. Учебно-методическое пособие «БАЗОВЫЕ СВЕДЕНИЯ К ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ». ВТОРОЙ СЕМЕСТР» / М. М. Заславский, А. А. Лисс, А. В. Гаврилов, С. А. Глазунов, Я. С. Государкин, С. А. Тиняков, В. П. Голубева, К. В. Чайка, В. Е. Допира. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2024.

ПРИЛОЖЕНИЕ А
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

ПРИМЕР 1 – вывод информации о файле

```
signature:      4d42 (19778)
filesize:       141a62 (1317474)
reserved1:      0 (0)
reserved2:      0 (0)
pixelArrOffset: 36 (54)
headerSize:     28 (40)
width:          30c (780)
height:         233 (563)
planes:         1 (1)
bitsPerPixel:   18 (24)
compression:    0 (0)
imageSize:      0 (0)
xPixelsPerMeter:        2e23 (11811)
yPixelsPerMeter:        2e23 (11811)
colorsInColorTable:     0 (0)
importantColorCount:    0 (0)
```

ПРИМЕР 2 – вывод информации о файле.

```
Course work for option 4.8, created by Ivan Rodnov.
--help, -h: get info, how to use programm
--info, -I: print info about file
--input, -i: name of input file
--output, -o: name of output file
--line, -l: function of drawing line
--start, -s: coords(x0, yo), where function start draw line
--end, -e: coords(x1, y1)< where function end draw line
--color, -c: color of line or petagram in circle'
--thickness, -t: the thickness of line of pentagram
--mirror, -m: function of mirroring the area on picture
-axis, -a: the axis of mirroring
--left_up, L: left_up coords of area to mirror
--right_down, -r: right_down coords of area to mirror
--pentagram, -p: function which draws pentagram in circle
--center, -C: coords(xc, yc) of circle
--radius, -R: radius of circle
```

ПРИМЕР 3 – ошибка цвета.

```
vanorodno@kaban:~/workspace$ ./cw --pentagram  --center -359.385 --radius 732 --thickness 600 --col
or 900.900.-900 --input ./simpsonsvr.bmp --output ./output.bmp
Error, wrong colorvanorodno@kaban:~/workspace$
```

ПРИМЕР 4 – ошибка файла.

```
vanorodno@kaban:~/workspace$ ./cw --pentagram  --center -359.385 --radius 732 --thickness 600 --col
or 900.900.-900 --input ./banana.bmp --output ./output.bmp
Error opening file: No such file or directory
```

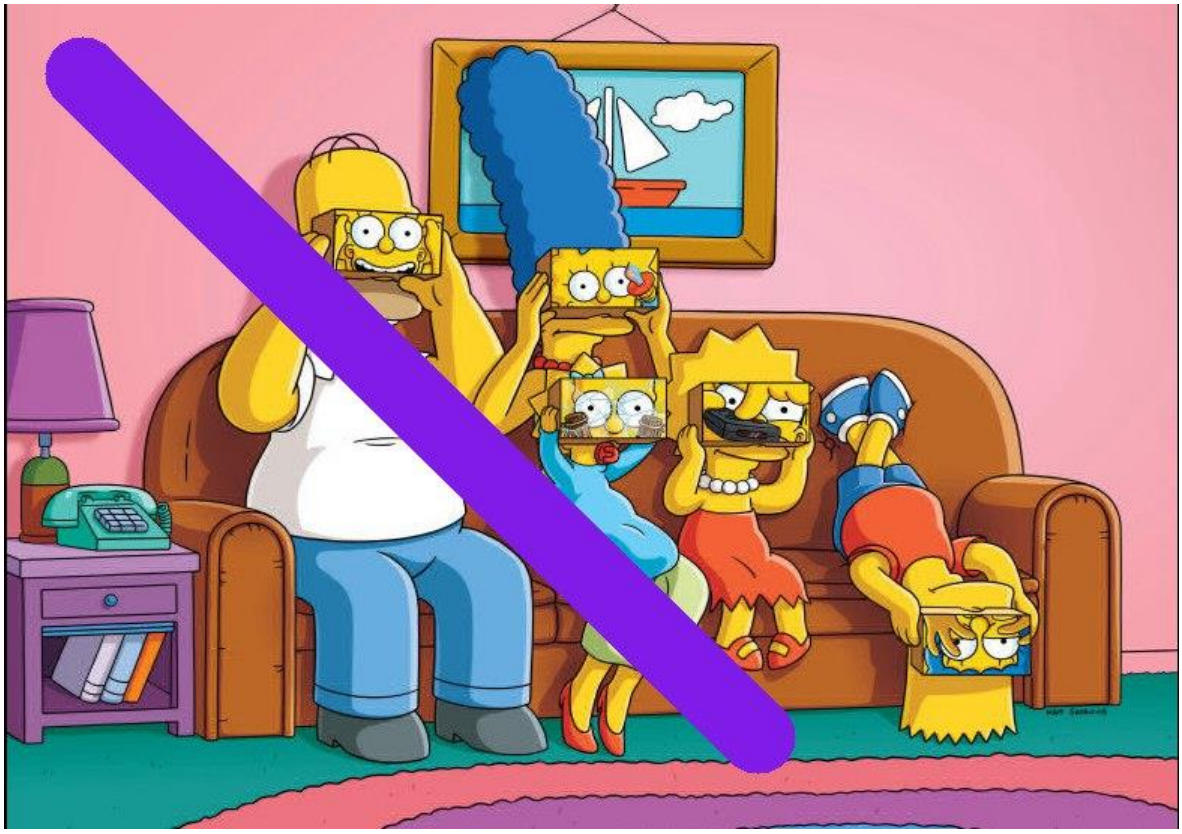
ПРИМЕР 5 –ошибка аргумента.

```
vanorodno@kaban:~/workspace$ ./cw --pentagram  --center 3bb52.bbbbb --radius 2a52 --thickness 20 --color 52.52.52 --input ./simpsonsvr.bmp --output
./output.bmp
Error, wrong argvanorodno@kaban:~/workspace$
```

ПРИМЕР 6 – Рисование линии.

```
vanorodno@kaban:~/workspace$ ./cw --line  --start 50.50 --end 50
0.500 --thickness 50 --color 129.27.231 --input ./simpsonsvr.bmp
 --output ./output.bmp
```

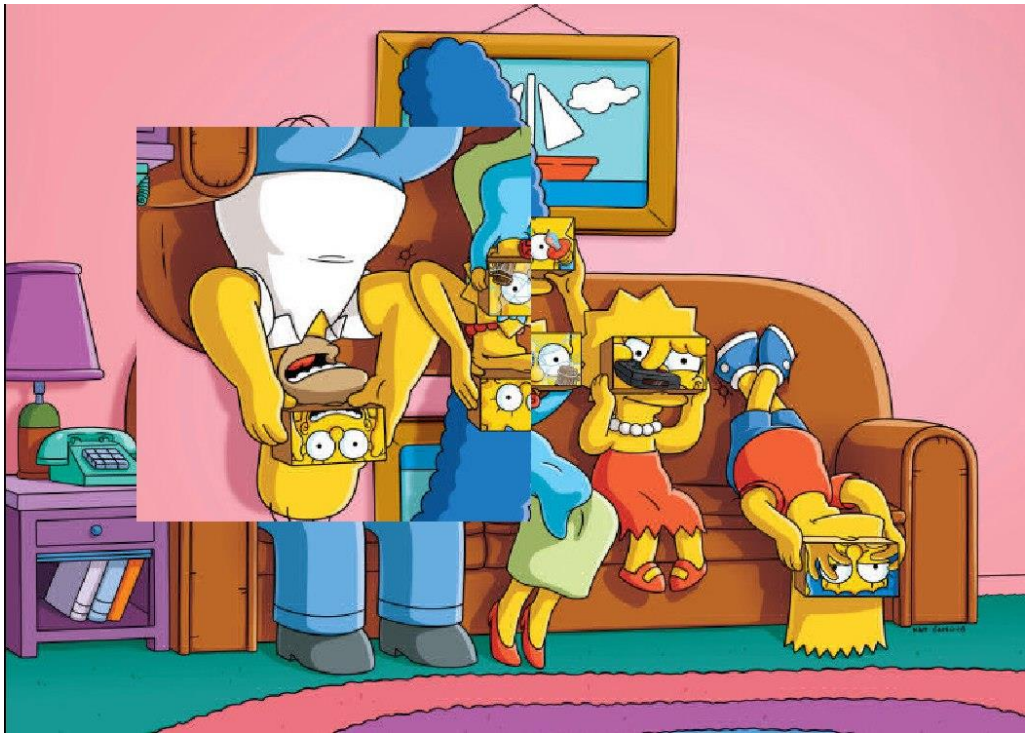
Рис_1 – Результат_1:



ПРИМЕР 7 – Отражение области

Параметры запуска: ./cw --mirror --left_up 100.100 --right_down 400.400 --axis y --input ./simpsonsvr.bmp --output ./output.bmp

Рис_2 – Результат_2:



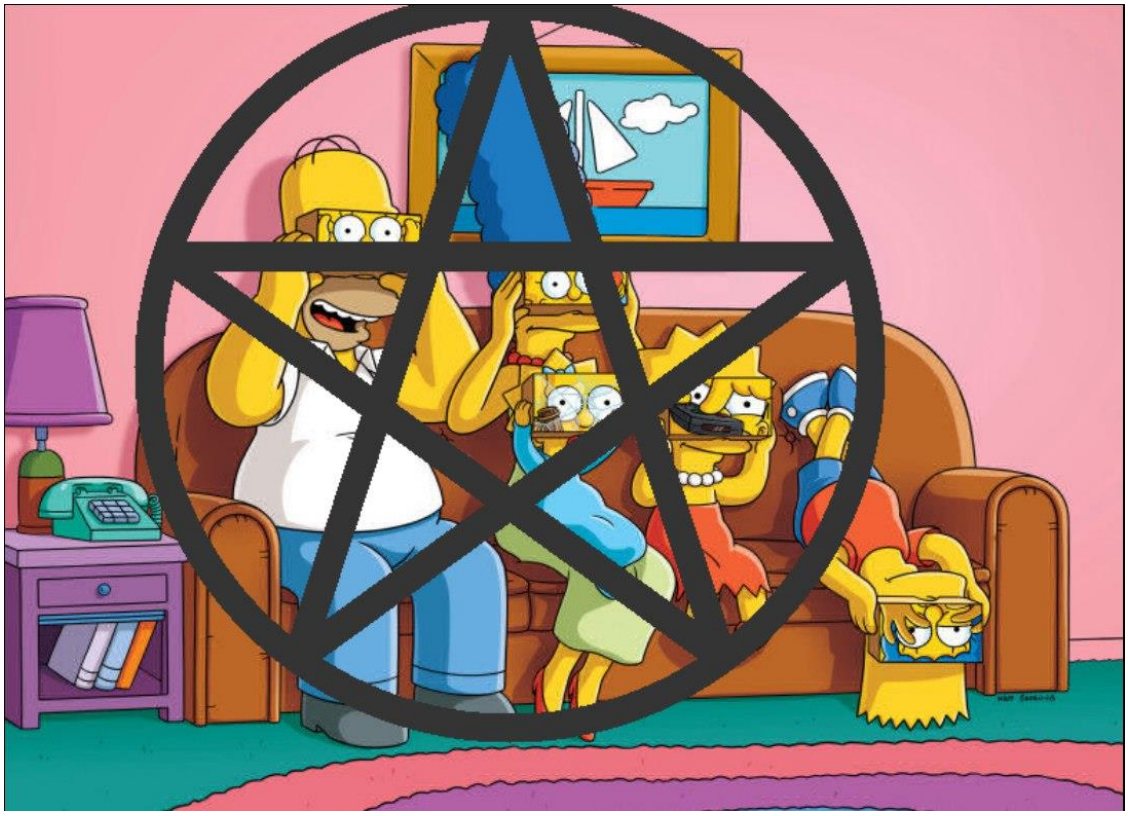
Параметры запуска: ./cw --mirror --left_up 100.100 --right_down 400.400 --axis x --input ./simpsonsvr.bmp --output ./output.bmp

Рис_3 – Результат_3:

ПРИМЕР 8 –Рисование пентаграммы в круге.

Параметры запуска: ./cw --pentagram --center 352.252 --radius 252 --thickness 20 --color 52.52.52 --input ./simpsonsvr.bmp --output ./output.bmp

Рис_4 – Результат_4:



ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД ПРОГРАММЫ

Makefile:

```
CC=gcc

all: exe

exe: main.c
$(CC) main.c -o cw -lm
```

Основной файл main.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <stdbool.h>
#include <ctype.h>
#include <getopt.h>
#include <math.h>

#define PI 3.14159265

#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 print_author_info(){
        printf("Course work for option 4.8, created by Ivan
Rodnov.\n");
    }

    void p_help(){
         printf("--help, -h: get info, how to use programm\n");
         printf("--info, -I: print info about file\n");
         printf("--input, -i: name of input file\n");
         printf("--output, -o: name of output file\n");
         printf("--line, -l: function of drawing line\n");
         printf("--start, -s: coords(x0, yo), where function start
draw line\n");
         printf("--end, -e: coords(x1, y1)< where function end draw
line\n");
         printf("--color, -c: color of line or petagram in circle'\n");
         printf("--thickness, -t: the thickness of line of
pentagram\n");
         printf("--mirror, -m: function of mirroring the area on
picture\n");
         printf("-axis, -a: the axis of mirroring\n");
         printf("--left_up, L: left_up coords of area to mirror\n");
         printf("--right_down, -r: right_down coords of area to
mirror\n");
         printf("--pentagram, -p: function which draws pentagram in
circle\n");
         printf("--center, -C: coords(xc, yc) of circle\n");
        printf("--radius, -R: radius of circle\n");
    }

    void printFileHeader(BitmapFileHeader header)
    {
        printf("signature:\t%x       (%hu)\n",       header.signature,
header.signature);
        printf("filesize:\t%x        (%u)\n",        header.filesize,
header.filesize);
        printf("reserved1:\t%x       (%hu)\n",       header.reserved1,
header.reserved1);
        printf("reserved2:\t%x       (%hu)\n",       header.reserved2,
header.reserved2);
        printf("pixelArrOffset:\t%x  (%u)\n",  header.pixelArrOffset,
header.pixelArrOffset);
    }

    void printInfoHeader(BitmapInfoHeader header)
    {
        printf("headerSize:\t%x      (%u)\n",      header.headerSize,
header.headerSize);
        printf("width:     \t%x (%u)\n", header.width, header.width);
        printf("height:   \t%x (%u)\n", header.height, header.height);
        printf("planes:            \t%x   (%hu)\n",   header.planes,
header.planes);
```

```
        printf("bitsPerPixel:\t%x    (%hu)\n",    header.bitsPerPixel,
header.bitsPerPixel);
        printf("compression:\t%x     (%u)\n",     header.compression,
header.compression);
        printf("imageSize:\t%x       (%u)\n",       header.imageSize,
header.imageSize);
        printf("xPixelsPerMeter:\t%x (%u)\n", header.xPixelsPerMeter,
header.xPixelsPerMeter);
        printf("yPixelsPerMeter:\t%x (%u)\n", header.yPixelsPerMeter,
header.yPixelsPerMeter);
        printf("colorsInColorTable:\t%x                        (%u)\n",
header.colorsInColorTable, header.colorsInColorTable);
        printf("importantColorCount:\t%x                       (%u)\n",
header.importantColorCount, header.importantColorCount);
    }

    int check_bmp(BitmapFileHeader bmfh, BitmapInfoHeader bmif){
        if(bmfh.signature  !=  0x4D42  ||  bmif.headerSize  !=  40  ||
bmif.bitsPerPixel != 24 || bmif.compression != 0){
            printf("Error: file has not BMP format\n");
            return 0;
        }
        return 1;
    }

    void  read_bitmap(const  char  *input_filename,  BitmapFileHeader
*bmfh, BitmapInfoHeader *bmif, Rgb ***arr){
        FILE *f = fopen(input_filename, "rb");
        if (f == NULL){
            perror("Error opening file");
            exit(40);
        }

        fread(bmfh, sizeof(BitmapFileHeader), 1, f);
        fread(bmif, sizeof(BitmapInfoHeader), 1, f);

        if(!check_bmp(*bmfh, *bmif)){
            exit(40);
        }

        unsigned int H = bmif->height;
        unsigned int W = bmif->width;

        *arr = malloc(H * sizeof(Rgb *));
        if(arr == NULL){
              printf("Memory error\n");
              exit(42);
         }
        for (int i = H-1; i >= 0; i--){
            (*arr)[i]  =  malloc(W  *  sizeof(Rgb)  +  (4  -  (W  *
sizeof(Rgb)) % 4) % 4);
            if((*arr)[i] == NULL){
                  printf("Memory error\n");
                  exit(42);
             }
            fread((*arr)[i],  1,  W  *  sizeof(Rgb)  +  (4-  (W  *
sizeof(Rgb)) % 4) % 4, f);
```

```
    }

    fclose(f);
}

void write_bitmap(const char *output_filename, BitmapFileHeader
bmfh, BitmapInfoHeader bmif, Rgb ***arr, int H, int W){
    FILE *ff = fopen(output_filename, "wb");
    if (ff == NULL)
    {
        perror("Error opening output file");
        exit(40);
    }

    bmif.height = H;
    bmif.width = W;
    fwrite(&bmfh, sizeof(BitmapFileHeader), 1, ff);
    fwrite(&bmif, sizeof(BitmapInfoHeader), 1, ff);

    int padding = (4 - (W * 3) % 4) % 4;
     uint8_t padding_bytes[3] = { 0 };
    for (int i = H-1; i >= 0; i--)
    {
        fwrite((*arr)[i], sizeof(Rgb), W, ff);
           fwrite(padding_bytes, sizeof(uint8_t), padding, ff);
    }

    if(!check_bmp(bmfh, bmif)){
        exit(40);
    }

    fclose(ff);
}

int get_number(char* str){
    if(strlen(str) == 1 && str[0] == '-'){
        printf("Error, wrong arg");
        exit(40);
    }
    for(int i = 0; i <strlen(str); i++){
    if(isdigit(str[i]) == 0 && str[i] != '-'){
            printf("Error, wrong arg");
            exit(40);
        }
    }
    int tmp = atoi(str);
    return tmp;
}

int* pars_args(char* str, int c_args){
    int* args = malloc(c_args+1 * sizeof(int));
    if(args == NULL){
        printf("Memory error\n");
           exit(42);
    }
    char* token;
    int i = 0;
```

```
    size_t str_len = strlen(str);
    char *str_copy = malloc((str_len + 1) * sizeof(char));
    if(str_copy == NULL){
        printf("Memory error\n");
           exit(42);
    }
    strcpy(str_copy, str);

    token = strtok(str_copy, ".");

    while(token != NULL && i < c_args){
        args[i++] = get_number(token);
        token = strtok(NULL, ".");
    }
    free(str_copy);
    if(i != c_args){
        printf("Error: wrong args format");
        exit(40);
    }
    return args;
}

Rgb pars_color(char* str, int c_args){
    Rgb color;
    int i = 0;
    char* token;
    size_t str_len = strlen(str);
    char *str_copy = malloc((str_len + 1) * sizeof(char));
    if(str_copy == NULL){
        printf("Memory error\n");
           exit(42);
    }
    strcpy(str_copy, str);

    token = strtok(str_copy, ".");
    while(token != NULL && i < c_args){
        switch(i){
            case 0: {
                int r = get_number(token);
                if(r > 255 || r < 0){
                    printf("Error, wrong color");
                    exit(40);
                }
                color.r = r;
                break;
            }
            case 1: {
                int g = get_number(token);
                if(g > 255 || g < 0){
                    printf("Error, wrong color");
                    exit(40);
                }
                color.g = g;
                break;
            }
            case 2: {
                int b = get_number(token);
```

```
                    if(b > 255 || b < 0){
                        printf("Error, wrong color");
                        exit(40);
                    }
                    color.b = b;
                    break;
                }
            }
            i++;
            token = strtok(NULL, ".");
        }
        free(str_copy);
        if(i != c_args){
            printf("Error: wrong args format");
            exit(40);
        }
        return color;
    }

    //ФУНКИЯ 1(ОТРЕЗОК)
    void check_borders_circ(int* xmax, int* xmin, int* ymax, int* ymin,
int H, int W){
        if (*xmin < 0) *xmin = 0;
        if (*ymin < 0) *ymin = 0;
        if (*xmax >= W) *xmax = W - 1;
        if (*ymax >= H) *ymax = H - 1;
    }

    void fill_circle(Rgb **arr, int xc, int yc, int radius, Rgb color,
int H, int W){
        int xmin = xc - radius;
        int xmax = xc + radius;
        int ymin = yc - radius;
        int ymax = yc + radius;

        check_borders_circ(&xmax, &xmin, &ymax, &ymin, H, W);

        for (int y = ymin; y <= ymax; y++) {
            for (int x = xmin; x <= xmax; x++) {
                if ((x - xc)*(x - xc) + (y - yc)*(y - yc) <=
radius*radius){
                    if(x >= 0 && y >= 0 && x < W && y < H){
                        arr[y][x] = color;
                    }
                }
            }
        }
    }

    void draw_line(Rgb **arr, int H, int W, int x0, int x1, int y0,
int y1, Rgb color, int thickness) {
        int dx = abs(x1 - x0);
        int dy = abs(y1 - y0);
        int sx = x0 < x1 ? 1 : -1;
        int sy = y0 < y1 ? 1 : -1;
        int err = dx - dy;
        while (1) {
```

```
            if (x0 >= 0 && x0 < W && y0 >= 0 && y0 < H) {
                if (thickness == 1) arr[y0][x0] = color;
                          arr[y0][x0] = color;
            }
            if(thickness > 1 && x0 - (thickness/2) < W && y0 -
(thickness/2) < H && x0 + (thickness/2) >= 0 && y0 + (thickness/2) >=
0){
                fill_circle(arr, x0, y0, thickness / 2, color, H, W);
            }

            if (x0 == x1 && y0 == y1) {
                break;
            }

            int e2 = 2 * err;
            if (e2 > -dy) {
                err -= dy;
                x0 += sx;
            }
            if (e2 < dx) {
                err += dx;
                y0 += sy;
            }
        }
    }

    //2
    void x_mirror(Rgb **arr, int H, int W, int xlup, int ylup, int xrd,
int yrd, int y_mid, int x_mid){
        int y_copy = yrd;
        int x_copy = xlup;

        for(int i = ylup; i <= y_mid; i++){
            for(int j = xlup; j <= xrd; j++){
                Rgb tmp = arr[i][j];
                arr[i][j] = arr[y_copy][x_copy];
                arr[y_copy][x_copy] = tmp;
                x_copy++;
            }
            y_copy--;
            x_copy = xlup;
        }
    }

    void y_mirror(Rgb **arr, int H, int W, int xlup, int ylup, int xrd,
int yrd, int y_mid, int x_mid){
        int y_copy = ylup;
        int x_copy = xrd;

        for(int i = ylup; i <= yrd; i++){
            for(int j = xlup; j <= x_mid; j++){
                Rgb tmp = arr[i][j];
                arr[i][j] = arr[y_copy][x_copy];
                arr[y_copy][x_copy] = tmp;
                x_copy--;
            }
            y_copy++;
```

```
            x_copy = xrd;
        }
    }

    void check_borders_mirror(int H, int W, int *xlup, int *ylup, int
*xrd, int *yrd){
        if(*xlup < 0){
            *xlup = 0;
        }
        else if(*xlup >= W){
            *xlup = W-1;
        }
        if(*xrd < 0){
            *xrd = 0;
        }
        else if(*xrd >= W){
            *xrd = W;
        }
        if(*ylup < 0){
            *ylup = 0;
        }
        else if(*ylup >= H){
            *ylup = H-1;
        }
        if(*yrd < 0){
            *yrd = 0;
        }
        else if(*yrd >= H){
            *yrd = H-1;
        }
    }

    void mirror_picture(Rgb **arr, int H, int W, char* axis, int xlup,
int ylup, int xrd, int yrd){
        check_borders_mirror(H, W, &xlup, &ylup, &xrd, &yrd);
        int y_mid = (ylup + yrd) / 2;
        int x_mid = (xlup + xrd) / 2;

        if(strcmp(axis, "y")) {
            y_mirror(arr, H, W, xlup, ylup, xrd, yrd, y_mid, x_mid);
        } else if(strcmp(axis,"x")) {
            x_mirror(arr, H, W, xlup, ylup, xrd, yrd, y_mid, x_mid);
        } else {
            printf("Error: Axis doesn`t exist");
            exit(40);
        }
    }

    //3
    void find_vertices(int center[2], int radius, int vertices[5][2])
{
        double angle = 0.0;
        double angle_increment = 2 * PI / 5;
        for (int i = 0; i < 5; i++){
            vertices[i][1] = center[1] - (int)(radius * cos(angle));
            vertices[i][0] = center[0] + (int)(radius * sin(angle));
            angle += angle_increment;
```

```
        }
    }

    void draw_star(Rgb **arr, int xc, int yc, int r, int H, int W, Rgb
color, int thickness){
        int center[2] = {xc, yc};
        int vertices[5][2];
        find_vertices(center, r, vertices);
        draw_line(arr,   H,   W,   vertices[0][0],   vertices[2][0],
vertices[0][1], vertices[2][1], color, thickness);
        draw_line(arr,   H,   W,   vertices[2][0],   vertices[4][0],
vertices[2][1], vertices[4][1], color, thickness);
        draw_line(arr,   H,   W,   vertices[4][0],   vertices[1][0],
vertices[4][1], vertices[1][1], color, thickness);
        draw_line(arr,   H,   W,   vertices[1][0],   vertices[3][0],
vertices[1][1], vertices[3][1], color, thickness);
        draw_line(arr,   H,   W,   vertices[3][0],   vertices[0][0],
vertices[3][1], vertices[0][1], color, thickness);
    }

    void draw_circle(Rgb **arr, int xc, int yc, int R, int r, Rgb
color, int H, int W){
        int xmin = xc - R;
        int xmax = xc + R;
        int ymin = yc - R;
        int ymax = yc + R;

        check_borders_circ(&xmax, &xmin, &ymax, &ymin, H, W);

        for (int y = ymin; y <= ymax; y++) {
            for (int x = xmin; x <= xmax; x++) {
                int is_inside_big =
                    (x - xc)*(x - xc) + (y - yc)*(y - yc) <= R*R;
                int is_outside_small =
                    (x - xc)*(x - xc) + (y - yc)*(y - yc) >= r*r;

                if (is_inside_big && is_outside_small){
                    if(x >= 0 && y >= 0 && x < W && y < H){
                        arr[y][x] = color;
                    }
                }
            }
        }
    }

    void draw_pentagram(Rgb **arr, int H, int W, int xc, int yc, int
radius, Rgb color, int thickness) {
        draw_circle(arr, xc, yc, radius + thickness/2, radius -
thickness/2, color, H, W);
        draw_star(arr, xc, yc, radius, H, W, color, thickness);
    }


    void romb(Rgb** arr, int H, int W, Rgb color, int x0, int y0, int
size){
        int diagonal = 0;
        diagonal = (int) (sqrt(size*size + size*size));
```

```
            int rx = x0;
            int ry = y0+diagonal/2;
            while(diagonal >= 0){
                for(int y = 0; y < H; y++){
                    for(int x = 0; x < W; x++){
                        if((abs(x-rx) + abs(y-ry)) == diagonal/2){
                            if(x >= 0 && x < W && y >= 0 && y < H){
                                arr[y][x] = color;
                            }
                        }
                    }
                }
                diagonal--;
            }
        }

        int main(int argc, char* argv[]){
            const             char*             short_options             =
"hIi:o:ls:e:c:t:ma:L:r:pC:R:HU:F:S:";
            const struct option long_options[] = {
                {"help", no_argument, NULL, 'h'},
                {"info", no_argument, NULL, 'I'},
                {"input", required_argument, NULL, 'i'},
                {"output", required_argument, NULL, 'o'},
                {"line", no_argument, NULL, 'l'},
                {"start", required_argument, NULL, 's'},
                {"end", required_argument, NULL, 'e'},
                {"color", required_argument, NULL, 'c'},
                {"thickness", required_argument, NULL, 't'},
                {"mirror", no_argument, NULL, 'm'},
                {"axis", required_argument, NULL, 'a'},
                {"left_up", required_argument, NULL, 'L'},
                {"right_down", required_argument, NULL, 'r'},
                {"pentagram", no_argument, NULL, 'p'},
                {"center", required_argument, NULL, 'C'},
                {"radius", required_argument, NULL, 'R'},
                {"square_rhombus", no_argument, NULL, 'H'},
                {"upper_vertex", required_argument, NULL, 'U'},
                {"fill_color", required_argument, NULL, 'F'},
                {"size", required_argument, NULL, 'S'}
            };

             int opt;
             int option_index;
            int option = -1;
            int print_file_info = 0;

            char* start = NULL;
            char* end = NULL;
            char* vertex = NULL;
            char* fill_color = NULL;
            char* color = NULL;
            int thickness = 0;
            char* axis = NULL;
            char* left_up = NULL;
            char* right_down = NULL;
            char* center = NULL;
```

```
        int radius = 0;
        int size = 0;
        char* output_filename = NULL;
        char* input_filename = NULL;

        while((opt=getopt_long(argc,argv,short_options,  long_options,
&option_index))!=-1){
            switch(opt){
                case 'h': {
                    print_author_info();
                    p_help();
                    exit(0);
                    break;
                };
                case 'I': {
                    print_file_info = 1;
                    break;
                };
                case 'i': {
                    input_filename = optarg;
                    break;
                };
                case 'o': {
                    output_filename = optarg;
                    break;
                };
                case 'l': {
                    option = 1;
                    break;
                };
                case 's': {
                    start = optarg;
                    break;
                };
                case 'e': {
                    end = optarg;
                    break;
                };
                case 'c': {
                    color = optarg;
                    break;
                };
                case 't': {
                    thickness = get_number(optarg);
                    if(thickness <= 0){
                        printf("Error: wrong thickness");
                        exit(40);
                    }
                    break;
                };
                case 'm': {
                    option = 2;
                    break;
                };
                case 'a': {
                    axis = optarg;
                    break;
```

```
        };
        case 'L': {
            left_up = optarg;
            break;
        };
        case 'r': {
            right_down = optarg;
            break;
        };
        case 'p': {
            option = 3;
            break;
        };
        case 'C': {
            center = optarg;
            break;
        };
        case 'R': {
            radius = get_number(optarg);
            if(radius <= 0){
                printf("Error, wrong radius");
                exit(40);
            }
            break;
        };
        case 'H': {
            option = 4;
            break;
        }
        case 'U': {
            vertex = optarg;
            break;
        }
        case 'F': {
            fill_color = optarg;
            break;
        }
        case 'S': {
            size = get_number(optarg);
            break;
        }
        case '?': {
            printf("unknown option\n");
            exit(0);
            break;
        };
    }
}

BitmapFileHeader bmfh;
BitmapInfoHeader bmif;
Rgb **arr;

read_bitmap(input_filename, &bmfh, &bmif, &arr);

if(print_file_info == 1){
    printFileHeader(bmfh);
```

```
            printInfoHeader(bmif);
            exit(0);
        }

        unsigned int H = bmif.height;
        unsigned int W = bmif.width;

        switch (option){
            case 1: {
                int* starts = pars_args(start, 2);
                int* ends = pars_args(end, 2);
                Rgb parsed_color = pars_color(color, 3);
                draw_line(arr, H, W, starts[0], ends[0], starts[1],
ends[1], parsed_color, thickness);
                free(starts);
                free(ends);
                break;
            }
            case 2: {
                int* left_ups = pars_args(left_up, 2);
                int* right_downs = pars_args(right_down, 2);
                mirror_picture(arr,   H,   W,   axis,   left_ups[0],
left_ups[1], right_downs[0], right_downs[1]);
                free(left_ups);
                free(right_downs);
                break;
            }
            case 3: {
                int* centers = pars_args(center, 2);
                Rgb parsed_color = pars_color(color, 3);
                draw_pentagram(arr,  H,  W,  centers[0],  centers[1],
radius, parsed_color, thickness);
                free(centers);
                break;
            }
            case 4: {
                int* u_vertex = pars_args(vertex, 2);
                Rgb parsed_color = pars_color(fill_color, 3);
                romb(arr,   H,   W,   parsed_color,   u_vertex[0],
u_vertex[1], size);
                free(u_vertex);
                break;
            }
            default:
                break;
        }

        write_bitmap(output_filename, bmfh, bmif, &arr, H, W);

        for (int i = 0; i < H; i++) {
            free(arr[i]);
        }
        free(arr);

        return 0;
    }
```