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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

ТЕМА: РАБОТА С ИЗОБРАЖЕНИЯМИ

Студент гр. 3342 _____ Песчатский С. Д.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Песчатский С. Д.

Группа 3342

Тема работы: Работа с изображениями

Вариант 5.9

Программа обязательно должна иметь CLI (опционально дополнительное использование GUI). Более подробно тут: http://se.moevm.info/doku.php/courses:programming:rules_extra_kurs

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:

Программа должна реализовывать весь следующий функционал по

обработке bmp-файла

Рисование квадрата с диагоналями. Флаг для выполнения данной операции: `--squared_lines`. Квадрат определяется:

Координатами левого верхнего угла. Флаг `--left_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y

Размером стороны. Флаг `--side_size`. На вход принимает число больше 0

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)

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`)

Фильтр rgb-компонент. Флаг для выполнения данной операции: `--rgbfilter`. Этот инструмент должен позволять для всего изображения либо установить в диапазоне от 0 до 255 значение заданной компоненты. Функционал определяется

Какую компоненту требуется изменить. Флаг `--component_name`. Возможные значения `red`, `green` и `blue`.

В какой значение ее требуется изменить. Флаг `--component_value`. Принимает значение в виде числа от 0 до 255

Поворот изображения (части) на 90/180/270 градусов. Флаг для выполнения данной операции: `—rotate`. Функционал определяется:

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

Углом поворота. Флаг `--angle`, возможные значения: `90`, `180`, `270`

Рисование окружности. Флаг для выполнения данной операции: `--circle`. Окружность определяется:

Координатами её центра. Флаг `--center` , значение задаётся в формате `left.up`, где left – координата по x, up – координата по y

Радиусом. Флаг `--radius`, на вход принимается число больше 0

Толщиной линии окружности. Флаг `--thickness`, на вход принимается число больше 0

Цветом линии окружности. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb - числа, задающие цветовую компоненту. Пример `--color 255.0.0`)

Окружность может быть залита или нет. Флаг `--fill`. Работает как бинарное значение: флага нет – false , флаг есть – true.

Цветом которым залита сама окружность, если пользователем выбран залитый. Флаг `--fill_color` (работает аналогично флагу `--color`)

Каждую подзадачу следует вынести в отдельную функцию, функции сгруппировать в несколько файлов (например, функции обработки текста в один, функции ввода/вывода в другой). Сборка должна осуществляться при помощи make и Makefile или другой системы сборки

Содержание пояснительной записки:

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

Дата выдачи задания: 18.03.2024

Дата сдачи реферата: 22.05.2024

Дата защиты реферата: 22.05.2024

Студент _____ Песчатский С. Д.

Преподаватель _____ Глазунов С.А.

АННОТАЦИЯ

Курсовая работа представляет собой программу, реализующую CLI и обрабатывающую изображение формата BMP в соответствии с переданными пользователем в неё опциями (рисование квадрата, фильтр RGB-компонент, поворот изображения (или его части), рисование окружности). Для выполнения этой задачи программа использует функции различных библиотек языка Си, в том числе “getopt.h” для реализации CLI. Выполнив необходимые преобразования создаётся файл с изменённым изображением.

SUMMARY

The course work is a program that implements the CLI and processes a BMP image in accordance with the options passed to it by the user (drawing a square, RGB component filter, rotating the image (or part of it), drawing a circle). To accomplish this task, the program uses the functions of various C language libraries, including "getopt.h” for the CLI implementation. After completing the necessary transformations, a file with the modified image is created.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является написание программы, осуществляющую обработку изображения в соответствии с опциями, введёнными пользователем. Для этого требуется:

- Создать функции, реализующие CLI.
- Создать функции, обрабатывающие изображение в соответствии с выбором пользователя.
- Написать Makefile, с помощью которого будет реализована сборка программы.

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, реализующие CLI

Функция check (int check(char *str)) проверяет, является ли переданная в функцию последовательность символов правильной записью цвета.

Функция smalcheck(int smalcheck(char *str)) проверяет, является ли переданная в функцию последовательность символов правильной записью компоненты цвета.

2.2. Функции первого задания

Функция square(void square(int cordx, int cordy, int side, int thick, int *fill, FILE *fptr, BitmapFileHeader *bmfh, BitmapInfoHeader *bmfh, Rgb **arr, int *color, char *output)) принимает на вход координаты верхнего левого угла квадрата, длину стороны, толщину линий, цвет закрашивания, указатель на файл, указатели на BitmapFileHeader и BitmapInfoHeader, массив пикселей, цвет линий и названия выходного файла. Попарядку вызываются функции для рисования отдельных линий, скругление углов, закрашивание квадрата, рисование диагоналей и вывод конечного результата.

Функция hor_line(void hor_line(Rgb **arr, int h, int w, int x0, int y0, int x1, int thickness, int* color)) принимает массив пикселей, высоту и ширину изображения, координаты начала и конца линии(верхний левый и правый нижний углы линии), толщину линии и цвет линии. Закрашивает все пиксели в необходимой области, смещая позицию линии в зависимости от толщины.

Функция vert_line(void vert_line(Rgb **arr, int h, int w, int x0, int y0,int y1, int thickness, int* color)) принимает массив пикселей, высоту и ширину изображения, координаты начала и конца линии(верхний левый и правый нижний углы линии), толщину линии и цвет линии. Закрашивает все пиксели в необходимой области, смещая позицию линии в зависимости от толщины.

Функция filler(int filler(Rgb **arr, int h, int w, int x0, int y0, int x1, int y1, int thickness, int *color)) принимает массив пикселей, высоту и ширину изображения, координаты левого верхнего и правого нижнего углов, лежищ

строго внутри квадрата, толщину линии и цвет заливки. Закрашивает все пиксели в необходимой области, если есть флаг `--fill`.

Функция diagonal(void diagonal(Rgb **arr, int h, int w, int x0,int y0, int x1, int side, int thickness, int *color))принимает массив пикселей, высоту и ширину изображения, координаты начала и конца линии(верхний левый и правый нижний углы линии), толщину линии и цвет линии. Работает схоже функции hor_line, но смещает значение y для каждого последующего значения x.

Функция cicle(cicle(int h, int w, Rgb**arr, int *xy, int rad, int thick, int *filcol, int *color)) принимает высоту и ширину изображения, массив пикселей, координаты угла, радиус, толщину линии, цвет линии и цвет заполнеия. Аргументы радиус и толщина, цвет линии и цвет заполнения дублируют друг друга. Рисует окружность в одном из углов квадрата.

2.3 Функции второго задания

Функция filter(void filter(BitmapFileHeader * bmfh, BitmapInfoHeader *bmih, Rgb **arr, char * name, int val, char *out)) принимает указатели на BitmapFileHeader и BitmapInfoHeader, массив пикселей, название компоненты которую нужно изменить, значение на которое нужно изменить и название выходного файла. Функция проходит по всем пикселям изображения и меняет в каждом из них компоненту, в зависимости от полученных значений.

2.4 Функции третьего задания

Функция rotate(void rotate(BitmapFileHeader * bmfh, BitmapInfoHeader *bmih, Rgb **arr, int *xy0, int *xy1, int deg, char *out)) принимает указатели на BitmapFileHeader и BitmapInfoHeader, массив пикселей, координаты левого верхнего угла и координаты прваго нижнего угла, градус поворота и название выходного файла. Сначала область которую нужно повернуть копируется в отдельный массив пикселей, затем в зависимости от угла поворота значения из нового массива переносятся в старый массив, одновременно с поворотом и смещением части изображения. Смещение необходимо чтобы поворот происходил относительно центра изначально выделенной области.

2.5 Функции четвертого задания

Функция circle(void circle(BitmapFileHeader* bmfh, BitmapInfoHeader* bmif, Rgb**arr, int *xy, int rad, int thick, int * filcol, int *color, char *out)) принимает указатели на BitmapFileHeader и BitmapInfoHeader, массив пикселей, координаты центра окружности, радиус, толщину линии, цвет заливки, цвет линии и название выходного файла. Используя уравнение окружности сначала происходит заливка необходимой области, а затем тем же методом рисуется линия окружности.

2.6 Функции работы с файлами

Функция readBMP (RGB ** readBMP(char * file_name, BitmapFileHeader * bmfh, BitmapInfoHeader * bmih)) считывает переданный файл с структуры BitmapFileHeader и BitmapInfoHeader, а также в двумерный массив пикселей.

Функция checkBMP (int checkBMP(BitmapFileHeader * bmfh, BitmapInfoHeader * bmih)) проверяет соответствие ожидаемому формату файла.

Функция writeBMP (void writeBMP(char * filename, RGB ** arr, unsigned int H, unsigned int W, BitmapFileHeader * bmfh, BitmapInfoHeader * bmih)) записывает в новый файл все данные, полученные после обработки изображения.

2.7 Прочие функции

Функция fotm(void fotm(int argc, char *const args[])) принимает количество опций и массив опций. Проводит все операции, связанные с библиотекой getopt и считыванием поступающих данных, чтение входного файла и вызов необходимой функции

2.8 Функция main

Функция int main() вывдит строчку о работе и вызывает функцию fotm

Примеры работы программы см. в приложении А.

Разработанный программный код см. в приложении В.

3. СБОРКА ПРОГРАММЫ

Для сборки программы использовался Makefile, в котором компилируются все исходные файлы и линкуются. Программа собирается в файл “cw”.

Также есть цель clean, которая удаляет все объектные файлы.

ЗАКЛЮЧЕНИЕ

Курсовая работа включала в себя систематизацию знаний об изображениях и способах их обработки, изучение и применение функций стандартной библиотеки языка C.

В процессе работы были использованы структуры для хранения заголовков файлов, различных опций. Были изучены особенности языка, связанные с обработкой изображений и реализацией CLI. С помощью стандартной библиотеки были реализованы основные функции чтения и обработки файлов, представляющих собой BMP изображение.

В итоге готовый программный код выполняет все поставленные перед ним задачи, а именно, обрабатывает изображение, выводит сообщения о возникающих ошибках, задействует функции стандартной библиотеки языка C, использует структуры для хранения информации.

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 C / Керниган Брайан, Ритчи Деннис. СПб.: "Финансы и статистика", 2003.

ПРИЛОЖЕНИЕ А
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

ПРИМЕР 1 – вывод справки.

```
sepesch@sepesch-Pulse-GL66-12UDK:~/educ/proga/sem2/cw/Peschatskii_Semen_cw/src$ ./cw -h
Course work for option 5.9, created by Semen Peschatskii
Each execution of programm requires file name of bmp format that will be change. Default name of output file is out.bmp(can be changed with --output). (req) mean option is required for corre
ct work of program.
--squared_lines draws a square with set cordinates, side length, side thickness, color and optionaly fillament. Next options are available:
 --left_up (req) sets cordinates of upper left corner(argument format: (num))
 --side_size (req) sets length of sides in pixels(argument format:(num))
 --thickness (req) sets thickness of all lines in pixels(argument format: (num))
 --color (req) sets color of lines(argument format: (num).(num).(num) each number must be in 0-255 interval)
 --fill determines if square is filled or not, requires --fill_color after(argument format: no argument))
 --fill_color (req if --fill is present) sets color of filler pixels(argument format: (num).(num).(num) each number must be in 0-255 interval)

--rgbfilter changes changes red, green, or blue of each pixel to set value. Next options available:
 --component_name (req) gets name of component that will be changed(argument format:red or green or blue))
 --component_val (req) value that each pixel's set component will be changed to(argument format:(num) must be in 0-255 interval)

--rotate rotates certain part of picture(segment of picture is always a square). Next options are available:
 --left_up (req) sets cordinates of upper left corner(argument format:(num))
 --right_down (req) sets cordinates of lower right corner(argument format:(num))
 --angle (req) sets angle that segemnt will be rotate for(argument format:90 or 180 or 270)

--circle draws a circle. Next options are available:
 --center (req) sets cords of center of the circle(argument format:(num).(num))
 --radius sets raduis in pixels(argument format:(num))
 --thickness determines thickness of lines(argument format:(num))
 --color (req) sets color of circle(argument format:(num).(num).(num) each num is 0-255 interval)
 --fill if present circle will be filled withc certain color, requires --fill_color after(argument format:no argument)
 --fill_color (req if --fill is present) sets filler color(argument format:(num).(num).(num) each num is in 0-255 interval)
```

ПРИМЕР 2 – вывод информации о файле.

```
sepesch@sepesch-Pulse-GL66-12UDK:~/educ/proga/sem2/cw/Peschatskii_Semen_cw/src$ ./cw --input input.bmp --info
Course work for option 5.9, created by Semen Peschatskii
---info---
Width: 1280
 Height: 853
 bitsPerPixel: 24
```

ПРИМЕР 3 – Рисование квадрата с диагоналями.

Параметры запуска: ./cw --squared_lines --left_up 200.200 --side_size 150 --color 0.0.255 --thickness 8 --fill --fill_color 0.155.0 --input ./input.bmp --output ./out.bmp

ПРИМЕР 4 – фильтр RGB компонент.

Параметры запуска: ./cw --rgbfilter --component_name red --component_value 40 --input ./input.bmp --output ./out.bmp


ПРИМЕР 5 – Поворот части изображения.

Параметры запуска: ./cw --circle --center 500.500 --radius 300 --color 128.128.0 --fill --fill_color 73.100.144 --thickness 35 --input ./input.bmp --output ./out.bmp


ПРИМЕР 6– Рисование круга.

Параметры запуска: ./cw --circle --center 500.500 --radius 300 --color 128.128.0 --fill --fill_color 73.100.144 --thickness 35 --input ./input.bmp --output ./out.bmp


ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: structures.h

```
typedef struct{
        unsigned short signa;
        unsigned int filesize;
        unsigned short reserved1;
        unsigned short reserved2;
        unsigned int pixelArrOffset;
}__attribute__ ((packed))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;
}__attribute__ ((packed))BitmapInfoHeader;
typedef struct Rgb{
        unsigned char b;
        unsigned char g;
        unsigned char r;
}__attribute__ ((packed))Rgb;

Rgb     **read_bmp(FILE*     fptr,     BitmapFileHeader*     bmfh,
BitmapInfoHeader* bmif);
void write_bmp(Rgb **arr, int h, int w, BitmapFileHeader bmfh,
BitmapInfoHeader bmif, char* out);
```

Название файла: structures.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#ifndef bfh
#define bfh
typedef struct {
        unsigned short signa;
        unsigned int filesize;
        unsigned short reserved1;
        unsigned short reserved2;
        unsigned int pixelArrOffset;
}__attribute__ ((packed)) BitmapFileHeader;
#endif
#ifndef bih
#define bih
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;
    }__attribute__ ((packed)) BitmapInfoHeader;
    #endif
    #ifndef rog
    #define rog
    typedef struct{
        unsigned char b;
        unsigned char g;
        unsigned char r;
    }__attribute__((packed))Rgb;
    #endif
    Rgb     **read_bmp(FILE     *fptr,     BitmapFileHeader*     bmfh,
BitmapInfoHeader* bmif){
        int w=bmif->width;
        int h=bmif->height;
        Rgb **arr = malloc(h * sizeof(Rgb*));
        if(arr==NULL){printf("Memory error\n"); exit(49);}
        for(int i = 0; i<h; i++){
            arr[i]      =      malloc(w*sizeof(Rgb)      +      (4-
(w*sizeof(Rgb))%4)%4);
            if(arr[i]==NULL){printf("Memory error\n"); exit(49);}
            fread(arr[i],      1,      w*sizeof(Rgb)      +      (4-
(w*sizeof(Rgb))%4)%4, fptr);
        }
        fclose(fptr);
        return arr;
    }
    void  write_bmp(Rgb  **arr,  int  h,  int  w,  BitmapFileHeader  bmfh,
BitmapInfoHeader bmif, char* out){
        FILE *output = fopen(out, "wb");
        fwrite(&bmfh, 1, sizeof(BitmapFileHeader), output);
        fwrite(&bmif, 1, sizeof(BitmapInfoHeader), output);
        int padding = (4-(w*3)%4)%4;
        uint8_t paddingBytes[3] = { 0 };
        for(int i = 0; i<h; i++){
            fwrite(arr[i], sizeof(Rgb), w, output);
            fwrite(paddingBytes, sizeof(uint8_t), padding, output);
        }
        free(arr);
        fclose(output);
    }
```

Название файла: check.h

```
    int check(char *str);
```

Название файла: check.c

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

```
int check(char *str){
      char *fir=strtok(str, ".");
      char *sec=strtok(NULL, ".");
      char *thi=strtok(NULL, ".");
      int flag =1;
      if(strlen(fir)<4 && strlen(sec)<4 && strlen(thi)<4){
            if(flag){
                  for(int i=0; i<strlen(fir); i++){if(fir[i]>'9' ||
fir[i]<'0'){flag=0;}}
            }
                  if(flag){
                    for(int  i=0;  i<strlen(sec);  i++){if(sec[i]>'9'
|| sec[i]<'0'){flag=0;}}
                  }
                  if(flag){
                    for(int  i=0;  i<strlen(thi);  i++){if(thi[i]>'9'
|| thi[i]<'0'){flag=0;}}
                  }
      }
      if(flag){
            int a=atoi(fir);
            int b=atoi(sec);
            int c=atoi(thi);
            if(a>255 || b>255 || c>255){flag=0;}
      }
return flag;
}
```

Название файла: libs.h

```
#ifndef CW_LIBS_H
#define CW_LIBS_H

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

#endif //CW_LIBS_H
```

Название файла: square.h

```
void swap(int *a, int *b);
void vert_line(Rgb **arr, int h, int w, int x0, int y0, int x1, int
y1, int thickness, int* color);
void vert_line(Rgb **arr, int h, int w, int x0, int y0, int x1, int
y1, int thickness, int* color);
void vert_line(Rgb **arr, int h, int w, int x0, int y0, int x1, int
y1, int thickness, int* color);
void vert_line(Rgb **arr, int h, int w, int x0, int y0, int x1, int
y1, int thickness, int* color);
void   cicle(BitmapFileHeader*   bmfh,   BitmapInfoHeader*   bmif,
Rgb**arr, int *xy, int rad, int thick, int*filcol, int*color);
void square(int cordx, int cordy, int side, int thick, int *fill,
FILE * fptr, BitmapFileHeader  *bmfh, BitmapInfoHeader  *bmih, Rgb **arr,
int *color, char* output);
```

Название файла: square.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "structures.h"
void swap(int  *a, int *b){
	int t = *a;
	*a=*b;
	*b=t;
}
void vert_line(Rgb **arr, int h, int w, int x0, int y0,int y1, int thickness, int* color){
	for(int y = y0; y < y1; y++){
		for (int j = -thickness/2; j < thickness/2+1; j++){
				if(y>0 && y<=h && x0-j>=0 && x0-j<w){
				arr[h-y][x0-j].r=*(color);
				arr[h-y][x0-j].g=*(color + 1);
				arr[h-y][x0-j].b=*(color + 2);
			}
		}
	}
}
void hor_line(Rgb **arr, int h, int w, int x0, int y0, int x1, int thickness, int* color){
	for (int x = x0; x < x1; x++){
		for (int j = -thickness/2; j <thickness/2+ 1; j++){
			if(y0-j<=h && y0-j>0 && x<w && x>=0){
				arr[h-y0+j][x].r=*(color);
				arr[h-y0+j][x].g=*(color + 1);
				arr[h-y0+j][x].b=*(color + 2);
			}
		}
	}

}
void filler(Rgb **arr, int h, int w, int x0, int y0, int x1, int y1, int thickness, int *color){
	for(int y = y0; y < y1; y++){
		for(int x = x0; x <x1; x++){
			if(y>0 && y<=h && x>=0 && x<w){
			arr[h-y][x].r=*(color);
			arr[h-y][x].g=*(color+1);
			arr[h-y][x].b=*(color+2);
			}
		}
	}
}
void diagonal(Rgb **arr, int h, int w, int x0,int y0, int x1, int side, int thickness, int *color){
	for(int x=x0; x<x1; x++){
		for(int j=0-thickness/2;j<thickness; j++){
			if(y0-j-(x0-x)<=h  &&  y0-j-(x0-x)>0  &&  x>=0  && x<w){
			arr[h-y0+j+(x0-x)][x].r=*(color);
			arr[h-y0+j+(x0-x)][x].g=*(color+1);
			arr[h-y0+j+(x0-x)][x].b=*(color+2);
			}
```

```
                }
            }
            for(int x=x0; x<x1; x++){
                for(int j=0-thickness/2; j<thickness; j++){
                    if(y0-(x-x0)-j+side-thickness/2<=h && y0-(x-x0)-
j+side-thickness/2>0 && x>=0 && x<w){
                    arr[h-y0+(x-x0)+j-side+thickness/2][x].r=*(color);
                    arr[h-y0+(x-x0)+j-side+thickness/2][x].g=*(color+1);
                    arr[h-y0+(x-x0)+j-side+thickness/2][x].b=*(color+2);
                    }
                }
            }
      }
      void cicle(int h, int w, Rgb**arr, int *xy, int rad, int thick, int
*filcol, int *color){
              for(int i =0; i<w; i++){
                      for(int j=0; j<h; j++){
                        if((j-*xy)*(j-*xy)+(i-xy[1])*(i-xy[1])<=rad*rad
&& (j-*xy)*(j-*xy)+(i-xy[1])*(i-xy[1])>=(rad-thick)*(rad-thick) && i>=0
&& i<h && j>=0 && j<w){
                                        arr[i][j].r=*(color);
                                        arr[i][j].g=*(color+1);
                                        arr[i][j].b=*(color+2);
                                }
                      }
              }
      }

      void square(int cordx, int cordy, int side, int thick, int*
fill,FILE * fptr, BitmapFileHeader * bmfh, BitmapInfoHeader *bmih, Rgb
**arr, int *color, char *output){
            hor_line(arr, bmih->height, bmih->width, cordx, cordy,
cordx+side,thick, color);
            hor_line(arr, bmih->height, bmih->width, cordx, cordy+side,
cordx+side,thick, color);
            vert_line(arr, bmih->height, bmih->width, cordx, cordy,
cordy+side, thick, color);
            vert_line(arr, bmih->height, bmih->width, cordx+side, cordy,
cordy+side,thick, color);
            int          cords[2];cords[0]=cordx;cords[1]=bmih->height-
cordy;cicle(bmih->height, bmih->width, arr, cords, thick/2, thick/2,
color, color);
            cords[0]=cordx;cords[1]=bmih->height-cordy-
side;cicle(bmih->height, bmih->width, arr, cords, thick/2, thick/2,
color, color);
            cords[0]=cordx+side;cords[1]=bmih->height-
cordy;cicle(bmih->height, bmih->width, arr, cords, thick/2, thick/2,
color, color);
            cords[0]=cordx+side;cords[1]=bmih->height-cordy-
side;cicle(bmih->height, bmih->width, arr, cords, thick/2, thick/2,
color, color);
            if(fill[0]!=-1){filler(arr,    bmih->height,    bmih->width,
cordx+thick/2,    cordy+thick/2,    cordx+side-thick/2,    cordy+side-
thick/2,thick, fill);}
```

```
          diagonal(arr,   bmih->height,   bmih->width,   cordx+thick/4,
cordy+thick/2, cordx+side-thick/4,side,thick, color);
          write_bmp(arr,   bmih->height,   bmih->width,   *bmfh,   *bmih,
output);
     };
```

Название файла: filter.h

```
     void filter(BitmapFileHeader * bmfh, BitmapInfoHeader *bmih, Rgb
**arr, char*name, int val, char* out);
```

Название файла: filter.c

```
     #include <string.h>
     #include <stdlib.h>
     #include <stdio.h>
     #include "structures.h"
     void filter(BitmapFileHeader * bmfh, BitmapInfoHeader *bmih, Rgb
**arr, char * name, int val, char *out){
          if(strcmp(name, "red")==0){
               for(int i=0; i<bmih->height; i++){
                    for(int j=0; j<bmih->width; j++){
                         arr[i][j].r=val;
                    }
               }
          }
          if(strcmp(name, "blue")==0){
                    for(int i=0; i<bmih->height; i++){
                            for(int j=0; j<bmih->width; j++){
                         arr[i][j].b=val;
                    }
               }
          }
          if(strcmp(name,"green")==0){
                    for(int i=0; i<bmih->height; i++){
                            for(int j=0; j<bmih->width; j++){
                                    arr[i][j].g=val;
                    }
               }
          }
          write_bmp(arr, bmih->height, bmih->width, *bmfh, *bmih, out);
     }
```

Название файла: rotate.h

```
     void  rotate(BitmapFileHeader  *bmfh,  BitmapInfoHeader  *bmih,  Rgb
**arr, int *xy0, int *xy1, int deg, char *out);
```

Название файла: rotate.c

```
     #include <stdio.h>
     #include <stdlib.h>
     #include <string.h>
     #include "structures.h"
     void rotate(BitmapFileHeader * bmfh, BitmapInfoHeader *bmih, Rgb
**arr, int *xy0, int *xy1, int deg, char *out){
     int h = bmih->height;
     int w = bmih->width;
```

```
      if(*xy0<0){*xy0=0;}if(*xy1<0){*xy1=0;}if(*xy1>w){*xy1=w;}if(*xy0>
w){*xy0=w;}
      if(*(xy0+1)<0){*(xy0+1)=0;}if(*(xy0+1)>h){*(xy0+1)=h;}if(*(xy1+1)
<0){*(xy1+1)=0;}if(*(xy1+1)>h){*(xy1+1)=h;}
      int a = *(xy1+1)-*(xy0+1);
      int c=0;int d=0;
      int b = *xy1 -*xy0;
      if(a>b){c=(a-b)/2;}else{c=-((b-a)/2);}
      d=c;
      if(a>0 && b>0){
              Rgb **new_arr=malloc(a*sizeof(Rgb*));
              if(new_arr==NULL){printf("Memory error\n"); exit(49);}
              for(int i =0; i<a;i++){
                      new_arr[i]=malloc(b*sizeof(Rgb)+(b*3)%4);
                      if(new_arr[i]==NULL){printf("Memory    error\n");
exit(49);}
              }
              for(int i=0; i<a; i++){
                      for(int j =0; j<b; j++){
                        if(*(xy1+1)-i>0  &&  *(xy1+1)-i<=h  &&  *xy0+j>=0
&& *xy0+j<w){
                              new_arr[i][j]=arr[h-*(xy1+1)+i][*xy0+j];
                              }
                      }

              }
              if(deg==90){
                      for(int i=0; i<a;i++){
                              for(int j = 0; j<b; j++){
                                if(*(xy1+1)-j-c>0   &&   *(xy1+1)-j-c<=h
&& *xy1-i+d>=0 && *xy1-i+d<w){
                                      arr[h-*(xy1+1)+j+c][*(xy1)-
i+d-1]=new_arr[i][j];
                                        }
                              }
                      }
              }
              if(deg==180){
                      for(int i =0;i<a; i++){
                              for(int j=0; j<b; j++){
                                      if(*(xy0+1)+i>0  &&  *(xy0+1)+i<=h
&& *xy1-j>=0 && *xy1-j<w){
                                      arr[h-*(xy0+1)-i-1][*(xy1)-
j-1]=new_arr[i][j];
                                        }
                              }
                      }
              }
              if(deg==270){
                      for(int i=0; i<a; i++){
                              for(int j=0; j<b; j++){
                              if(*(xy0+1)+j+c+1>0  &&  *(xy0+1)+j+c+1<=h
&& *xy0+i-d>=0 && *xy0+i-d<w){
                                      arr[h-*(xy0+1)-j-c-1][*(xy0)+i-
d]=new_arr[i][j];
                                        }
                              }
```

```
                    }
                }
        free(new_arr);
        }
        write_bmp(arr, bmih->height, bmih->width, *bmfh, *bmih, out);
        }
```

Название файла: circle.h

```
        void circle(BitmapFileHeader* bmfh, BitmapInfoHeader* bmif, Rgb
**arr, int *xy, int rad, int thick, int *filcol, int *color, char *out);
```

Название файла: circle.c

```
        #include <stdio.h>
        #include <string.h>
        #include <stdlib.h>
        #include "structures.h"
        #include <math.h>
        void  circle(BitmapFileHeader*  bmfh,  BitmapInfoHeader*  bmif,
Rgb**arr, int *xy, int rad, int thick, int * filcol, int *color, char
*out){
        int a=thick/2;
        if(thick/2>=rad){a=rad;}
        int h = bmif->height;
              for(int i =0; i<bmif->width; i++){
                    for(int j=0; j<h; j++){
                          if(filcol[3]!=-1){
                                i f ( ( i - * x y ) * ( i - * x y ) + ( j - * ( x y + 1 ) ) * ( j -
*(xy+1))<=rad*rad && j>0 && j<=h){
                                      arr[h-j][i].r=*(filcol);
                                      arr[h-j][i].g=*(filcol+1);
                                      arr[h-j][i].b=*(filcol+2);
                                }
                          }
                          i f ( ( i - * x y ) * ( i - * x y ) + ( j - x y [ 1 ] ) * ( j -
xy[1])<=(rad+thick/2)*(rad+thick/2)  &&  (i-*xy)*(i-*xy)+(j-xy[1])*(j-
xy[1])>=(rad-a)*(rad-a) && j>0 && j<=h){
                                arr[h-j][i].r=*(color);
                                arr[h-j][i].g=*(color+1);
                                arr[h-j][i].b=*(color+2);
                          }
                    }
              }
        write_bmp(arr, bmif->height, bmif->width, *bmfh, *bmif, out);
        }
```

Название файла: fotm.h

```
        void fotm(int argc, char* const args[]);
```

Название файла: fotm.c

```
        #include <stdlib.h>
        #include <stdio.h>
        #include <string.h>
        #include <unistd.h>
        #include <getopt.h>
        #include "fotm.h"
```

```
#include "structures.h"
#include "square.h"
#include "filter.h"
#include "rotate.h"
#include "circle.h"
#include "check.h"
#include "romb.h"
#include "smalcheck.h"
void fotm(int argc, char * const args[]){
     FILE* fptr;
     int sqropt=0;
     int fopt=0;
     int ropt=0;
     int copt=0;
     int Ropt=0;
     char buffer[1000];
     char * opts = "crgsoih";
     int opt_index = 0;
     char input[1000];
     char output[1000];
     strcpy(output, "out.bmp");
     int cordf[2];
     int cords[2];
     int side_size;
     int thickness;
     int fill[4];
     fill[3]=-1;
     fill[0]=-1;
     int color[4];
     char name[7];
     int whyDoINeedSoManyVars=0;
     int val;
     int key=0;
     struct option long_options[]={
          {"help", 0, 0,'h'},
          {"squared_lines", 0, 0,'s'},
          {"rgbfilter", 0, 0, 'g'},
          {"rotate", 0, 0, 'r'},
          {"circle", 0, 0, 'c'},
          {"left_up", 1, 0, 'l'},
          {"right_down", 1, 0, 'd'},
          {"angle", 1, 0, 'a'},
          {"side_size", 1, 0, 'z'},
          {"thickness", 1, 0, 't'},
          {"fill_color", 1, 0, 'j'},
          {"fill", 0, 0, 'f'},
          {"rhombus", 0, 0, 'R'},
          {"color", 1, 0, 'q'},
          {"center", 1, 0, 'e'},
          {"radius", 1, 0, 'u'},
          {"component_name", 1, 0, 'n'},
          {"component_value", 1, 0, 'v'},
          {"info", 0, 0, 'i'},
          {"output", 1, 0, 'o'},
          {"input", 1, 0, 'p'},
          {0, 0, 0, 0}
          };
```

```
          int opt = getopt_long(argc, args, opts, long_options,
&opt_index);
          while(1){
          if(argc==0){opt='h';}
          if(opt!='?'){
          if(opt==-1){break;}
          switch(opt){
               case 'h':
               printf("Each execution of programm requires file name
of bmp format that will be change. Default name of output file is
out.bmp(can be changed with --output). (req) mean option is required for
correct work of program.\n--squared_lines draws a square with set
cordinates, side length, side thickness, color and optionaly fillament.
Next options are available:\n --left_up (req) sets cordinates of upper
left corner(argument format: (num))\n --side_size (req) sets length of
sides in pixels(argument format:(num))\n --thickness (req) sets
thickness of all lines in pixels(argument format: (num))\n --color (req)
sets color of lines(argument format: (num).(num).(num) each number must
be in 0-255 interval)\n --fill determines if square is filled or not,
requires --fill_color after(argument format: no argument))\n --
fill_color (req if --fill is present) sets color of filler
pixels(argument format: (num).(num).(num) each number must be in 0-255
interval)\n\n--rgbfilter changes changes red, green, or blue of each
pixel to set value. Next options available:\n --component_name (req)
gets name of component that will be changed(argument format:red or green
or blue))\n --component_val (req) value that each pixel's set component
will be changed to(argument format:(num) must be in 0-255 interval)\n\n-
-rotate rotates certain part of picture(segment of picture is always a
square). Next options are available:\n --left_up (req) sets cordinates
of upper left corner(argument format:(num))\n --right_down (req) sets
cordinates of lower right corner(argument format:(num))\n --angle (req)
sets angle that segemnt will be rotate for(argument format:90 or 180 or
270)\n\n--circle draws a circle. Next options are available:\n --center
(req) sets cords of center of the circle(argument format:(num).(num))\n
--radius sets raduis in pixels(argument format:(num))\n --thickness
determines thickness of lines(argument format:(num))\n --color (req)
sets color of circle(argument format:(num).(num).(num) each num is 0-
255 interval)\n --fill if present circle will be filled withc certain
color, requires --fill_color after(argument format:no argument)\n --
fill_color (req if --fill is present) sets filler color(argument
format:(num).(num).(num) each num is in 0-255 interval)\n");return;
               break;
               case 'i':
                    whyDoINeedSoManyVars=1;
                    break;
               case 's':
                              if(key==0){key=1;}
                              else{exit(41);}
                              break;
               case 'l':
                    cordf[0]=atoi(strtok(optarg, "."));
                    cordf[1]=atoi(strtok(NULL, "."));
                    sqropt++;
                    ropt++;
                    break;
               case 'z':
                    side_size =atoi(optarg);
```

```
                    sqropt++;
                    break;
               case 't':
                    thickness =atoi(optarg);
                    sqropt++;
                    copt++;
                    break;
               case 'j':
                    strcpy(buffer, optarg);
                    if(check(buffer)){
                         fill[0]=atoi(strtok(optarg, "."));
                         fill[1]=atoi(strtok(NULL, "."));
                         fill[2]=atoi(strtok(NULL, "."));
                    }
                    else{exit(42);}
                    break;
               case 'q':
                    strcpy(buffer, optarg);
                    if(check(buffer)){
                         color[0]=atoi(strtok(optarg, "."));
                         color[1]=atoi(strtok(NULL, "."));
                         color[2]=atoi(strtok(NULL, "."));
                         sqropt++;
                         copt++;
                         Ropt++;
                         }
                    else{exit(42);}
                    break;
               case 'R':
                    if(key==0){key=5;}
                    else{exit(41);}
                    break;
               case 'g':
                              if(key==0){key=2;}
                              else{exit(41);}
                              break;
                    case 'n':
                    strcpy(name, optarg);
                    if(strcmp(name,    "red")!=0    &&    strcmp(name,
"green")!=0 && strcmp(name, "blue")!=0){exit(43);}
                    fopt++;
                              break;
                    case 'v':
                    if(smalcheck(optarg)){val = atoi(optarg);}
                    else{exit(42);}
                    fopt++;
                              break;
               case 'r':
                    if(key==0){key=3;}
                    else{exit(41);}
                    break;
                    case 'd':
                    cords[0]=atoi(strtok(optarg, "."));
                              cords[1]=atoi(strtok(NULL, "."));
                    ropt++;
                                 break;
                    case 'a':
```

```
                    val = atoi(optarg);
                    if(val!=90 && val!=180 && val!=270){exit(45);}
                    ropt++;
                            break;
               case 'c':
                            if(key==0){key=4;}
                            else{exit(41);}
                    break;
                    case 'e':
                       cordf[0]=atoi(strtok(optarg, "."));
                    cordf[1]=atoi(strtok(NULL, "."));
                    copt++;
                            break;
                       case 'u':
                    val = atoi(optarg);
                    copt++;
                            break;
               case 'p':
                    strcpy(input, optarg);
                    break;
               case 'o':
                    strcpy(output, optarg);
                    break;
               case 'f':
                    fill[3]=0;
                    break;
               default:
                    break;
               }
          }
          else{exit(48);}
          opt   =   getopt_long(argc,   args,   opts,   long_options,
&opt_index);
          }
          if(key==1    ||    key==4){if(fill[3]!=-1    &&    fill[0]==-
1){exit(44);}}
          if(strcmp(output, input)==0){exit(47);}
                         BitmapFileHeader * bmfh = (BitmapFileHeader*)
malloc(sizeof(BitmapFileHeader));
               if(bmfh==NULL){printf("Memory error\n"); exit(49);}
                         BitmapInfoHeader * bmih = (BitmapInfoHeader*)
malloc(sizeof(BitmapInfoHeader));
               if(bmih==NULL){printf("Memory error\n"); exit(49);}
          fptr=fopen(input, "rb");
          if(!fptr){exit(40);}
               fread(bmfh, 1, sizeof(BitmapFileHeader), fptr);
               if(bmfh->signa==0x4d42){
                    fread(bmih, 1, sizeof(BitmapInfoHeader), fptr);
                    if(whyDoINeedSoManyVars){
                    printf("---info---\n");
                    printf("Width:      %d\n       Height:       %d\n
bitsPerPixel: %d\n", bmih->width, bmih->height, bmih->bitsPerPixel);}
                     if(bmih->bitsPerPixel==24 && bmih->compression==0
&& bmih->headerSize==40){
                    Rgb **arr = read_bmp(fptr, bmfh, bmih);
                    switch(key){
                         case 1:
```

```
                                    if(sqropt==4){
                                    square(cordf[0], cordf[1], side_size,
thickness, fill, fptr, bmfh, bmih, arr, color, output);}
                                    else{exit(46);}
                                    break;
                               case 2:
                                    if(fopt==2){
                                    filter(bmfh, bmih, arr, name, val,
output);}
                                    else{exit(46);}
                                    break;
                               case 3:
                                    if(ropt==3){
                                    rotate(bmfh, bmih, arr, cordf, cords,
val,output);}
                                    else{exit(46);}
                                    break;
                               case 4:
                                    if(copt==4){
                                    circle(bmfh, bmih, arr, cordf, val,
thickness, fill, color, output);}
                                    else{exit(46);}
                                    break;
                               case 5:
                                    if(Ropt==1){
                                    romb(bmfh, bmih, arr, color, output);}
                                    else{exit(46);}
                                    break;
                               default:
                                    break;
                          }
                     }
               }
               free(bmih);
               free(bmfh);
          }
```

Название файла: main.c

```
     #include <stdio.h>
     #include <stdlib.h>
     #include <string.h>
     #include <getopt.h>
     #include <unistd.h>
     #include "fotm.h"
     int main(int argc, char * const args[]){
          printf("Course work for option 5.9, created by Semen
Peschatskii\n");
          fotm(argc, args);
     return 0;
     }
```

Название файла: Makefile

```
cw: fotm.o main.o structures.o square.o filter.o rotate.o circle.o
smalcheck.o check.o romb.o
     gcc fotm.o main.o structures.o square.o filter.o rotate.o circle.o
smalcheck.o check.o romb.o -o cw
```

```
fotm.o: fotm.c fotm.h structures.o square.o filter.o
	gcc -c fotm.c
structures.o: structures.c structures.h
	gcc -c structures.c
square.o: square.c square.h structures.h
	gcc -c square.c
filter.o: filter.c filter.h
	gcc -c filter.c
rotate.o: rotate.c rotate.h
	gcc -c rotate.c
circle.o: circle.c circle.h
	gcc -c circle.c
check.o: check.c check.h
	gcc -c check.c
smalcheck.o: smalcheck.c smalcheck.h
	gcc -c smalcheck.c
romb.o: romb.c romb.h structures.o
	gcc -c romb.c
main.o: main.c
	gcc -c main.c
clean:
	rm -rf *.o main
```