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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка BMP файла

Студент гр. 0382 _____ Гудов Н.Р.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2021

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Гудов Н.Р.

Группа 0382

Тема работы: Обработка BMP файла

Исходные данные:
Вариант 7

Программа должна реализовывать весь следующий функционал по обработке bmp-файла

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 следующий функционал по обработке bmp-файла

(1) Рисование треугольника. Треугольник определяется

- Координатами его вершин
- Толщиной линий
- Цветом линий
- Треугольник может быть залит или нет
- цветом которым он залит, если пользователем выбран залитый

(2) Находит самый большой прямоугольник заданного цвета и перекрашивает его в другой цвет. Функционал определяется:

- Цветом, прямоугольник которого надо найти
- Цветом, в который надо его перекрасить

(3) Создать коллаж размера N*M из одного либо нескольких фото -- на выбор студента (либо оба варианта по желанию). В случае с одним изображением коллаж представляет собой это же самое изображение повторяющееся N*M раз.

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

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 5.04.2021

Дата сдачи реферата: 24.05.2021

Дата защиты реферата: 25.05.2021

Студент _____ Гудов Н.Р.

Преподаватель _____ Берленко Т.А.

АННОТАЦИЯ

Создана программа, обрабатывающая BMP файлы с глубиной 24 бита и не имеющих сжатие. Взаимодействие с программой происходит при помощи реализации CLI.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы-создать программу для обработки BMP файла с возможностью взаимодействия с ней.

Для этого был реализован CLI при помощи функций библиотеки getopt.h.

Были определены пользовательские структуры и функции.

2. ВТОРОЙ РАЗДЕЛ

2.1. Ввод аргументов

Ввод аргументов осуществляется при запуске программы через консоль строкой вида:

“prog_name func_name input_file out_file -flag1 –flag2 flag…”

Запуск программы без аргументов или с функцией help выводит список доступных команд и ключей.

Запуск программы с неправильно заданной функцией сопровождается предупреждением о несуществующей команде.

Запуск программы с неправильно заданным файлом для чтения сопровождается предупреждением о невозможности открыть файл.

Доступные функции:

triangle-рисование треугольника по вершинам;

rectfind-нахождение прямоугольника заданного цвета;

collage-составление из заданной картинки коллажа;

write-функция перезаписи из файла в файл.

Доступные ключи:

Для первой функции: c/1color; C/2color; A/Apoint; B/Bpoint; D/Cpoint; f/Fill; V/lWidth;

Для второй функции: c/1color; C/2color;

Для третьей функции: X/xPic; Y/yPi;

Где: c/1color; C/2color – флаги для цвета. A/Apoint; B/Bpoint; D/Cpoint – флаги для точек вершин треугольника. f/Fill; V/lWidth-заливка и толщина линии. X/xPic; Y/yPi-количество картинок по горизонталии.

2.2. Чтение файла

Чтение файла осуществляется при помощи пользовательской функции *BMPREAD (),* принимающей название файла и указатель на пользовательскую структуру типа BMP.

Внутри функции проверяются возможности чтения файла, такие как имя файла, битность изображения, сжатие, размер. Функция возвращает целое значение, от которого зависит продолжение работы программы.

2.3. Выполнение первой операции

Операция triangle запускает работу нескольких функций. Функция *drawLine()*, основана на алгоритме Брезенхема, вызывается три раза для разных пар точек, тем самым создавая контур треугольника.
Внутри *drawLine()* на каждой итерации вызывается еще одна функция *lineW(),* которая для каждого текущего пикселя отступает на расстояние равное половине заданного в каждую сторону, тем самым придавая линии толщину.
В случае необходимости залить треугольник вызывается функция *Palit(),* которая проходя все точки массива пикселей, определяет их принадлежность области внутри треугольника заданного вершинами и закрашивает их.

2.4. Выполнение второй операции

Операция findrect ищет наибольший прямоугольник заданного цвета и, если такая фигура есть, заменяет цвет на другой. Функция *BIGPIC()* проверяет каждый пиксель на следующие условия:

Удовлетворяет ли следующий по строке от исходного пиксель условию цвета, тогда

Удовлетворяет ли следующий по столбцу от исходного пикселя условию, тогда проверяются пиксели справа от верхнего. Проверка до пикселя, стоящего в столбце на расстоянии которое равно количество шагов вправо, которое к

этому времени успели сделать. Если проверка покажет, что пиксель не того цвета, счетчик по высоте сбрасывается на один назад и количество шагов, сделанных по вертикали и горизонтали записывается в структуру вместе с координатами исходной точки и площадью, при условии, что последняя больше хранящейся в структуре.

Далее по информации, хранящейся в структуре устанавливается координата точки отсчета и количество успешных шагов вверх и в сторону. По этим данным в цикле закрашиваются точки, попадающие в область.

2.5. Выполнение третьей операции

Операция collage создает коллаж из исходного изображения, повторяя его по осям столько раз, сколько было передано аргументами в –X и –Y. Для этого вызывается одноименная функция, внутри которой создается временная структура и ее массив пикселей, растянутый исходя из значений, выбранных пользователем. Туда и записывается коллаж, после чего поля новой структуры перезаписываются в старые.

2.6. Запись файла

Осуществляется при помощи функции *BMPWRITE(),* которая открывает файл с переданным именем, куда записывается поля структуры типа BMP;

ЗАКЛЮЧЕНИЕ

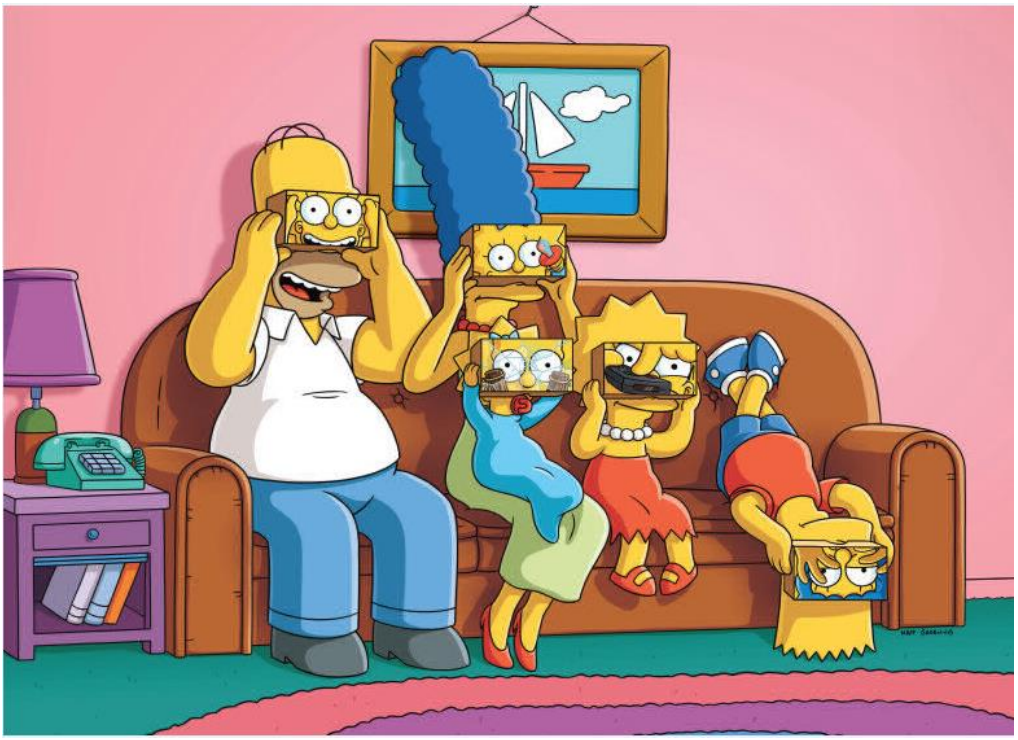
В ходе выполнения курсовой работы была создана программа для обработки BMP файлов и имеющая CLI. Имеются средства для предотвращения неправильного ввода. При правильном вводе программа выполняет поставленную задачу. Результат работы соответствует техническому заданию.

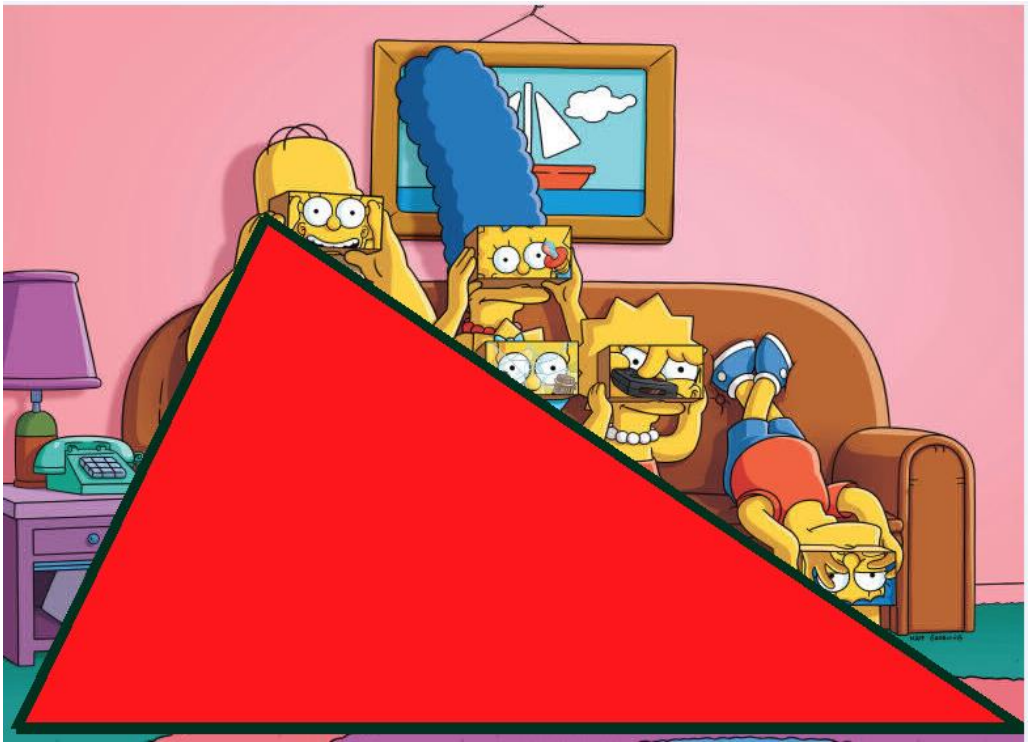
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://ru.wikipedia.org/wiki/BMP - структура файла BMP;
2. https://ru.wikibooks.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%91%D1%80%D0%B5%D0%B7%D0%B5%D0%BD%D1%85%D1%8D%D0%BC%D0%B0 – реализация алгоритма Брезенхэма.
3. http://cppstudio.com/ - общая и вспомогательная информация.

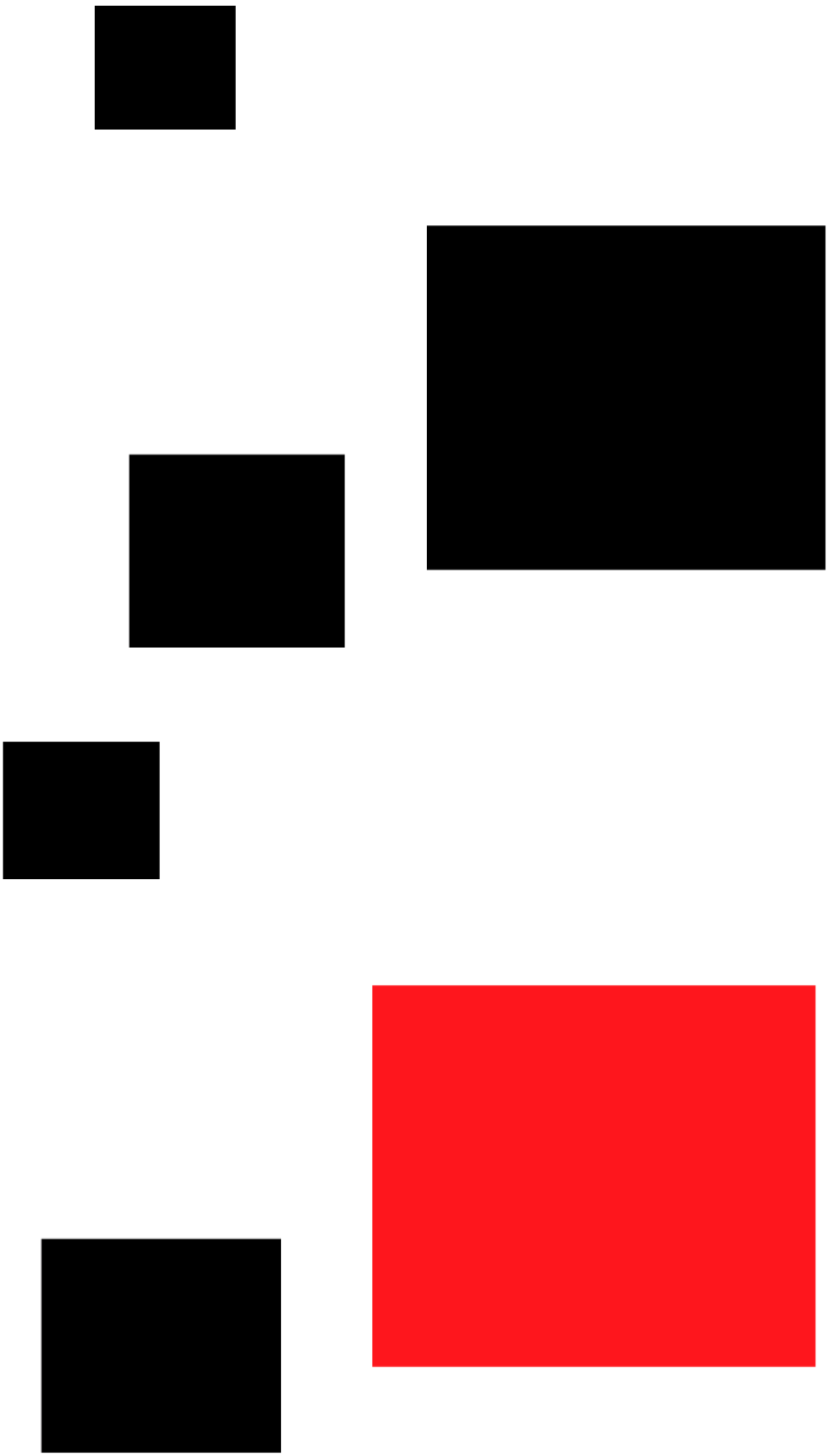
ПРИЛОЖЕНИЕ А
РАБОТА ПРОГРАММЫ

```
'>prog triangle simpsonsvr.bmp out.bmp -C 255.0.0  -A 10;10 -B 200;400 -D 779;10 -V 10 -c 0.45.24 -f
```





```
'>prog rectfind bo.bmp bo2.bmp -c 0.0.0 -C 255.0.0
```



```
>prog collage simpsonsvr.bmp simp.bmp -Y 3 -X 4
```

ПРИЛОЖЕНИЕ Б
КОД ПРОГРАММЫ

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <conio.h>
#include <locale.h>
#include <getopt.h>
#include <string.h>

#pragma pack (push,1)

typedef struct{
    uint16_t signature;
    uint32_t filesize;
    uint16_t reserved1;
    uint16_t reserved2;
    uint32_t pixelArrOffset;
} BitmapFileHeader;

typedef struct{
    uint32_t headerSize;
    uint32_t width;
    uint32_t height;
    uint16_t planes;
    uint16_t bitsPerPixel;
    uint32_t compression;
    uint32_t imageSize;
    uint32_t xPixelsPerMeter;
    uint32_t yPixelsPerMeter;
    uint32_t colorsInColorTable;
    uint32_t importantColorCount;
} BitmapInfoHeader;

typedef struct {
    uint8_t b;
    uint8_t g;
    uint8_t r;
} Rgb;

typedef struct{
    BitmapFileHeader fileHead;
    BitmapInfoHeader infoHead;
    Rgb** pixels;
} BMP;

#pragma pack(pop)


typedef struct
{
    uint32_t x;
    uint32_t y;
    uint32_t I;
    uint32_t J;
    uint64_t sqr;
} SQR;

void HELP(){
```

```
    wprintf(L"\n\nДля запуска программы использовать следующий порядок
аргументов:\n 'Program   Function    Readfile    Outfile  //   Flags\n\n");
    wprintf(L"-------------Доступные функции------------- \n");
    wprintf(L"triangle: рисование трегольника\n");
    wprintf(L"флаги:\n-A/--Apoint; -B/--Bpoint; -D/Cpoint: аргумент вида
число;число -координаты вершин\n");
    wprintf(L"-c/--1color; -C/--2color : цвет 1 и 2; аргумент вида
число.число.число\n");
    wprintf(L"-V/--lWidth: толщина линии; аргумент вида число\n");
    wprintf(L"-f/--Fill заливка\n\n" );
    wprintf(L"rectfind: поиск прямоуголька заданного цвета; смена цвета\n ");
    wprintf(L"флаги:\n ");
    wprintf(L"-c/--1color; -C/--2color : цвет 1 и 2; аргумент вида
число.число.число\n\n");
    wprintf(L"collage: коллаж nxm\n ");
    wprintf(L"флаги:\n ");
    wprintf(L"-X/--xPic -Y/--yPic число\n ");
    getch();
}

void info(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 BMPREAD(char* filename, BMP* picture){
    FILE* file=fopen(filename, "rb");
    if(!file){
        wprintf(L"Кажется, я не могу открыть этот файл: %hs", filename);
        return 0;
    }
    fread(&(picture->fileHead),1,sizeof(BitmapFileHeader), file);
    if(picture->fileHead.signature != 0x4d42){
        wprintf(L"Кажется, Ваш файл не поддерживается");
        return 0;
    }
    fread(&(picture->infoHead),1,sizeof(BitmapInfoHeader), file);
    if(picture->infoHead.headerSize != 40){wprintf(L"Версия BMP не
поддерживается/Необхожимая версия-3"); return 0;}
    if(picture->infoHead.bitsPerPixel != 24){wprintf(L"Только 24 битные
изображения!!!"); return 0;}
    if(picture->infoHead.compression !=0){wprintf(L"Кажется, Ваш файл использует
сжатие."); return 0;}
    uint32_t H=picture->infoHead.height;
    uint32_t W=picture->infoHead.width;
    if(H>65535 || W>65535){wprintf(L"Кажется, изображение слишком большое");
return 0;}
    picture->pixels=malloc(H*sizeof(Rgb*));
```

```
    for(int i=0; i<H; i++){
        picture->pixels[i]=malloc(W * sizeof(Rgb)+(4-(W*sizeof(Rgb))%4)%4);
        fread(picture->pixels[i],1,W*sizeof(Rgb)+(4-(W*sizeof(Rgb))%4)%4,file);
    }
    fclose(file);
    return 1;
}

void setPixel(uint16_t x, uint16_t y, Rgb*** mas, Rgb* color){
    (*mas)[x][y].b=color->b;
    (*mas)[x][y].g=color->g;
    (*mas)[x][y].r=color->r;
}

int BIGPIC(uint8_t R, uint8_t G, uint8_t B, Rgb*** mas,unsigned int H,unsigned
int W, Rgb* color){

    SQR Str={0, 0, 0, 0, 0};

    for(int i=0; i<H; i++){
        for(int j=0; j<W; j++){
            if(((*mas)[i][j].b==B && (*mas)[i][j].g==G &&(*mas)[i][j].r==R)){
                uint16_t i1=i;
                uint16_t j1=j;
                uint16_t countJ = 0;
                uint16_t  count =1;

                while(j1<W || i1<H){
                    if(((*mas)[i][j1+1].b==B && (*mas)[i][j1+1].g==G
&&(*mas)[i][j1+1].r==R) && (j1+1<W)){
                        j1++;
                    }
                    else{
                        if((i1-i)*(j1-j)>Str.sqr){
                            Str.x=i;
                            Str.y=j;
                            Str.I=i1-i;
                            Str.J=j1-j;
                            Str.sqr=(i1-i)*(j1-j);
                        }
                        break;
                    }

                    if(((*mas)[i1+1][j].b==B && (*mas)[i1+1][j].g==G
&&(*mas)[i1+1][j].r==R) && (i1+1<H)){
                        i1++;
                    }
                    if(count<=j1-j){
                        if (((*mas)[i][j + count].b == B &&
                             (*mas)[i][j + count].g == G &&
                             (*mas)[i][j + count].r == R) &&
                            (j1 + 1 < W)){
                            countJ++;
                        }
                        count++;
                    }

                    if(countJ==count){
                        if((i1-i)*(j1-j)>=Str.sqr){
                            Str.x=i;
                            Str.y=j;
                            Str.I=i1-i;
                            Str.J=j1-j;
```

```
                            Str.sqr=(i1-i)*(j1-j);
                        }
                        break;
                    }
                }
            }
        }

    printf("%d %d %d %d %d",Str.sqr, Str.I, Str.J, Str.x, Str.y);
    //if(Str.x==0 || Str.y==0 || Str.I==0 || Str.J==0 || Str.sqr==0 ){
    if(Str.I==0 || Str.J==0 || Str.sqr==0 ){
        wprintf(L"Ошибка, не могу найти заданую область"); return -1;
    }
    uint16_t a=0;
    uint16_t b;
    while(a<=Str.I){
        b=0;
        while(b<=Str.J){
            setPixel(Str.x+a, Str.y+b, mas, color);
            b++;
        }
        a++;
    }
    return 0;
}

void lineW(uint16_t x,uint16_t y, Rgb*** mas,uint8_t width, unsigned int H,
unsigned int W, Rgb* color){
    uint16_t i=0;
    uint16_t x1=x;
    while(i<width/2 && x1<H && x1>=0){
        setPixel(x1,y,mas, color);
        i++;
        x1++;
    }
    i=0;
    x1=x;
    while(i<width/2 && x1<H && x1>=0){
        setPixel(x1,y,mas, color);
        i++;
        x1--;
    }
    i=0;
    uint16_t y1=y;
    while(i<width/2 && y1<W && y1>=0){
        setPixel(x,y1,mas, color);
        i++;
        y1++;
    }
    i=0;
    y1=y;
    while(i<width/2 && y1<W && y1>=0){
        setPixel(x,y1,mas, color);
        i++;
        y1--;
    }
}

void Palit(uint16_t x1, uint16_t y1,uint16_t x2,uint16_t y2, uint16_t
x3,uint16_t y3, Rgb*** mas, unsigned int H, unsigned int W, Rgb* color){
    for(int i=0; i<H; i++){
        for(int j=0; j<W; j++){
```

```
            if((((x1-i)*(y2-y1)-(x2-x1)*(y1-j)) >0 &&
                ((x2-i)*(y3-y2)-(x3-x2)*(y2-j))>0 &&
                ((x3-i)*(y1-y3)-(x1-x3)*(y3-j))>0)
               || (((x1-i)*(y2-y1)-(x2-x1)*(y1-j)) <0 &&
                   ((x2-i)*(y3-y2)-(x3-x2)*(y2-j))<0 &&
                   ((x3-i)*(y1-y3)-(x1-x3)*(y3-j))<0))
            {
                setPixel(i,j,mas, color);
            }
        }
    }

}

void drawLine(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, Rgb***
mas,uint8_t width, unsigned int H, unsigned int W, Rgb* color) {
    const int16_t deltaX = abs(x2 - x1);
    const int16_t deltaY = abs(y2 - y1);
    const int16_t signX = x1 < x2 ? 1 : -1;
    const int16_t signY = y1 < y2 ? 1 : -1;
    int16_t error = deltaX - deltaY;
    setPixel(x2, y2, mas, color);
    lineW(x2,y2, mas, width, H, W, color);
    while(x1 != x2 || y1 != y2)
    {
        setPixel(x1, y1, mas, color);
        lineW(x1,y1, mas, width, H, W, color);
        int16_t error2 = error * 2;
        if(error2 > -deltaY)
        {
            error -= deltaY;
            x1 += signX;
        }
        if(error2 < deltaX)
        {
            error += deltaX;
            y1 += signY;
        }
    }

}

void BMPWRITE(char* filename, BMP* picture){
    FILE *ff = fopen(filename, "wb");
    fwrite(&(picture->fileHead),1,sizeof(BitmapFileHeader), ff);
    fwrite(&(picture->infoHead),1,sizeof(BitmapInfoHeader), ff);
    unsigned int w = (picture->infoHead.width) * sizeof(Rgb) + ((picture-
>infoHead.width)*3)%4;
    for(int i=0; i<picture->infoHead.height; i++){
        fwrite(picture->pixels[i],1,w,ff);
    }
    fclose(ff);
}

Rgb creatergb(uint8_t blue, uint8_t green, uint8_t red){
    Rgb bgr = {blue, green, red};
    return bgr;
}

int trgb(char* str, Rgb* color){
    int16_t red = -1, green = -1, blue = -1;
    sscanf(str, "%d.%d.%d", &red, &green, &blue);
    sscanf(str, "%d.%d", &red, &green);
```

```
    sscanf(str, "%d", &red);
    if(red < 0 || red > 255 || green < 0 || green > 255 || blue < 0 || blue >
255){
        wprintf(L"Введён некорректный цвет\n"); return 0;
    }else *color = creatergb(blue, green, red);
    return 1;
}

void collage(BMP* picture, uint8_t n, uint8_t m){

    BMP widepic;

    uint32_t W=picture->infoHead.width;
    uint32_t H=picture->infoHead.height;

    widepic.infoHead=picture->infoHead;
    widepic.fileHead=picture->fileHead;

    widepic.infoHead.height=n*H;
    widepic.infoHead.width=m*W;

    if(widepic.infoHead.height>32767 || widepic.infoHead.width> 32767){
        wprintf(L"Большой размер изображения");
        return;
    }

    widepic.pixels=malloc(n*H*sizeof(Rgb*));

    for(int i=0; i<H*n; i++){
        widepic.pixels[i]=malloc(m*W*sizeof(Rgb)+(4-(W*sizeof(Rgb))%4)%4);
    }

    for(int i=0; i<m*W; i++){
        for(int j=0; j<n*H;j++){
            widepic.pixels[j][i]=picture->pixels[j%H][i%W];
        }
    }


    picture->infoHead=widepic.infoHead;
    picture->fileHead=widepic.fileHead;
    picture->pixels=widepic.pixels;

}


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

    setlocale(LC_ALL, "");

    const struct option lineWidth = {"lWidth", required_argument, NULL, 'V'};
    const struct option firstColor = {"1color", required_argument, NULL, 'c'};
    const struct option secondColor = {"2color", required_argument, NULL, 'C'};
    const struct option xPic = {"xPic", required_argument, NULL, 'X'};
    const struct option yPic = {"yPic", required_argument, NULL, 'Y'};
    const struct option APoint = {"Apoint", required_argument, NULL, 'A'};
    const struct option BPoint = {"Bpoint", required_argument, NULL, 'B'};
    const struct option CPoint = {"Cpoint", required_argument, NULL, 'D'};
    const struct option Fill = {"Fill", no_argument, NULL, 'f'};

    BMP picture;
    int key;
    Rgb color1 = {0, 0, 0};
```

```
    Rgb color2 = {0, 0, 0};
    int zalivka = 0;
    int i;
    uint16_t Ax=0;
    uint16_t Ay=0;
    uint16_t Bx=0;
    uint16_t By=0;
    uint16_t Cx=0;
    uint16_t Cy=0;
    char* end;

    char *outfile = argv[3];
    int w=0;
    int x=1;
    int y=1;

    if (argc == 1) {
        HELP();
        return 0;
    }
    else if (!strcmp(argv[1], "help")) {
        HELP();
        return 0;
    }

    else if (!BMPREAD(argv[2], &picture)) {
        return -1;
    }
    else if (!strcmp(argv[1], "help")) {
        HELP();
        return 0;
    }
    else if (!strcmp(argv[1], "info")) {
        info(picture.infoHead);
    }
    else
        if (!strcmp(argv[1], "triangle")) {
            struct option opts[] = {firstColor, secondColor, APoint, BPoint,
CPoint, Fill, lineWidth};
            key = getopt_long(argc, argv, "c:C:A:B:D:fV:", opts, &i);
            while (key != -1) {
                switch (key) {
                    case 'c':
                        trgb(optarg, &color1);
                        break;
                    case 'C':
                        trgb(optarg, &color2);
                        break;
                    case 'A':
                        Ay=atoi(strtok(optarg,";"));
                        Ax=atoi(strtok(NULL,";"));
                        if((Ay >= picture.infoHead.width) || (Ax >=
picture.infoHead.height)){
                            wprintf(L"слишком большие координаты точки A");
                            return 1;
                        }

                        break;
                    case 'B':
                        By=atoi(strtok(optarg,";"));
                        Bx=atoi(strtok(NULL,";"));
                        if((By >= picture.infoHead.width) || (Bx >=
picture.infoHead.height)){
```

```
                            wprintf(L"слишком большие координаты точки B");
                            return 1;
                        }

                        break;
                    case 'D':
                        Cy=atoi(strtok(optarg,";"));
                        Cx=atoi(strtok(NULL,";"));
                        if((Cy >= picture.infoHead.width) || (Cx >=
picture.infoHead.height)){
                            wprintf(L"слишком большие координаты точки C");
                            return 1;
                        }

                        break;
                    case 'f':
                        zalivka = 1;
                        break;
                    case 'V':
                        w=atoi(optarg);
                        break;
                    case '?':
                        wprintf(L" нечитаемый ключ ");
                        break;
                }
                key = getopt_long(argc, argv, "c:C:A:B:D:fV:", opts, &i);
            }
            if (zalivka == 1) {
                Palit(Ax, Ay, Bx, By, Cx, Cy, &(picture.pixels),
picture.infoHead.height, picture.infoHead.width,
                      &color2);
                drawLine(Ax, Ay, Cx, Cy, &(picture.pixels), w,
picture.infoHead.height, picture.infoHead.width,
                         &color1);
                drawLine(Bx, By, Cx, Cy, &(picture.pixels), w,
picture.infoHead.height, picture.infoHead.width,
                         &color1);
                drawLine(Ax, Ay, Bx, By, &(picture.pixels), w,
picture.infoHead.height, picture.infoHead.width,
                         &color1);
                BMPWRITE(outfile, &picture);
                return 0;
            }
            else{
                drawLine(Ax, Ay, Cx, Cy, &(picture.pixels), w,
picture.infoHead.height, picture.infoHead.width,
                         &color1);
                drawLine(Bx, By, Cx, Cy, &(picture.pixels), w,
picture.infoHead.height, picture.infoHead.width,
                         &color1);
                drawLine(Ax, Ay, Bx, By, &(picture.pixels), w,
picture.infoHead.height, picture.infoHead.width,
                         &color1);
                BMPWRITE(outfile, &picture);
                return 0;
            }
        }
        else if (!strcmp(argv[1], "rectfind")) {
            struct option opts[] = {firstColor, secondColor};
            key = getopt_long(argc, argv, "c:C:", opts, &i);
            while (key != -1) {
                switch (key) {
                    case 'c':
```

```
                        trgb(optarg, &color1);
                        break;
                    case 'C':
                        trgb(optarg, &color2);
                        break;
                    case '?':
                        wprintf(L" нечитаемый ключ ");
                        break;
                }
                key = getopt_long(argc, argv, "c:C:", opts, &i);
            }

BIGPIC(color1.r,color1.g,color1.b,&(picture.pixels),picture.infoHead.height,
picture.infoHead.width,&color2);
            BMPWRITE(outfile, &picture);
            return 0;

        }

        else if (!strcmp(argv[1], "collage")) {
            struct option opts[] = {xPic, yPic};
            key = getopt_long(argc, argv, "X:Y:", opts, &i);
            while (key != -1) {
                switch (key) {
                    case 'X':
                        x=atoi(optarg);
                        break;
                    case 'Y':
                        y=atoi(optarg);
                        break;
                    case '?':
                        wprintf(L" нечитаемый ключ ");
                        break;
                }
                key = getopt_long(argc, argv, "X:Y:", opts, &i);
            }
            collage(&picture,x,y);
            BMPWRITE(outfile, &picture);
            return 0;
        }
        else if (!strcmp(argv[1], "write")) {
            BMPWRITE(outfile, &picture);
        }
        else { wprintf(L"Команды %hs не существует\n", argv[1]); }

        return 0;
}
```