МИНОБРНАУКИ РОССИИ
Санкт-Петербургский государственный
электротехнический университет
«ЛЭТИ» им. В.И. Ульянова (Ленина)
Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка BMP-изображений

Студент гр. 1382 _____ Коренев Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2022

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Коренев Д.А.

Группа 1382

Тема работы: Обработка BMP-файлов

Исходные Данные:

Программа принимает на вход аргументы и изображение в формате BMP. Необходимо преобразовать картинку в соответствии с условиями и сохранить изменившуюся копию. Поддержка ведется через терминальный интерфейс (CLI — Command Line Interface).

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания 22.03.2022

Дата сдачи: 01.06.2022

Дата защиты: 03.06.2022

Студент гр. 1382 _____ Коренев Д.А.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

В ходе выполнения курсовой работы была создана программа на языке программирования C, которая обрабатывает BMP-файл. Программа имеет CLI(Command Line Interface) с возможностью вывода справки о программе, реализуемых функциях, ключах и их аргументов. Программа поддерживает BMP-файлы 3 (третьей) версии, глубиной кодирования 24 бита, без сжатия. Разработка велась на операционной системе Linux Ubuntu 20.04 в IDE CLion с использованием компилятора gcc.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы — создать программу на языке C, которая обрабатывает BMP-файл согласно запросу пользователя в соответствии с заявленным функционалом.

Для выполнения работы необходимо:

1. Создать структуры для работы BMP-файлов
2. Реализовать считывание и запись BMP-файлов
3. Написать функции обработки изображения
4. Написать функции помощи и вывода информации о файле
5. Реализовать обработку запросов пользователя с помощью CLI

Для чтения и записи файлов будут использоваться функции библиотеки stdio.h.

Для обработки запросов пользователя будет реализован с помощью библиотеки getopt.h.

Для обработки изображения будет использоваться двумерный массив структур-пикселей.

Цель работы.

Изучить принцип обработки изображения в формате BMP, научится принимать запросы пользователей которые были получены с помощью CLI, реализовать функции обработки изображения с расширением BMP на языке Си.

Задание.

Вариант 2.

- Общие сведения
- 24 бита на цвет
- без сжатия
- файл всегда соответствует формату BMP (но стоит помнить, что версий у формата несколько)
- 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 реализовывать весь следующий функционал по обработке bmp-файла
- Отражение заданной области. Этот функционал определяется:

1. выбором оси относительно которой отражать (горизонтальная или вертикальная)
2. Координатами левого верхнего угла области

3. Координатами правого нижнего угла области

- Копирование заданной области. Функционал определяется:
  1. Координатами левого верхнего угла области-источника
  2. Координатами правого нижнего угла области-источника
  3. Координатами левого верхнего угла области-назначения
- Заменяет все пиксели одного заданного цвета на другой цвет. Функционал определяется:
  1. Цвет, который требуется заменить
  2. Цвет на который требуется заменить
- Разделяет изображение на N*M частей. Реализация: провести линии заданной толщины, тем самым разделив изображение
  1. Количество частей по “оси” Y
  2. Количество частей по “оси” X
  3. Толщина линии
  4. Цвет линии

2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1 Структуры

Так как существует выравнивание структур, в памяти могут образовываться незаполненные ячейки, которые могут создать неудобства при считывании заголовков BMP-файла. Чтобы устранить их, существуют директивы #pragma pack(push,1) и #pragma pack(pop). Первая директива устанавливает «паковку» по 1 байту, а вторая – возвращает к исходной настройке.

Далее создаются структуры BitmapFileHeader и BitmapInfoHeader с полями, которые соответствуют 3 версии формата BMP. Также создается структура Rgb с полями, соответствующими каналам RGB-компонент. Каждое поле будет принимать значение от 0 до 255. А массив из таких структур будет хранить картинку. Создается структура bmpFile с полями из трех вышеописанных структур.

2.2 Считывание и сохранение

Для считывания изображения использую функция readImg. С помощью функции fopen открываю файл с именем, полученным в качестве аргумента, на чтение. Из него считывают данные в переменную типа bmpFile и возвращаю ее.

Для сохранения открываю файл на запись и записываю все значения из переменной типа bmpFile в него.

2.3 Отражение заданной области

Для отражения заданной области была написана функция invert принимающая на вход переменную типа bmpFile, название под которым изображение сохранится, каким образом необходимо будет отразить, две координаты противоположных углов. Значения высоты у координат меняются с «языка» пользователя на их устройство в BMP формате.

Корректирую координаты, провожу всевозможные тесты во избежание ошибок. С помощью двух циклов for производится swap двух пикселей в массиве относительно горизонтальной или вертикальной осей, на участке ограниченном координатами пользователя. Изображение сохраняется с помощью функции saveImg.

2.4 Копирование заданной области

Для копирования заданной области была написана функция copy, принимающая переменную типа bmpFile, название под которым изображение сохранится, три координаты: первые две — координаты противоположных углов выделенной области, третья координата — координата назначения. Значения высоты у координат меняются с «языка» пользователя на их устройство в BMP формате. Корректирую координаты, провожу всевозможные тесты во избежание ошибок. Выделенная область копируется в массив, а потом копируется из массива в нужную координату, если это возможно. Выделенная память для массива-буффера очищается. Изображение сохраняется с помощью функции saveImg.

2.5 Замена пикселей одного цвета на другой

Для замены цвета была написана функция replace, принимающая переменную типа bmpFile, название под которым изображение сохранится, два цвета, для каждого по три переменных. С помощью двух циклов for «пробегаюсь» по всем пикселям изображения, заменяя один цвет на другой при необходимости. Изображение сохраняется с помощью функции saveImg.

2.6 Разделение изображения

Для разделения изображения на части путем рисования линий заданной толщины, которые и будут разделять изображение на части, была реализована функция lines, принимает переменную типа bmpFile, название под которым сохраниться изображение, количество частей по высоте и ширине, толщину линии, цвет линии. Сначала создаю новую переменную

типа bmpFile с увеличенными полями высоты и шириной, а так же бОльшим массивом пикселей, остальное соответствует полям полям исходного изображениям. Изображение из исходной картинки копирую в новый массив, так как пиксель с координатами (0,0) находится в нижнем левом углу изображения, то скопированное изображение находится в нижнем левом углу. Далее с помощью циклов изображение «разрезается» и «вытягивается» вверх на высоту равную ширине линии, а дублирующиеся пиксели — закрашиваются, аналогично происходит и по ширине — влево. Полученное изображение сохраняется с помощью функции saveImg.

2.7 Вывод справки и информации о файле

Для вывода справки — помощь были написана функция help которые выводит в терминал описание программы, описаний функций, ключи необходимые для каждой, информацию о ключах.

Для вывода информации о файле была написана функция printImageInfo, которыа принимает в качестве аргумента переменную типа bmpFile и выводит о ней данные.

2.8 Считывание пользовательских данных

Для считывания данных использовалась библиотека getopt. Были созданы короткие и длинные ключи, в которые пользователь передает данные, далее, в зависимости от вызванной функции, программа выполнит задачу и сохранить файл под нужным именем либо выведет сообщение-ошибку в терминал.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была написана программа на языке Си, имеющая CLI, для обработки изображений - BMP-файлов.

Пользователь вводит ключ функции, название файла, который необходимо изменить, ключи, необходимые для выполнения функции, название файла в который будет сохранен результат. Преобразования, которые может сделать программа: отразить заданную область, копирование заданной области, изменение одного цвета пикселя на другой, разделить изображение на части, с помощью рисования линий.

Полученный результат соответствует поставленной цели.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Информация о BMP-формате и его версиях.

https://ru.wikipedia.org/wiki/BMP

2. Работа с библиотекой getopt.h для работы с CLI

https://man7.org/linux/man-pages/man3/getopt.3.html

ПРИЛОЖЕНИЕ А

ПРИМЕР РАБОТЫ ПРОГРАММЫ

Пример 1:

Отражение заданной области с координатами (1800,400) и (700,1600), название выходного файла out.bmp. Несмотря на то, что входные координаты являлись правым верхним и левым нижним, а также выходили за пределы изображения, программа выполнила функцию на той области, которую предполагал пользователь.

```
ajems@ajems:~/CLionProjects/pr_cw$ gcc main.c -o cw
ajems@ajems:~/CLionProjects/pr_cw$ ./cw -i leti.bmp -s 1800,400 -e 700,1600 -o h out.bmp
saved to out.bmp
```

Рис 1: Отображение области

Пример 2:

Копирование заданной области с координатам (200,200) и (600,650) в точку назначения с координатой (700,300), название выходного файла out.bmp.

```
ajems@ajems:~/CLionProjects/pr_cw$ gcc main.c -o cw
ajems@ajems:~/CLionProjects/pr_cw$ ./cw -c leti.bmp -s 200,200 -e 600,650 -d 700,300 out.bmp
saved to out.bmp
```

Рис 2: Копирование области

Пример 3:

Замена пикселей одного цвета на другой. Данный пример меняет белый (255,255,255) цвет на черный (0,0,0), название выходного файла out.bmp.

```
ajems@ajems:~/CLionProjects/pr_cw$ gcc main.c -o cw
ajems@ajems:~/CLionProjects/pr_cw$ ./cw --replace simpsonsvr.bmp -1 255,255,255 -2 0,0,0 out.bmp
saved to out.bmp
```

Рис 3: Замена пикселей

Пример 4:

Разделение изображения на части путем рисования линий заданной толщины. В данном примере изображение делится на 5*10 частей (5 по вертикали, 10 по горизонтали) с толщиной 40 пикселей, желтого цвета, название выходного файла out.bmp.

```
ajems@ajems:~/CLionProjects/pr_cw$ gcc main.c -o cw
ajems@ajems:~/CLionProjects/pr_cw$ ./cw -l shrek.bmp -x 10 -y 15 -t 60 -1 255,255,0 out.bmp
saved to out.bmp
```

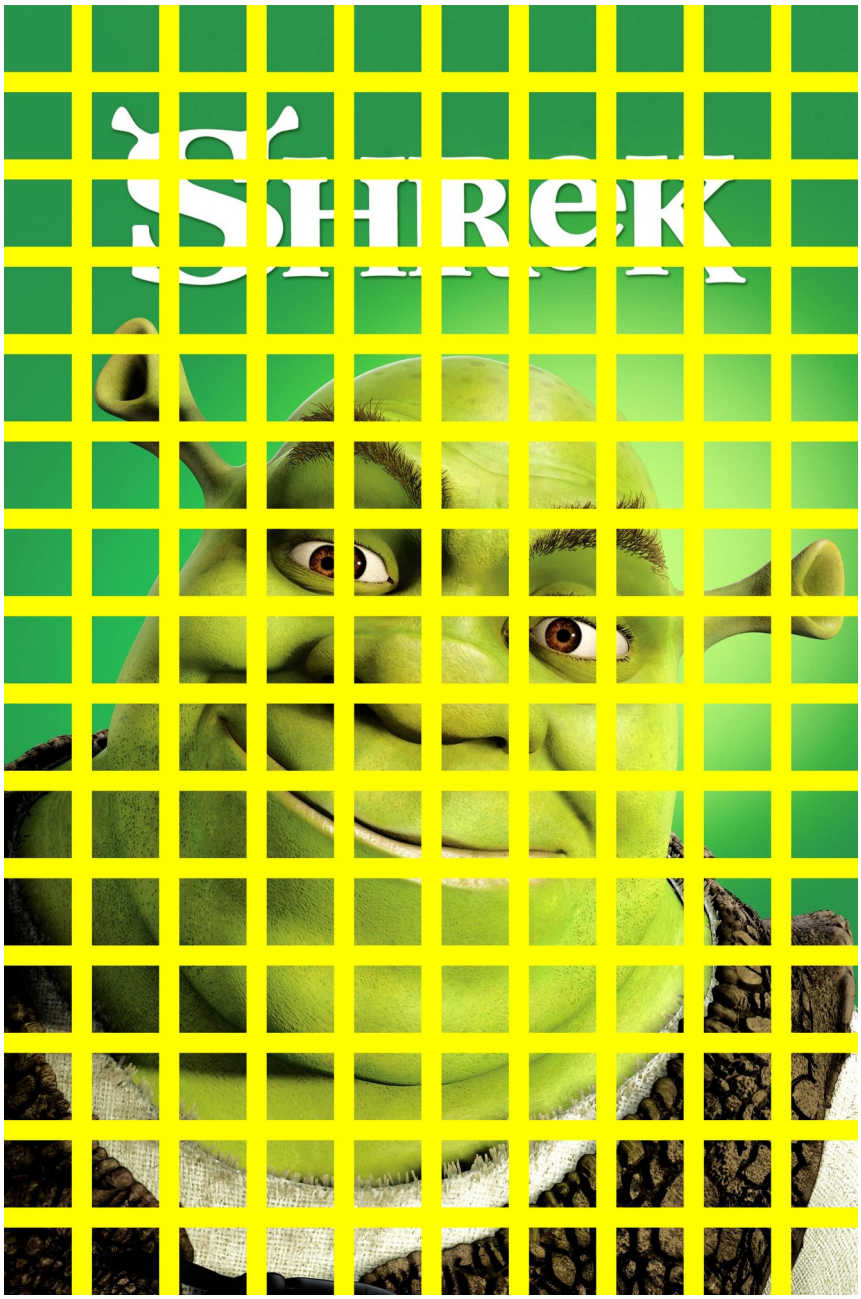

Рис 4: Разделение изображения на части

Пример 5:

Вывод информации об изображении.

```
ajems@ajems:~/CLionProjects/pr_cw$ ./cw -p shrek.bmp
Signature:      4d42 (19778)
filesize:       112a8b6 (18000054)
reserved1:      0 (0)
reserved2:      0 (0)
pixelArrOffset: 36 (54)
headerSize:     28 (40)
width:          7d0 (2000)
height:         bb8 (3000)
planes:         1 (1)
bitsPerPixel:   18 (24)
compression:    0 (0)
imageSize:      0 (0)
xPixelsPerMeter:        b13 (2835)
yPixelsPerMeter:        b13 (2835)
colorsInColorTable:     0 (0)
importantColorCount:    0 (0)
```

Рис 5: Информация о файле

Пример 6:

Вывод справки.

```
ajems@ajems:~/CLionProjects/pr_cw$ ./cw --help
NAME
                        BMP Photo editor

DESCRIPTION
       Program supports CLI and only works with version 3 BMP files
       BMP files with color table are not supported
       The program only supports files with a depth of 24 pixels per bit
       File must not be compressed

FUNCTIONS
       1 - Replace Color (-r/--replace)
       Replace one color to another
       Required arguments:
              -r/--replace
              -1/--firstColor
              -2/--secondColor

       2 - Invert Area Image (-i/--invert)
       Inverts vertically or horizontally the selected area
       Required arguments:
              -i/--invert
              -o/--option
              -s/--start
              -e/--end

       3 - Copy Area Image (-c/--copy)
       Copy selected area to destination
       Required arguments:
              -c/--copy
              -s/--start
              -e/--end
              -d/--destination

       4 - Draw Line Collage (-l/--lines)
       Draws lines vertically and horizontally creating a collage
       Required arguments:
              -l/--line
              -x/--xLines
              -y/--yLines
              -t/--thickness
              -1/--firstColor

KEYS
       -r/-i/-c/-l [Filename.bmp]                          called the entered function
       -s/--start [value width] [value height]             sets the starting coordinates
       -e/--end [value width[ [value height]               sets the ending coordinates
       -d/--destination [value width] [value height]       sets the destination coordinates
       -1/--firstColor/-2/--secondColor [red] [green] [blue]   sets color in RGB format
       -o/--option [h/v]                                   sets option for invert:
                                                           h - horizontal, v - vertical
       -x/--xLines [value]                                 number of lines in width
       -y/--yLines [value]                                 number of lines in height
       -t/--thickness                                      line thickness
```

Рис 6: Вывод справки

Пример 7:

Обработка всевозможных ошибок.

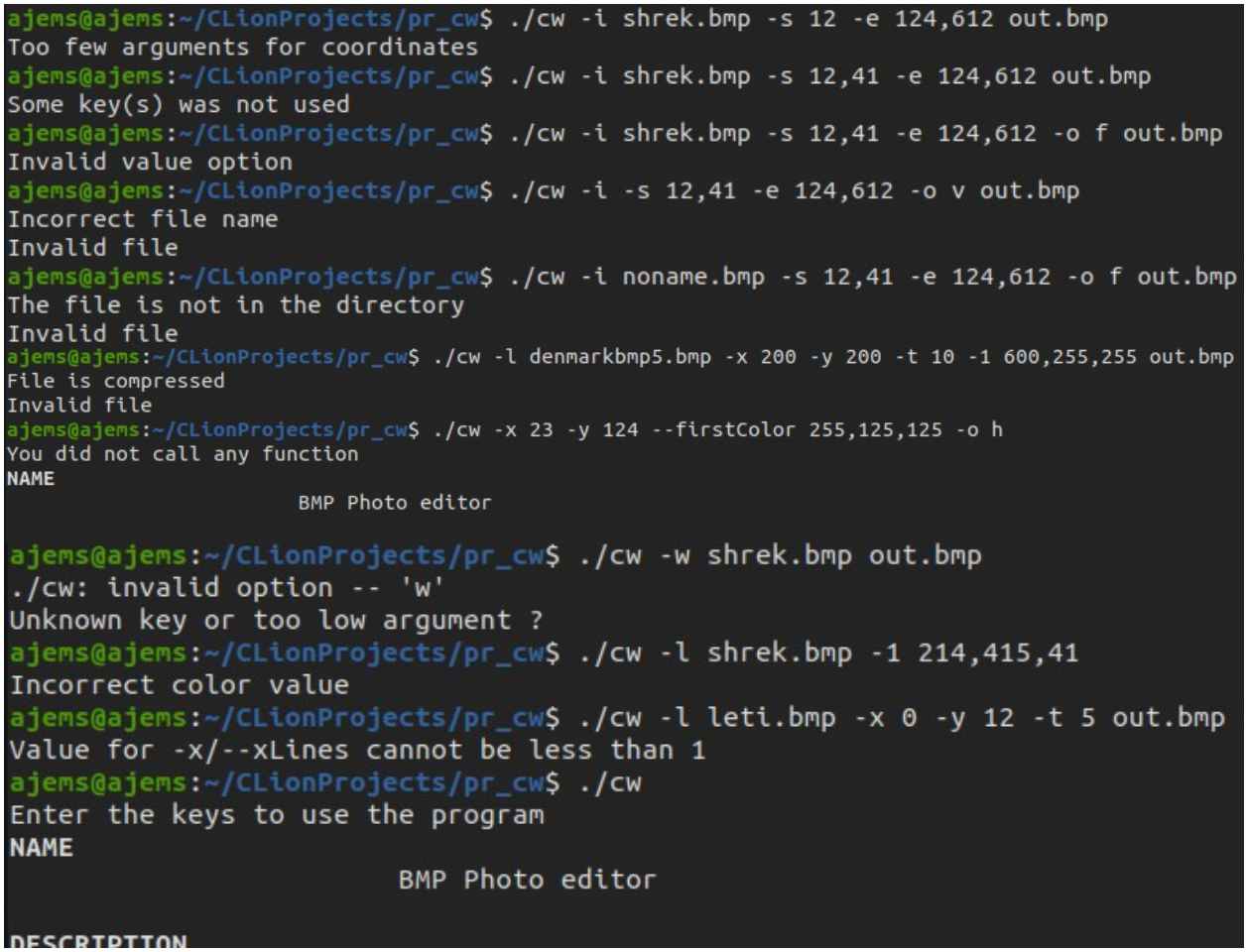

Рис 7: Обработка ошибок

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#define REPLACE 1
#define INVERT 2
#define COPY 3
#define LINES 4

#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;

typedef struct{
    BitmapFileHeader fileHeader;
    BitmapInfoHeader fileInfo;
    Rgb** rgb;
}bmpFile;
#pragma pack(pop)

void help();

bmpFile readImg(char* name){
    FILE *f = fopen(name, "rb");
    bmpFile img;
```

```
    fread(&img.fileHeader ,1,sizeof(BitmapFileHeader),f);
    fread(&img.fileInfo,1,sizeof(BitmapInfoHeader),f);

    unsigned int H = img.fileInfo.height;
    unsigned int W = img.fileInfo.width;

    img.rgb = malloc(H*sizeof(Rgb*));
    for(int i=0; i<H; i++){
        img.rgb[i] = malloc(W * sizeof(Rgb) + (4-(W*sizeof(Rgb))
%4)%4);
        fread(img.rgb[i],1,W * sizeof(Rgb) + (4-(W*sizeof(Rgb))
%4)%4,f);
    }
    return img;
}

void saveImg(bmpFile* img, char* nameOut){
    int len = (int)strlen(nameOut);
    if(nameOut[len-4] != '.' || nameOut[len-3] != 'b' || nameOut[len-
2] != 'm' || nameOut[len-1] != 'p'){
        printf("Incorrect file name\n");
        return;
    }

    FILE *f = fopen(nameOut, "wb");
    if(!f){
        printf("File cannot to be opened\n");
        return;
    }
    fwrite(&img->fileHeader, 1, sizeof(BitmapFileHeader),f);
    fwrite(&img->fileInfo, 1, sizeof(BitmapInfoHeader),f);

    unsigned int W = (img->fileInfo.width)*sizeof(Rgb)+ (4-(img-
>fileInfo.width*sizeof(Rgb))%4)%4;

    for(int i=0; i<img->fileInfo.height; i++){
        fwrite(img->rgb[i],1, W ,f);
        free(img->rgb[i]);
    }
    //printf("saved to %s\n", nameOut);
    fclose(f);
}


void replace(bmpFile* img,
                 char* nameOut,
                 unsigned char r1,
                 unsigned char g1,
                 unsigned char b1,
                 unsigned char r2,
                 unsigned char g2,
                 unsigned char b2){

    for (int h = 0; h < img->fileInfo.height ; h++){
        for (int w = 0; w < img->fileInfo.width; w++){
            if ((img->rgb[h][w].r == r1) && (img->rgb[h][w].g == g1)
```

```
&& (img->rgb[h][w].b == b1)){
                img->rgb[h][w].r = r2;
                img->rgb[h][w].g = g2;
                img->rgb[h][w].b = b2;
            }
        }
    }
    saveImg(img, nameOut);
}

void invert(bmpFile* img,
            char* nameOut,
            char var,
            int leftWidth,
            int leftHeight,
            int rightWidth,
            int rightHeight){

    leftHeight = (int)img->fileInfo.height - leftHeight - 1;
    rightHeight = (int)img->fileInfo.height - rightHeight - 1;


    if (leftHeight < rightHeight){
        int tmp = rightHeight;
        rightHeight = leftHeight;
        leftHeight = tmp;
    }

    if (rightWidth < leftWidth){
        int tmp = rightWidth;
        rightWidth = leftWidth;
        leftWidth = tmp;
    }

    if (rightWidth > (int)img->fileInfo.width - 1) rightWidth =
(int)img->fileInfo.width - 1;
    if (rightHeight < 0) rightHeight = 0;
    if (rightWidth < 0) rightWidth = 0;
    if (rightHeight > (int)img->fileInfo.height -1) rightHeight =
(int)img->fileInfo.height-1;
    if (leftWidth > (int)img->fileInfo.width - 1) leftWidth =
(int)img->fileInfo.width - 1;
    if (leftHeight < 0) leftHeight = 0;
    if (leftWidth < 0) leftWidth = 0;
    if (leftHeight > (int)img->fileInfo.height -1) leftHeight =
(int)img->fileInfo.height-1;


    if (var == 'h') {
        for (unsigned int h = rightHeight; h <=
(leftHeight+rightHeight)/2; h++){
            for (unsigned int w = leftWidth; w <= rightWidth; w++) {
                Rgb tmp;
                tmp = img->rgb[h][w];
                img->rgb[h][w] = img->rgb[leftHeight+rightHeight-h]
[w];
```

```
                img->rgb[leftHeight+rightHeight-h][w] = tmp;
            }
        }
    } else if(var == 'v'){
        for (unsigned int h = rightHeight; h <= leftHeight; h++) {
            for (unsigned int w = leftWidth; w <=
(rightWidth+leftWidth)/2; w++) {
                Rgb tmp;
                tmp = img->rgb[h][w];
                img->rgb[h][w] = img->rgb[h][rightWidth+leftWidth-w];
                img->rgb[h][rightWidth+leftWidth-w] = tmp;
            }
        }
    }


    saveImg(img, nameOut);
}

void copy(bmpFile* img,
            char* nameOut,
            int leftWidth,
            int leftHeight,
            int rightWidth,
            int rightHeight,
            int toLeftWidth,
            int toLeftHeight){

    leftHeight = (int)img->fileInfo.height - leftHeight - 1;
    rightHeight = (int)img->fileInfo.height - rightHeight - 1;
    toLeftHeight = (int)img->fileInfo.height - toLeftHeight - 1;

    if (leftHeight < rightHeight){
        int tmp = rightHeight;
        rightHeight = leftHeight;
        leftHeight = tmp;
    }
    if (rightWidth < leftWidth){
        int tmp = rightWidth;
        rightWidth = leftWidth;
        leftWidth = tmp;
    }

    if (rightWidth > (int)img->fileInfo.width - 1) rightWidth =
(int)img->fileInfo.width - 1;
    if (rightHeight < 0) rightHeight = 0;
    if (rightWidth < 0) rightWidth = 0;
    if (rightHeight > (int)img->fileInfo.height -1) rightHeight =
(int)img->fileInfo.height-1;
    if (leftWidth > (int)img->fileInfo.width - 1) leftWidth =
(int)img->fileInfo.width - 1;
    if (leftHeight < 0) leftHeight = 0;
    if (leftWidth < 0) leftWidth = 0;
    if (leftHeight > (int)img->fileInfo.height -1) leftHeight =
(int)img->fileInfo.height-1;
```

```
    //copy to buf
    bmpFile buf;
    int heightBuf = leftHeight-rightHeight+1;
    int widthBuf = rightWidth-leftWidth+1;
    buf.fileInfo.height = heightBuf;
    buf.fileInfo.width = widthBuf;

    buf.rgb = malloc(heightBuf*sizeof(Rgb*));
    for(unsigned int h = 0; h < heightBuf; h++){
        buf.rgb[h] = malloc(widthBuf * sizeof(Rgb));
        for (int w = 0; w < widthBuf; w++){
            buf.rgb[h][w] = img->rgb[rightHeight+h][leftWidth+w];
        }
    }

    //insert from buf
    for(int h = 0; h < heightBuf; h++){
        for (int w = 0; w < widthBuf; w++){
            if ((toLeftHeight-(heightBuf-h) < 0) || (toLeftWidth+w) >=
img->fileInfo.width)
                continue;

            img->rgb[toLeftHeight-(heightBuf-h)][toLeftWidth+w] =
buf.rgb[h][w];
        }
    }

    for(unsigned int h = 0; h <= heightBuf; h++){
        free(buf.rgb[h]);
    }
    saveImg(img, nameOut);
}

void lines(bmpFile* img,
             char* nameOut,
             int n,//h
             int m,//w
             int t,
             char r,
             char g,
             char b){
    int oH = (int)img->fileInfo.height;
    int oW = (int)img->fileInfo.width;

    bmpFile nimg;
    nimg.fileInfo = img->fileInfo;
    nimg.fileHeader = img->fileHeader;
    nimg.fileInfo.height+=t*(n-1);
    nimg.fileInfo.width+=t*(m-1);

    int nH = (int)nimg.fileInfo.height;
    int nW = (int)nimg.fileInfo.width;


    nimg.rgb = malloc(nH*sizeof(Rgb*) + (4-(nH*sizeof(Rgb))%4)%4);
```

```
    for(int i=0; i<nH; i++){
        nimg.rgb[i] = malloc(nW * sizeof(Rgb) + (4-(nW*sizeof(Rgb))
%4)%4);
    }

    for (int h = 0; h < oH; h++){
        for (int w = 0; w < oW; w++){
            nimg.rgb[h][w] = img->rgb[h][w];
        }
    }

    for(int ph = 1; ph < n; ph++){
        for (int h = oH-1+t*ph; h > (ph*oH)/n+t*(ph-1); h--){
            for (int w = 0; w < oW; w++){
                if (h+t < nH)
                    nimg.rgb[h+t][w] = nimg.rgb[h][w];
            }
        }

        for (int h = 0; h < t; h++){
            for (int w = 0; w < nW; w++){
                nimg.rgb[(ph*oH)/n+t*(ph-1)+1+h][w].r = r;
                nimg.rgb[(ph*oH)/n+t*(ph-1)+1+h][w].g = g;
                nimg.rgb[(ph*oH)/n+t*(ph-1)+1+h][w].b = b;
            }
        }

    }

    for(int pw = 1; pw < m; pw++){
        for (int w = oW-1+t*pw; w > (pw*oW)/m+t*(pw-1); w--){
            for (int h = 0; h < nH; h++){
                if (w+t < nW)
                    nimg.rgb[h][w+t] = nimg.rgb[h][w];
            }
        }
        for (int w = 0; w < t; w++){
            for (int h = 0; h < nH; h++){
                nimg.rgb[h][(pw*oW)/m+t*(pw-1)+1+w].r = r;
                nimg.rgb[h][(pw*oW)/m+t*(pw-1)+1+w].g = g;
                nimg.rgb[h][(pw*oW)/m+t*(pw-1)+1+w].b = b;
            }
        }
    }

    for(unsigned int h = 0; h <oH; h++){
        free(img->rgb[h]);
    }

    saveImg(&nimg, nameOut);

}


int correctFile(bmpFile* img, char* name){
    unsigned long len = strlen(name);
```

```
    if(name[len-4] != '.' || name[len-3] != 'b' || name[len-2] != 'm'
|| name[len-1] != 'p'){
        printf("Incorrect file name\n");
        return 1;
    }
    FILE* file = fopen(name, "rb");
    if(!file){
        printf("The file is not in the directory\n");
        return 1;
    }
    *img = readImg(name);
    if(img->fileInfo.compression != 0){
        printf("File is compressed\n");
        return 1;
    }
    if(img->fileInfo.bitsPerPixel != 24){
        printf("File color depth is not 24 bits per color\n");
        return 1;
    }
    if(img->fileInfo.headerSize != 40){
        printf("This version of the BMP file is not supported\n");
        return 1;
    }
    if(img->fileInfo.colorsInColorTable != 0 || img-
>fileInfo.importantColorCount != 0){
        printf("File must not use color table\n");
        return 1;
    }
    return 0;
}

void printImageInfo(bmpFile* image){

    printf("Signature:\t%x (%hu)\n", image->fileHeader.signature,
image->fileHeader.signature);
    printf("filesize:\t%x (%u)\n", image->fileHeader.filesize, image-
>fileHeader.filesize);
    printf("reserved1:\t%x (%hu)\n", image->fileHeader.reserved1,
image->fileHeader.reserved1);
    printf("reserved2:\t%x (%hu)\n", image->fileHeader.reserved2,
image->fileHeader.reserved2);
    printf("pixelArrOffset:\t%x (%u)\n", image-
>fileHeader.pixelArrOffset, image->fileHeader.pixelArrOffset);
    printf("headerSize:\t%x (%u)\n", image->fileInfo.headerSize,
image->fileInfo.headerSize);
    printf("width:     \t%x (%u)\n", image->fileInfo.width, image-
>fileInfo.width);
    printf("height:    \t%x (%u)\n", image->fileInfo.height, image-
>fileInfo.height);
    printf("planes:    \t%x (%hu)\n", image->fileInfo.planes, image-
>fileInfo.planes);
    printf("bitsPerPixel:\t%x (%hu)\n", image->fileInfo.bitsPerPixel,
image->fileInfo.bitsPerPixel);
    printf("compression:\t%x (%u)\n", image->fileInfo.compression,
image->fileInfo.compression);
    printf("imageSize:\t%x (%u)\n", image->fileInfo.imageSize, image-
```

```
>fileInfo.imageSize);
    printf("xPixelsPerMeter:\t%x (%u)\n", image-
>fileInfo.xPixelsPerMeter, image->fileInfo.xPixelsPerMeter);
    printf("yPixelsPerMeter:\t%x (%u)\n", image-
>fileInfo.yPixelsPerMeter, image->fileInfo.yPixelsPerMeter);
    printf("colorsInColorTable:\t%x (%u)\n", image-
>fileInfo.colorsInColorTable, image->fileInfo.colorsInColorTable);
    printf("importantColorCount:\t%x (%u)\n",image-
>fileInfo.importantColorCount, image->fileInfo.importantColorCount);
}


void help(){
    char text[] =  "\033[1mNAME\033[0m\n"
                   "\t\t\tBMP Photo editor\n\n"
                   "\033[1mDESCRIPTION\033[0m\n"
                   "\tProgram supports CLI and only works with version
3 BMP files\n"
                   "\tBMP files with color table are not supported\n"
                   "\tThe program only supports files with a depth of
24 pixels per bit\n"
                   "\tFile must not be compressed\n\n"
                   "\033[1mFUNCTIONS\033[0m\n"
                   "\t1 - Replace Color (-r/--replace)\n"
                   "\tReplace one color to another\n"
                   "\t\033[1mRequired arguments:\033[0m\n"
                   "\t\t-r/--replace\n"
                   "\t\t-1/--firstColor\n"
                   "\t\t-2/--secondColor\n\n"
                   "\t2 - Invert Area Image (-i/--invert)\n"
                   "\tInverts vertically or horizontally the selected
area\n"
                   "\t\033[1mRequired arguments:\033[0m\n"
                   "\t\t-i/--invert\n"
                   "\t\t-o/--option\n"
                   "\t\t-s/--start\n"
                   "\t\t-e/--end\n\n"
                   "\t3 - Copy Area Image (-c/--copy)\n"
                   "\tCopy selected area to destination\n"
                   "\t\033[1mRequired arguments:\033[0m\n"
                   "\t\t-c/--copy\n"
                   "\t\t-s/--start\n"
                   "\t\t-e/--end\n"
                   "\t\t-d/--destination\n\n"
                   "\t4 - Draw Line Collage (-l/--lines)\n"
                   "\tDraws lines vertically and horizontally creating
a collage\n"
                   "\t\033[1mRequired arguments:\033[0m\n"
                   "\t\t-l/--line\n"
                   "\t\t-x/--xLines\n"
                   "\t\t-y/--yLines\n"
                   "\t\t-t/--thickness\n"
                   "\t\t-1/--firstColor\n\n"
                   "\033[1mKEYS\033[0m\n"
                   "\t-r/-i/-c/-l [Filename.bmp]\t\t\t\tcalled the
entered function\n"
```

```
                    "\t-s/--start [value width] [value height]\t\t\
tsets the starting coordinates\n"
                    "\t-e/--end [value width[ [value height]\t\t\tsets
the ending coordinates\n"
                    "\t-d/--destination [value width] [value height]\t\
tsets the destination coordinates\n"
                    "\t-1/--firstColor/-2/--secondColor [red] [green]
[blue]\tsets color in RGB format\n"
                    "\t-o/--option [h/v]\t\t\t\tsets option for
invert:\n"
                    "\t\t\t\t\t\t\th - horizontal, v - vertical\n"
                    "\t-x/--xLines [value]\t\t\t\tnumber of lines in
width\n"
                    "\t-y/--yLines [value]\t\t\t\tnumber of lines in
height\n"
                    "\t-t/--thickness\t\t\t\t\tline thickness\n";
    puts(text);
}


int main(int argc, char *argv[]){

    char *opts = "r:i:c:l:f:1:2:s:e:o:d:y:x:t:z:hp:";
    struct option longOpts[]={
            {"replace", required_argument, NULL, 'r'},
            {"invert", required_argument, NULL, 'i'},
            {"copy", required_argument, NULL, 'c'},
            {"lines", required_argument, NULL, 'l'},
            {"file", required_argument, NULL, 'f'},
            {"firstColor", required_argument, NULL, '1'},
            {"secondColor", required_argument, NULL, '2'},
            {"start", required_argument, NULL,'s'},
            {"end", required_argument, NULL, 'e'},
            {"option", required_argument, NULL, 'o'},
            {"destination", required_argument, NULL,'d'},
            {"yLines", required_argument, NULL,'y'},
            {"xLines", required_argument, NULL,'x'},
            {"thickness", required_argument, NULL,'t'},
            {"help", no_argument, NULL, 'h'},
            {"printInfo", no_argument, NULL, 'p'},
            {NULL, 0, NULL, 0}
    };

    int opt;
    int longOpt;
    opt = getopt_long(argc, argv, opts, longOpts, &longOpt);


    if (argc < 2){
        printf("Enter the keys to use the program\n");
        help();
        return 0;
    }

    char inputFile[256];
    char outputFile[255];
```

```
    strcpy(outputFile, argv[argc-1]);
    bmpFile img;

    int way = 0;
    int x1, y1, x2, y2, dx, dy, r1, g1, b1, r2, g2, b2, xLines,
yLines, thickness = 0;
    int startCoord, endCoord, distCoord, firstClr, secondClr, thick,
xCnt, yCnt = 0;

    char option = 'n';
    int countRead;

    while (opt != -1){
        switch (opt) {
            case 'r':{
                countRead = sscanf(optarg, "%s", inputFile);
                if (countRead < 1){
                    printf("File name was not entered\n");
                    return 1;
                }
                if (correctFile(&img, inputFile) != 0){
                    printf("Invalid file\n");
                    return 1;
                }
                way = REPLACE;
                break;
            }
            case 'i':{
                countRead = sscanf(optarg, "%s", inputFile);
                if (correctFile(&img, inputFile) != 0){
                    printf("Invalid file\n");
                    return 1;
                }
                if (countRead < 1){
                    printf("Too few arguments\n");
                    return 1;
                }
                way = INVERT;
                break;
            }
            case 'c':{
                countRead = sscanf(optarg, "%s", inputFile);
                if (correctFile(&img, inputFile) != 0){
                    printf("Invalid file\n");
                    return 1;
                }
                if (countRead < 1){
                    printf("Too few arguments\n");
                    return 1;
                }
                way = COPY;
                break;
            }
            case 'l':{
                countRead = sscanf(optarg, "%s", inputFile);
                if (correctFile(&img, inputFile) != 0){
```

```
                    printf("Invalid file\n");
                    return 1;
                }
                if (countRead < 1){
                    printf("Too few arguments\n");
                    return 1;
                }
                way = LINES;
                break;
            }
            case 'f':{
                countRead = sscanf(optarg, "%s", outputFile);
                if (countRead < 1){
                    printf("Too few arguments for file name\n");
                    return 1;
                }
                if (strcmp(outputFile, "no") == 0){
                    strcpy(outputFile, inputFile);
                    break;
                }
                int len = strlen(outputFile);
                if (len < 5 || outputFile[len-4] != '.' ||
outputFile[len-3] != 'b' || outputFile[len-2] != 'm' ||
outputFile[len-1] != 'p'){
                    printf("Incorrect file name\n");
                    return 1;
                }
                break;
            }
            case 's':{
                countRead = sscanf(optarg, "%d,%d", &x1, &y1);
                if (countRead < 2){
                    printf("Too few arguments for coordinates\n");
                    return 1;
                }
                startCoord = 1;
                break;
            }
            case 'e':{
                countRead = sscanf(optarg, "%d,%d", &x2,&y2);
                if (countRead < 2){
                    printf("Too few arguments for coordinates\n");
                    return 1;
                }
                endCoord = 1;
                break;
            }
            case 'o':{
                countRead = sscanf(optarg, "%c", &option);
                if (countRead < 1){
                    printf("Too few arguments for option\n");
                    return 1;
                }
                if (option != 'v' && option != 'h'){
                    printf("Invalid value option\n");
                    return 1;
```

```
                }
                break;
            }
            case 'd':{
                countRead = sscanf(optarg, "%d,%d", &dx, &dy);
                if (countRead < 2){
                    printf("Too few arguments for coordinates\n");
                    return 1;
                }
                distCoord = 1;
                break;
            }
            case '1':{
                countRead = sscanf(optarg, "%d,%d,%d", &r1, &g1, &b1);
                if (countRead < 3){
                    printf("Too few arguments for color\n");
                    return 1;
                }
                if (r1 > 255 || r1 < 0 || g1 > 255 || g1 < 0 || b1 >
255 || b1 < 0){
                    printf("Incorrect color value\n");
                    return 1;
                }
                firstClr = 1;
                break;
            }
            case '2':{
                countRead = sscanf(optarg, "%d,%d,%d", &r2, &g2, &b2);
                if (countRead < 3){
                    printf("Too few arguments for color\n");
                    return 1;
                }
                if (r2 > 255 || r2 < 0 || g2 > 255 || g2 < 0 || b2 >
255 || b2 < 0){
                    printf("Incorrect color value\n");
                    return 1;
                }
                secondClr = 1;
                break;
            }
            case 'y':{
                countRead = sscanf(optarg, "%d", &yLines);
                if (countRead < 1){
                    printf("Too few arguments for number lines by Y\
n");
                    return 1;
                }
                if (yLines < 1){
                    printf("Value for -y/--yLines cannot be less than
1\n");
                    return 1;
                }
                yCnt = 1;
                break;
            }
            case 'x':{
```

```
                countRead = sscanf(optarg, "%d", &xLines);
                if (countRead < 1){
                    printf("Too few arguments for number lines by X\
n");
                    return 1;
                }
                if (xLines < 1){
                    printf("Value for -x/--xLines cannot be less than
1\n");
                    return 1;
                }
                xCnt = 1;
                break;
            }
            case 't':{
                countRead = sscanf(optarg, "%d", &thickness);
                if (countRead < 1){
                    printf("Too few arguments for thickness\n");
                    return 1;
                }
                if (thickness < 1){
                    printf("Value for -t/--thickness cannot be less
than 1n\n");
                    return 1;
                }
                thick = 1;
                break;
            }
            case 'h':{
                help();
                return 0;
            }
            case 'p':{
                countRead = sscanf(optarg, "%s", inputFile);
                if (countRead < 1){
                    printf("Too few argument for file info");
                    return 1;
                }
                if (correctFile(&img, inputFile) != 0){
                    printf("Invalid file\n");
                    return 1;
                }
                printImageInfo(&img);
                return 0;
            }
            default:{
                printf("Unknown key\n");
                return 1;
            }
        }
        opt = getopt_long(argc, argv, opts, longOpts, &longOpt);
    }

    switch (way) {
        case REPLACE:{
            if (firstClr == 1 && secondClr == 1)
```

```
                replace(&img, outputFile, r1, g1, b1, r2, g2, b2);
            else
                printf("Some key(s) was not used\n");
            break;
        }
        case COPY:{
            if (startCoord == 1 && endCoord == 1 && distCoord == 1)
                copy(&img, outputFile, x1, y1, x2, y2, dx, dy);
            else
                printf("Some key(s) was not used\n");
            break;
        }
        case INVERT:{
            if (option != 'n' && startCoord == 1 && endCoord == 1)
                invert(&img, outputFile, option,x1, y1, x2, y2);
            else
                printf("Some key(s) was not used\n");
            break;
        }
        case LINES:{
            if (secondClr == 1 && firstClr != 1){
                r1 = r2;
                g1 = g2;
                b1 = b2;
                firstClr = 1;
            }
            if (xCnt == 1 && yCnt == 1 && thick == 1 && firstClr == 1)
                lines(&img, outputFile, yLines, xLines, thickness,
(char)r1, (char)g1, (char)b1);
            else {
                printf("Some key(s) was not used\n");
            }
            break;
        }
        default:{
            printf("You did not call any function\n");
            help();
        }
    }

    return 0;
}
```