**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Обработка BMP-файла

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Виноградов К. А. |
| Преподаватель |  | Берленко Т. А. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| **Студент** Виноградов Кирилл Алексеевич | | |
| **Группа** 6304 | | |
| **Тема работы:** Обработка BMP-файла | | |
| **Исходные данные:**  Требуется написать программу, которая рисует две диагонали толщиной 1 пиксель для квадратной заданной области BMP-файла черным цветом и сохраняет результат в новом файле. | | |
| **Содержание пояснительной записки:**   * Аннотация * Введение * Описание необходимых для обработки BMP-файла функций * Создание makefile * Тестирование работоспособности программы * Заключение * Список источников * Приложение с исходным кодом программы | | |
| **Предполагаемый объем пояснительной записки:**  25 страниц. | | |
| **Дата выдачи задания:** | | |
| **Дата сдачи реферата:** | | |
| **Дата защиты реферата:** | | |
| Студент |  | Виноградов К. А. |
| Преподаватель |  | Берленко Т. А. |

# **АННОТАЦИЯ**

В этой работе описывается процесс создания программы для изменения изображения в формате BMP. Файл считывается и происходит его обработка: проверка корректности файла, запрос координат, проверка корректности вводимых координат, действия над выделенной областью. Затем результат записывается в новый файл формата BMP. В документе приведен полный код программы на языке Си, а также итерации тестирования этой программы.

Оглавление

[**АННОТАЦИЯ** 4](#_Toc484157323)

[Оглавление 5](#_Toc484157324)

[**ВВЕДЕНИЕ** 6](#_Toc484157325)

[Цель работы 6](#_Toc484157326)

[Формулировка задачи 6](#_Toc484157327)

[Общие сведения 6](#_Toc484157328)

[**ТЕОРИЯ** 8](#_Toc484157329)

[Общая структура 8](#_Toc484157330)

[**Ход работы** 9](#_Toc484157331)

[1. Создание элементов программы 9](#_Toc484157332)

[2. Функции проверки 10](#_Toc484157333)

[3. Функции считывания файла. 11](#_Toc484157334)

[4. Функция рисовки диагоналей. 12](#_Toc484157335)

[5. Функция записи результата в новый файл. 12](#_Toc484157336)

[6. Функция main. 13](#_Toc484157337)

[**Работа с Makefile** 15](#_Toc484157338)

[**Тестирование программы.** 16](#_Toc484157339)

[**ЗАКЛЮЧЕНИЕ** 18](#_Toc484157340)

[**СПИСОК ИСТОЧНИКОВ** 19](#_Toc484157341)

[**ПРИЛОЖЕНИЕ** 20](#_Toc484157342)

[*Файл main.c* 20](#_Toc484157343)

[*Файл functions.c* 21](#_Toc484157344)

[*Файл structures.h* 23](#_Toc484157345)

[*Файл functions.h* 24](#_Toc484157346)

# **ВВЕДЕНИЕ**

## Цель работы

Изучение структуры BMP-файла и создание программы для обработки BMP-файла с помощью языка программирование C.

## Формулировка задачи

Требуется написать программу, которая рисует две диагонали толщиной 1 пиксель для квадратной заданной области BMP-файла черным цветом и сохраняет результат в новом файле.

Программа получает параметры из входного потока и должна проверить их корректность (в том числе, что область является квадратом).

Параметры:

* input\_file
* x0
* y0
* x1
* y1
* input\_file - имя BMP файла
* x0 y0 левый верхний угол области (отсчет с точки 0, 0)
* x1 y1 правый нижний угол области

В случае, если заданная область не является квадратной, то:

* не создается выходного в файла
* программа должна вывести строку “The area is not a square shape”.

### Общие сведения

* 24 бита на цвет
* без сжатия
* файл всегда соответствует формату (проверять не нужно)
* мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями.
* все поля стандартных BMP заголовков в выходном файле должны иметь те же значения что и во входном (разумеется кроме тех, которые должны быть изменены).

# **ТЕОРИЯ**

**BMP** (от [англ.](https://ru.wikipedia.org/wiki/Английский_язык) *Bitmap Picture*) — формат хранения растровых изображений, разработанный компанией Microsoft Файлы формата BMP могут иметь расширения .bmp, .dib и .rle.

### Общая структура

Данные в формате BMP состоят из трёх основных блоков различного размера:

1. Заголовок из структуры BITMAPFILEHEADER и блока BITMAPINFO. Последний содержит:
   * Информационные поля.
   * Битовые маски для извлечения значений цветовых каналов (опциональные).
   * Таблица цветов (опциональная).
2. Цветовой профиль (опциональный).
3. Пиксельные данные.

При хранении в файле все заголовки идут с самого первого байта. Пиксельные данные могут находиться на произвольной позиции в файле (она указывается в поле OffBits структуры BITMAPFILEHEADER), в том числе и в удалении от заголовков. Опциональный цветовой профиль появился в версии 5 и он также может свободно располагаться, но его позиция указывается от начала BITMAPINFO (в поле ProfileData).

BITMAPFILEHEADER — 14-байтная структура, которая располагается в самом начале файла. BITMAPINFO в файле идёт сразу за BITMAPFILEHEADER.

Блок BITMAPINFO состоит из трёх частей:

1. Структура с информационными полями.
2. [Битовые маски для извлечения значений цветовых каналов](https://ru.wikipedia.org/wiki/BMP#.D0.9C.D0.B0.D1.81.D0.BA.D0.B8_.D0.BA.D0.B0.D0.BD.D0.B0.D0.BB.D0.BE.D0.B2) (присутствуют не всегда).
3. [Таблица цветов](https://ru.wikipedia.org/wiki/BMP#.D0.A2.D0.B0.D0.B1.D0.BB.D0.B8.D1.86.D0.B0_.D1.86.D0.B2.D0.B5.D1.82.D0.BE.D0.B2) (присутствует не всегда).

В BMP-файле с глубиной цвета 24 бит 3 байта определяют 3 компоненты цвета (RGB). Изображение хранится по точкам, в построчной развертке, начиная с нижней строки картинки.

# **Ход работы**

* + 1. Создание элементов программы

Вначале необходимо считать заголовок файла BITMAPFILEHEADER. Его описывает следующая структура

typedef struct BITMAPFILEHEADER

{

unsigned short signature;

unsigned int fileSize;

unsigned short Reserved1;

unsigned short Reserved2;

unsigned int offsetToPixels;

} BITMAPFILEHEADER;

Данная структура занимает 14 байт в памяти.

Роль каждого поля данной структуры:

1. Поле signature определяет тип файла. По условию нашей задачи тип обрабатываемого файла всегда BMP, значит первыми двумя символами файла являются две буквы “BM”.

2. Поле fileSize отвечает за размер самого файла в байтах.

3. Поля Reserved1 и Reserved2 зарезервированы и должны быть нулями.

4. Поле offsetToPixels показывает, где начинается битовый массив относительно начала файла (от начала структуры BITMAPFILEHEADER), который описывает изображение.

Далее размещена структура BITMAPINFOHEADER, которая объявлена следующим образом:

typedef struct BITMAPINFOHEADER

{

unsigned int DIBSize;

unsigned int width;

unsigned int height;

unsigned short planes;

unsigned short bitCount;

unsigned int compression;

unsigned int sizeImage;

unsigned int xPixPerMeter;

unsigned int yPixPerMeter;

unsigned int colourUsed;

unsigned int colourImportant;

} BITMAPINFOHEADER;

Данная структура занимает 40 байт в памяти. Роль каждого поля структуры BITMAPINFOHEADER:

1. Поле biSize – размер самой структуры.

2. Поле biWidth задает ширину изображения.

3. Поле biHeight задает высоту изображения.

4. Поле biPlanes задает количество плоскостей.

5. Поле biBitCount – количество бит на один пиксель.

6. Поле biCompression обозначает тип сжатия.

7. Поле biSizeImage – размер изображения в байтах.

8. Поля biXPelsPerMeter и biYPelsPerMeter обозначают соответственно горизонтальное и вертикальное разрешение (в пикселях на метр) конечного устройства, на которое будет выводиться битовый массив (растр).

9. Поле biClrUsed определяет количество используемых цветов из таблицы.

10.Поле biClrImportant определяет число цветов, которые необходимы для того, чтобы изобразить рисунок.

Последним элементом представления данных в BMP-файле является битовый массив растрового изображения. Количество байт памяти определяется размерами растра и числом бит на пиксель.

typedef struct RGB

{

char Blue;

char Green;

char Red;

} RGB;

Так как поля структур выравниваются по границе кратной своему же размеру, необходимо применить следующие директивы:

#pragma pack(push, 1) - Устанавливается размер выравнивания в 1 байт.

#pragma pack(pop) - Возвращаются предыдущие настройки выравнивания (применяется после описания структур).

* + 1. Функции проверки

Функция correctionCheck1 проверяет корректность открытия файла. Если файл не был открыт или был открыт некорректно, то программа напишет сообщение о соответствующей ошибке.

int correctionCheck1(FILE \*bmpFile, char\* bmpFileN) {

if (bmpFile==NULL) { //проверка на корректность открытия файла

printf("Fail with %s\n.", bmpFileN);

return 0;

}

return 1;

}

Функция correctionCheck1 проверяет корректность ввода координат и форму выделенного поля.

int correctionCheck2(FILE \*bmpFile, int x0, int y0, int x1, int y1) {

//проверка на корректность координат

if (x0<0 || y0<0 || x1<0 || y1<0 || (x1-x0)<0 || (y0-y1)<0) {

printf("\nFail with coordinates. The entered coordinates are not correct!\n");

fclose(bmpFile);

return 0;

}

/\*проверка на то, чтобы разность x и y координат была нечетной, для корректного

разбиения на 4 равные части\*/

if((x1-x0)!=(y0-y1)) {

//if (((x1-x0)%2!=0) /\*|| ((y0-y1)%2==0)\*/) {

printf("\nFail with coordinates. The area is not a square shape!\n");

fclose(bmpFile);

return 0;

}

return 1;

}

* + 1. Функции считывания файла.

Функция, используя информацию, находящуюся в структурах BMP- файла, считывает информацию о файле и получает данные о растре. Сначала с помощью функций fseek и ftell мы узнаем размер файла. Далее происходит считывание файла в строку и извлечение заголовков. Чтобы правильно извлечь заголовок с информацией о изображении, мы двигаем указатель bmpArr на размер заголовка. Также определяется длина строки. Затем происходит заполнение двумерного массива (растра).

char \*\*bmpScan(FILE \*bmpFile, BITMAPFILEHEADER \*BMhead, BITMAPINFOHEADER \*DIBhead) {

int i,j,k=0;

fseek(bmpFile,0,SEEK\_END);

int bmpSize=ftell(bmpFile);

fseek(bmpFile,0,SEEK\_SET);

char\* bmpArr=(char\*)malloc(sizeof(char)\*bmpSize);

fread(bmpArr, sizeof(char), bmpSize, bmpFile);

\*BMhead=\*((BITMAPFILEHEADER\*)bmpArr);

bmpArr+=sizeof(BITMAPFILEHEADER);

\*DIBhead=\*((BITMAPINFOHEADER\*)bmpArr);

bmpArr-=sizeof(BITMAPFILEHEADER);

bmpArr+=BMhead->offsetToPixels;

int line=3\*(DIBhead->width)+(DIBhead->width%4);

char\*\* pixArr=(char\*\*)malloc(sizeof(char\*)\*DIBhead->height);

for(i=0; i<DIBhead->height; i++) {

pixArr[i]=(char\*)malloc(sizeof(char)\*line);

for (j=0; j<line; j++)

pixArr[i][j]=bmpArr[k++];

}

return pixArr;

}

* + 1. Функция рисовки диагоналей.

Функция принимает в качестве аргументов растр изображения, а также область изображения, которую необходимо изменить. После этого построчно рисуются две пересекающие черные линии толщиной 1 пиксель.

char \*\*Diags(char \*\*pixArr, int x0, int y0, int x1, int y1) {

int i=y1;

int j=x0;

RGB\* pixelsString=NULL;

RGB temp;

for(i; i<=y0; i++) {

pixelsString=(RGB\*)pixArr[i];

for(j; j<=x1; j++) {

temp=pixelsString[j];

temp.Blue=0x00;

temp.Green=0x00;

temp.Red=0x00;

pixelsString[j]=temp;

temp=pixelsString[x1-j];

temp.Blue=0x00;

temp.Green=0x00;

temp.Red=0x00;

pixelsString[x1-j]=temp;

j=x1+1;

}

j=x0+i+1;

}

}

* + 1. Функция записи результата в новый файл.

Функция принимает на вход растр и заголовки соответствующих структур. Создается новый файл, в который записываются ранее упомянутые заголовки. Далее записывается сам растр и нули, необходимые для выравнивания.

char\* createNewBmp(char\*\* pixArr, BITMAPFILEHEADER\* BMhead, BITMAPINFOHEADER\* DIBhead) {

int i,j;

char\* newBmpN=(char\*)malloc(100\*sizeof(char));

printf("Insert the name of new BMP file, please\n");

scanf("%s", newBmpN);

FILE\* newBMPFile=fopen(newBmpN,"wb");

fwrite(BMhead, sizeof(BITMAPFILEHEADER), 1, newBMPFile);

fwrite(DIBhead, sizeof(BITMAPINFOHEADER), 1, newBMPFile);

int realPix=3\*DIBhead->width;

int align=DIBhead->width%4;

for(i=0; i<DIBhead->height; i++) {

fwrite(pixArr[i], sizeof(char), realPix, newBMPFile); //запись растра

for (j=0; j<align; j++)

fputc(0, newBMPFile);

}

fclose(newBMPFile);

return newBmpN;

}

* + 1. Функция main.

В функции main реализовано обращение к определенному BMP-файлу, вызовы вышеназванных функций, а также ввод названия создаваемого файла.

int main() {

char\* bmpFileN=(char\*)malloc(100\*sizeof(char));

char\* newBmpN=(char\*)malloc(100\*sizeof(char));

printf("Insert the BMP file name, please\n");

scanf("%s", bmpFileN);

FILE\* bmpFile=fopen(bmpFileN, "rb");

BITMAPFILEHEADER BMhead;

BITMAPINFOHEADER DIBhead;

if(!correctionCheck1(bmpFile, bmpFileN))

return 0;

int x0=-1,y0=-1,x1=-1,y1=-1;

printf("Coordinates of the left upper corner (x0, y0): ");

scanf("%d %d", &x0, &y0);

printf("Coordinates of the right bottom corner (x1, y1): ");

scanf("%d %d", &x1, &y1);

//Проверка введенных данных

if(!correctionCheck2(bmpFile, x0, y0, x1, y1))

return 0;

char\*\* pixArr=bmpScan(bmpFile, &BMhead, &DIBhead);

fclose(bmpFile);

if (DIBhead.width<(x1+1) || DIBhead.height<(y0+1)) {

printf("Fail with the entered area\n");

return 0;

}

newBmpN=createNewBmp(Diags(pixArr, x0, y0, x1, y1), &BMhead, &DIBhead);

printf("Bmp file /%s/ successfully changed in the same directory with new name /%s/\n\n", bmpFileN, newBmpN);

free(bmpFileN);

free(newBmpN);

return 0;

}

# **Работа с Makefile**

Для удобства работы с программой проект был разделен на следующие файлы:

* main.c – основная функция
* functions.c – функции, обрабатывающие BMP-файл
* functions.h – объявление прототипов функций из “functions.c”
* structures.h – структуры, необходимы для работы с BMP-файлами

Для компиляции проекта был создан следующий makefile:

bmptest: main.o functions.o

gcc main.o -o bmptest functions.o

rm \*.o

main.o: main.c structures.h functions.h

gcc -c main.c

functions.o: functions.c functions.h structures.h

gcc -c functions.c

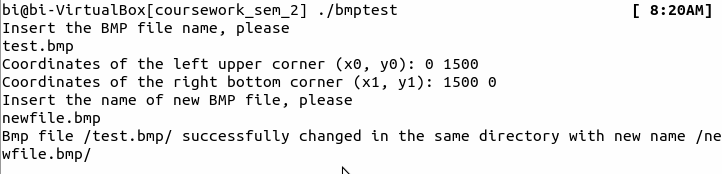
# **Тестирование программы.**

Изображение:

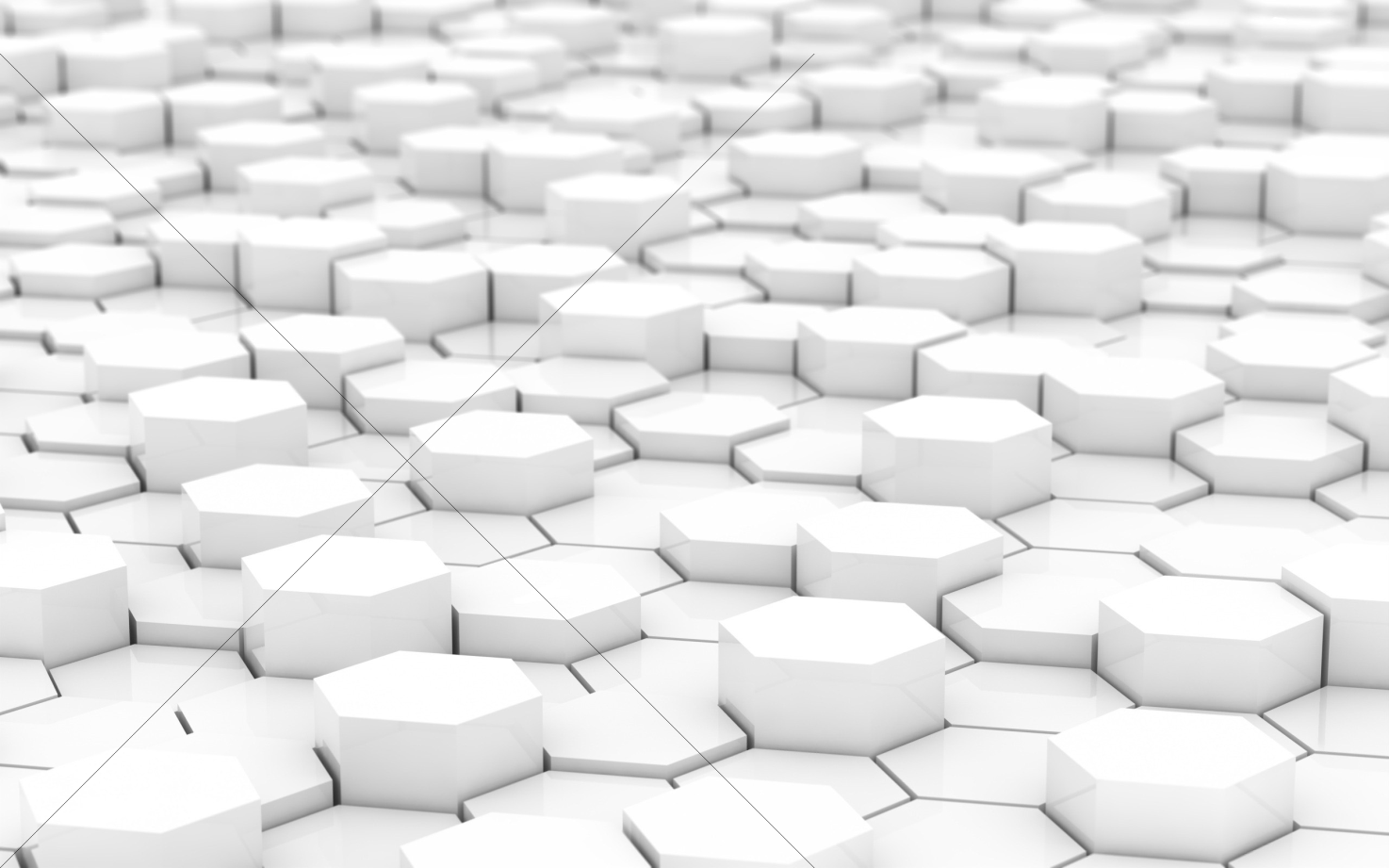


1)

Ввод:

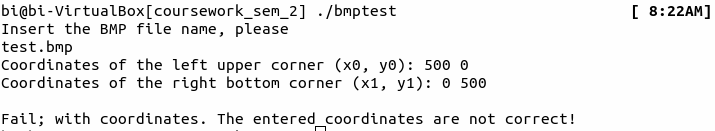


Результат:



2)

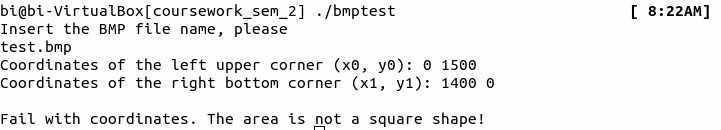
Ввод:



Результат: -

3)

Ввод:



Результат: -

# **ЗАКЛЮЧЕНИЕ**

При выполнении данной работы было закреплено на практике применение структур и стандартных функций для работы с файлами языка Си для обработки растровых изображений, в частности BMP-формата с 24 бит на цвет. Была написана программа, позволяющая перечеркнуть квадратную область изображения и записать это изображение в новый файл. В работе использовались файлы исходного кода(\*.c), заголовочные файлы(\*.h), makefile.

# **СПИСОК ИСТОЧНИКОВ**

1. Керниган Б., Ритчи Д. Язык программирования СИ. СПб.: Издательство "Невский Диалект", 2001. - 352 с.
2. Wicipedia. Bmp. 2016. URL: https://ru.wikipedia.org/wiki/BMP

# **ПРИЛОЖЕНИЕ**

## *Файл main.c*

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include "structures.h"

#include "functions.h"

int main() {

char\* bmpFileN=(char\*)malloc(100\*sizeof(char));

char\* newBmpN=(char\*)malloc(100\*sizeof(char));

printf("Insert the BMP file name, please\n");

scanf("%s", bmpFileN);

FILE\* bmpFile=fopen(bmpFileN, "rb");

BITMAPFILEHEADER BMhead; //заголовок с информацией о файле

BITMAPINFOHEADER DIBhead; //заголовок с информацией о изображении

//проверка открытия файла

if(!correctionCheck1(bmpFile, bmpFileN))

return 0;

int x0=-1,y0=-1,x1=-1,y1=-1;

printf("Coordinates of the left upper corner (x0, y0): ");

scanf("%d %d", &x0, &y0);

printf("Coordinates of the right bottom corner (x1, y1): ");

scanf("%d %d", &x1, &y1);

//Проверка введенных координат

if(!correctionCheck2(bmpFile, x0, y0, x1, y1))

return 0;

/\*Создание и заполнение двумерного массива символов информацией о пикселях bmp изображения\*/

char\*\* pixArr=bmpScan(bmpFile, &BMhead, &DIBhead);

fclose(bmpFile);

if (DIBhead.width<(x1+1) || DIBhead.height<(y0+1)) { //Если введенная область больше самого изображения

printf("Fail with the entered area\n");

return 0;

}

newBmpN=createNewBmp(Diags(pixArr, x0, y0, x1, y1), &BMhead, &DIBhead);

printf("Bmp file /%s/ successfully changed in the same directory with new name /%s/\n\n", bmpFileN, newBmpN);

free(bmpFileN);

free(newBmpN);

return 0;

}

## *Файл functions.c*

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include "structures.h"

#include "functions.h"

int correctionCheck1(FILE \*bmpFile, char\* bmpFileN) {

if (bmpFile==NULL) { //проверка на корректность открытия файла

printf("Fail with %s\n.", bmpFileN);

return 0;

}

return 1;

}

int correctionCheck2(FILE \*bmpFile, int x0, int y0, int x1, int y1) {

//проверка на корректность координат

if (x0<0 || y0<0 || x1<0 || y1<0 || (x1-x0)<0 || (y0-y1)<0) {

printf("\nFail with coordinates. The entered coordinates are not correct!\n");

fclose(bmpFile);

return 0;

}

/\*проверка на то, чтобы разность x и y координат была одинаковой, для выделения квадратной области\*/

if((x1-x0)!=(y0-y1)) {

printf("\nFail with coordinates. The area is not a square shape!\n");

fclose(bmpFile);

return 0;

}

return 1;

}

char \*\*bmpScan(FILE \*bmpFile, BITMAPFILEHEADER \*BMhead, BITMAPINFOHEADER \*DIBhead) {

int i,j,k=0;

/\*Корректное определение размера файла\*/

fseek(bmpFile,0,SEEK\_END); //перемещает указатель в конец файла

/\*возвращает значение указателя текущего положения

(значение соответствующее количеству байт от начала файла)\*/

int bmpSize=ftell(bmpFile);

fseek(bmpFile,0,SEEK\_SET); //перемещает указатель обратно в начало

//fread(&BMhead, sizeof(bmHeader), 1, bmpFile);

//fread(&DIBhead, sizeof(DIBheader), 1, bmpFile);

char\* bmpArr=(char\*)malloc(sizeof(char)\*bmpSize);

fread(bmpArr, sizeof(char), bmpSize, bmpFile); //считывание файла

\*BMhead=\*((BITMAPFILEHEADER\*)bmpArr); //заголовок с информацией о файле

bmpArr+=sizeof(BITMAPFILEHEADER); //указатель сдвигается на размер заголовка

\*DIBhead=\*((BITMAPINFOHEADER\*)bmpArr); //заголовок с информацией о изображении

bmpArr-=sizeof(BITMAPFILEHEADER);

bmpArr+=BMhead->offsetToPixels; //передвигаем указатель на битовый массив, описывающий само изображение

int line=3\*(DIBhead->width)+(DIBhead->width%4); //каждый пиксель кодируется 24 битами(3 байтами) + выравнивание

//заполнение двумерного массива

char\*\* pixArr=(char\*\*)malloc(sizeof(char\*)\*DIBhead->height);

for(i=0; i<DIBhead->height; i++) {

pixArr[i]=(char\*)malloc(sizeof(char)\*line);

for (j=0; j<line; j++)

pixArr[i][j]=bmpArr[k++];

}

return pixArr;

}

char \*\*Diags(char \*\*pixArr, int x0, int y0, int x1, int y1) {

int i=y1;

int j=x0;

RGB\* pixelsString=NULL; //указатель на строку из пикселей

RGB temp;

for(i; i<=y0; i++) {

pixelsString=(RGB\*)pixArr[i]; //char строка приводится к строке пикселей

for(j; j<=x1; j++) {

temp=pixelsString[j];

temp.Blue=0x00;

temp.Green=0x00;

temp.Red=0x00;

pixelsString[j]=temp;

temp=pixelsString[x1-j];

temp.Blue=0x00;

temp.Green=0x00;

temp.Red=0x00;

pixelsString[x1-j]=temp;

j=x1+1;

}

j=x0+i+1;

}

}

/\*char \*\*bmpReflection(char \*\*pixArr, int x0, int y0, int x1, int y1) {

int i,j;

//int x=(x1+x0)/2; //х координата половины области

//int y=(y0+y1)/2; //y координата половины области

int mid=(x1-x0)/2;

//int height=(y0-y1)/2;

RGB\* pixels\_string=NULL; //указатель на строку из пикселей

RGB temp; //указатель на пиксель

for (i=y1; i<=y0; i++) {

pixels\_string=(RGB\*)pixArr[i]; //char строка приводится к строке пикселей

for (j=x0; j<=mid; j++) {

temp=pixels\_string[j];

printf("b:%x\ng:%x\nr:%x\n", temp.Blue, temp.Green, temp.Red);

temp.Blue=0x00;

temp.Green=0x00;

temp.Red=0x00;

pixels\_string[j]=temp;

//pixels\_string[x1-j]=pixels\_string[j];

}

}

return pixArr;

}\*/

char\* createNewBmp(char\*\* pixArr, BITMAPFILEHEADER\* BMhead, BITMAPINFOHEADER\* DIBhead) {

int i,j;

char\* newBmpN=(char\*)malloc(100\*sizeof(char));

printf("Insert the name of new BMP file, please\n");

scanf("%s", newBmpN);

FILE\* newBMPFile=fopen(newBmpN,"wb");

//запись заголовков

fwrite(BMhead, sizeof(BITMAPFILEHEADER), 1, newBMPFile);

fwrite(DIBhead, sizeof(BITMAPINFOHEADER), 1, newBMPFile);

int realPix=3\*DIBhead->width; //число байт содержащих информацию о пикселях

int align=DIBhead->width%4; //число байт для выравнивания

for(i=0; i<DIBhead->height; i++) {

fwrite(pixArr[i], sizeof(char), realPix, newBMPFile); //запись растра

for (j=0; j<align; j++)

fputc(0, newBMPFile); //дописывание нулей

}

fclose(newBMPFile);

return newBmpN;

}

## *Файл structures.h*

#pragma pack(push, 1) //выравнивание в 1 байт

typedef struct bitmap\_file\_header //заголовок файла

{

unsigned short bf\_type; // тип файла

unsigned int bf\_size; //размер файла в байтах

unsigned short bf\_reserved1; //зарезервированы и должны быть нулями

unsigned short bf\_reserved2;

unsigned int bf\_off\_bits; //показывает, где начинается сам битовый массив относительно начала файла

} bitmap\_file\_header;

typedef struct bitmap\_info\_header //структура файла (информация об изображении)

{

unsigned int bi\_size; //размер самой структуры

unsigned int bi\_width; //ширина картинки (в пикселях)

unsigned int bi\_height; //высота картинки (в пикселях)

unsigned short bi\_planes; //количество плоскостей (устанавливается в 1)

unsigned short bi\_bit\_count; //количество бит на один пиксель

unsigned int bi\_compression; //тип сжатия(BI\_RGB, если картинка не сжата)

unsigned int bi\_size\_image; // размер картинки (в байтах), если изображение не сжато, то должен быть записан ноль

unsigned int bi\_X\_pels\_per\_meter; //горизонтальное разрешение (в пикселях на метр) устройства, на который будет выводиться битовый массив (растр)

unsigned int bi\_Y\_pels\_per\_meter; //вертикальное разрешение (в пикселях на метр)

unsigned int bi\_color\_used; //определяет кол-во используемых цветов из таблицы (если значение 0 - то используется максимальное кол-во цветов, которые разрешены значением biBitCount)

unsigned int bi\_color\_important; //кол-во важных цветов, определяет число цветов, которые необходимы для того, чтобы изобразить рисунок (если значение 0 - то все цвета важные)

} bitmap\_info\_header;

#pragma pack(pop)

typedef struct rgb\_triple //структура для описания одного пикселя, 3 байта определяют 3 компонента цвета

{

char rgb\_blue;

char rgb\_green;

char rgb\_red;

} rgb\_triple;

## *Файл functions.h*

// Функции, которая проверяет корректность введенных данных

int correctionCheck1(FILE \*bmpFile, char\* bmpFileN);

int correctionCheck2(FILE \*bmpFile, int x0, int y0, int x1, int y1);

//Функция, считывающая информацию о bmp файле

char \*\*bmpScan(FILE \*bmpFile, BITMAPFILEHEADER \*BMhead, BITMAPINFOHEADER \*DIBhead);

//Функция, изменяющая изображение

char \*\*Diags(char \*\*pixArr, int x0, int y0, int x1, int y1);

//Функция, создающая новый файл и записывающая в него измененное изображение

char\* createNewBmp(char\*\* pixArr, BITMAPFILEHEADER\* BMhead, BITMAPINFOHEADER\* DIBhead);