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

Курсовая работа

по дисциплине «Программирование»

Тема: BMP файл

Студент гр. 6304 _____ Васильев А.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2017

Задание на курсовую работу

Студент Васильев А.А. Иванов И.И.

Группа 6304

Тема работы: BMP файл

Содержание пояснительной записки:

- Аннотация
- Содержание
- Введение (цель работы, формулировка задачи)
- Описание функций работы с BMP файлами
- Тестирование работоспособности программы
- Разбиение на файлы и работа с make
- Заключение
- Список использованных источников
- Приложение

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: _____

Дата сдачи реферата: _____

Дата защиты реферата: _____

Студент _____ Васильев А.А.

Преподаватель _____ Берленко Т.А.

Аннотация

В данной работе описывается процесс создания программы для работы с BMP файлами (открытие, извлечение заголовков, редактирование изображения, создание и сохранение нового файла). Для реализации данных функций, в работе используются широкие возможности языка программирования C (динамическая память, работа с файлами, сложные типы данных, указатели и т.д.).

Содержание

Введение

Цель работы

- Создание функций для чтения и работы с BMP файлами;
- Закрепление знаний о работе с файлами в языке C;
- Закрепление знаний о работе с утилитой “make” и репозиторием “github”.

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 программа получила некорректные параметры, то:

- не создается выходного в файла
- выводится сообщение об ошибке “Fail with <имя параметра>”.

Общие сведения

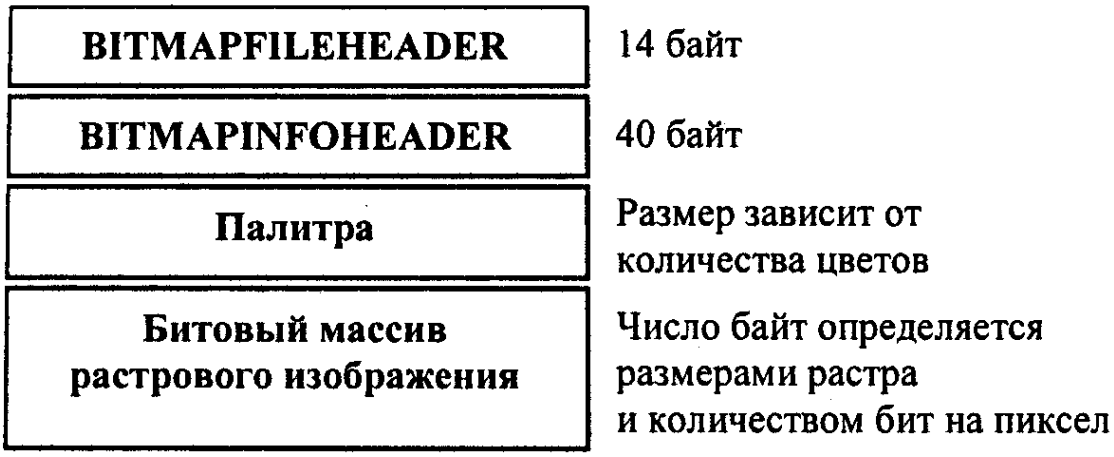
- 24 бита на цвет
- без сжатия
- файл всегда соответствует формату (проверять не нужно)
- обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями.
- обратите внимание на порядок записи пикселей
- все поля стандартных BMP заголовков в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

Ход работы

Создание отдельных элементов проекта

Структура BMP файла

BMP файлы имеют следующую структуру:



Палитра цветов используется в случае, когда используется 8 и меньше бит на цвет.

Структура BITMAPFILEHEADER представлена ниже:

```
typedef struct BITMAPFILEHEADER
{
	unsigned short	bfType; //Тип файла
	unsigned int	bfSize;  //Размер файла
	unsigned short	bfReserved1;  //Зарезервированные поля
	unsigned short	bfReserved2;
	unsigned int bfOffBits;  //Смещения массива пикселей относительно структуры
} BITMAPFILEHEADER;
```

Структура BITMAPINFOHEADER:

```
typedef struct BITMAPINFOHEADER
{
	unsigned int	biSize;   //Размер данной структуры
	unsigned int	biWidth;  //Ширина изображения
	unsigned int	biHeight; //Высота
	unsigned short	biPlanes; //Количество плоскостей
	unsigned short	biBitCount; //Количество бит на один пиксель
	unsigned int	biCompression; //Тип сжатия
	unsigned int	biSizeImage;   //Размер массива пикселей
	unsigned int	biXPelsPerMeter;   //Кол-во пикселей на метр
	unsigned int	biYPelsPerMeter;
	unsigned int	biClrUsed;	//Кол-во используемых цветов
```

```
	unsigned int    biClrImportant;  //Кол-во важных цветов
} BITMAPINFOHEADER;
```

Структура для описания 1-го пикселя RGBTRIPLE:

```
typedef struct RGBTRIPLE
{
	char  rgbBlue;   //Интенсивность основных цветов
	char  rgbGreen;
	char  rgbRed;
} RGBTRIPLE;
```

Структура BITMAPFILEHEADER должна занимать 14 байт, а BITMAPINFOHEADER – 40, но из-за выравнивания данные структуры занимают большее количество байт. Для решения данной проблемы требуется отменить выравнивание с помощью директивы **#pragma pack (push, 1).**

Проверка корректности введенных данных

Для контроля за введенными данными создана отдельная функция, которая проверяет их на наличие ошибок.

```
int input_errors_checker(FILE* bmp_file, int x0, int y0, int x1, int y1)
{
/*Если файл не открыт, то указатель обнуляется.
В этом случае возвращается 0 (критерий ошибки)*/
	if (bmp_file == NULL)
	{
		printf("Fail with input file\n");
		return 0;
	};
/*Проверки на корректность введенных координат.
В случае обнаружения ошибки, возращается 0, а также
закрывается открытый файл.*/
	if (x0<0 || x1<0 || y0<0 || y1<0)
	{
		printf("Fail with coordinates\n");
		fclose(bmp_file);
		return 0;
	};
	if ((y0-y1)!=(x1-x0) || (y0-y1) < 0 || (x1-x0) < 0)
	{
		printf("Fail with area\n");
```

```
            fclose(bmp_file);
            return 0;
        }
        return 1;
}
```

Работа с BMP файлами

Как видно из структуры BMP файла, в заголовочных структурах содержится важная информация, необходимая для корректного отображения. Для извлечения из входного файла заголовков и непосредственно самого массива пикселей создана отдельная функция.

```
char** get_headers_and_raster(FILE* bmp_file, BITMAPFILEHEADER* file_info,
                                                BITMAPINFOHEADER* picture_info)
{
/*Определяется размер файла и выделяется память под строку, куда скопируется файл,
для дальнейшей с ним работы.*/
        fseek(bmp_file,0,SEEK_END);
        int picture_size = ftell(bmp_file);
        fseek(bmp_file,0, SEEK_SET);

        char* array_for_extract = (char*)malloc(sizeof(char)*picture_size);
        char* copy_of_arr = array_for_extract;

/*Файл считывается в выделенную память.*/
        fread(array_for_extract, sizeof(char), picture_size, bmp_file);

/*Заголовок с информацией о файле считывается в отдельную переменную.
Указатель в строке сдвигается на размер заголовка.*/
        *file_info = *((BITMAPFILEHEADER*)array_for_extract);
        array_for_extract += sizeof(BITMAPFILEHEADER);

/*Аналогично считывается информация об изображении. Затем указатель перемещается
на начало самого изображения (растра).*/
        *picture_info = *((BITMAPINFOHEADER*)array_for_extract);

        array_for_extract -= sizeof(BITMAPFILEHEADER);
        array_for_extract+=file_info->bfOffBits;

/*Согласно с полученной из заголовков информацией, выделяется память под
строки (каждый пиксель кодируется 3 байтами) с непосредственной информацией
о пикселях. Также в каждой строке выделяется память, которая необходима для
корректного выравнивания.*/
        int string_len = 3*picture_info->biWidth + (picture_info->biWidth%4);
        char** raster = (char**)malloc(sizeof(char*)*picture_info->biHeight);
```

```
	int k = 0;
	/*Двумерный массив заполняется информацией*/
	for(int i = 0; i < picture_info->biHeight; i++)
	{
		raster[i] = (char*)malloc(sizeof(char)*string_len);
		for (int e = 0; e < string_len; e++)
		{
			raster[i][e] = array_for_extract[k++];
		}
	}
	free(copy_of_arr); //Освобождается память для строки
	return raster;
}
```

Далее в массиве пикселей, в заданной области, необходимо нарисовать крест. Ниже приведена функция для этого.

```
char** write_cross(char** raster, int x0, int y0, int x1, int y1)
{
	int black_point_a = x0;
	int black_point_b = x1;
	RGBTRIPLE* string_for_work = NULL; //Указатель на пиксель (строку из пикселей)
	RGBTRIPLE black_pixel = {0,0,0};//Пиксель черного цвета
/*Цикл "бегает" только в пределах введеных координат*/
	for (int i = y1; i <= y0; i++)
	{
/*Указатель на строку из символов приводится к указателю на строку из пикселей*/
		string_for_work = (RGBTRIPLE*)raster[i];

/*Цикл ставит две точки в каждой строке и тем самым
рисует крест*/
		for(int k = x0; k <= x1; k++)
		{
			if(k==black_point_a || k==black_point_b)
			string_for_work[k] = black_pixel;
		}
			black_point_a++;
			black_point_b--;
	}
	return raster;

}
```

Сохранение нового файла

После совершения необходимых манипуляций с открытым изображением, его необходимо сохранить в новом файле. Для этого создана отдельная функция.

```
void create_new_bmp(char* old_name, char** raster, BITMAPFILEHEADER* file_info,
                                BITMAPINFOHEADER* picture_info)
{
/*К старому имени файла прибавляется строчка "with_cross".
Создается новый файл, с новым именем.*/
	*(strchr(old_name, '.')) = '\0';
	strcat(old_name, "_with_cross.bmp");
	FILE* new_bmp_file = fopen(old_name,"wb");

/*В новый файл записываются заголовки, переданные функции*/
	fwrite(file_info,sizeof(BITMAPFILEHEADER), 1, new_bmp_file);
	fwrite(picture_info,sizeof(BITMAPINFOHEADER),1,new_bmp_file);

/*Определяется число байт в строчке, которые содержат информацию
о пикселях изображения. Также определяется количество байт, необходимых
для выравнивания. Данные байты должны содержать нули.*/
	int part_with_pixels = 3*picture_info->biWidth;
	int part_with_garbage = picture_info->biWidth%4;

/*В новый файл построчно записывается растр изображения.
Также в каждой строке в конце дописывается необходимое число
нулевых байт, необходимых для выравнивания.*/
	for(int i = 0; i < picture_info->biHeight; i++)
	{
		fwrite(raster[i], sizeof(char),part_with_pixels, new_bmp_file);
		for (int k = 0; k < part_with_garbage; k++)
			fputc(0, new_bmp_file);
	}
	fclose(new_bmp_file); //Новый файл закрывается.
}
```

Сохранение нового файла

В функции main описанные выше функции вызываются в нужном порядке. Также в ней реализован интерфейс ввода данных для работы программы

```
int main()
{
	system("clear");
```

```
/*Выделяется память под строку и считывается имя файла и
координаты левого верхнего и правого нижнего углов,
в пределах которых будет нарисован крест.*/
	char* bmp_file_name = (char*)malloc(sizeof(char)*NAME_SIZE);
	printf("Write name of an existing bmp file: ");
	fgets (bmp_file_name, 100, stdin);
	*(strchr(bmp_file_name, '\n')) = '\0';

	int x0,x1,y0,y1;
	x0 = x1 = y0 = y1 = -1;
	printf("Top left coordinates (x, y): ");
	scanf("%d %d", &x0, &y0);
	printf("Bottom right coordinates (x, y): ");
	scanf("%d %d", &x1, &y1);

/*Специальной функцией проверяется корректность введенных данных
(реализация в "functions.c")*/
	FILE* bmp_file = fopen(bmp_file_name, "rb");
	if (input_errors_checker(bmp_file, x0,y0,x1,y1) == 0)
	{
		free(bmp_file_name);
		return 0;
	}

/*Создается двумерный массив символов и заполняется информацией о пикселях
bmp картинки. Также выделяются заголовки с информацией о файле и изображении.
Реализация функции в "functions.c"*/
	BITMAPFILEHEADER file_info;
	BITMAPINFOHEADER picture_info;
	char** raster = get_headers_and_raster(bmp_file, &file_info, &picture_info);
	fclose(bmp_file); //Закрытие файла.


/*Дополнительная проверка на основании полученных данных об изображении.
Проверка на то, вмещается ли введенная область в рамки изображения.*/
	if (picture_info.biWidth < x1+1 || picture_info.biHeight < y0+1)
	{
		printf("Fail with size of picture\n");
		free(bmp_file_name);
		return 0;
	}

/*Специальная функция, которая рисует крест, в заданной области.
Реализация в "functions.c"*/
	raster = write_cross(raster, x0, y0, x1,y1);
```

```
/*Функция создает новый bmp файл с нарисованным крестом.
Сохраняет его под именем "'старое имя'_with_cross.bmp".
Реализация функции в "functions.c"*/
	create_new_bmp(bmp_file_name,raster, &file_info, &picture_info);
	printf("\nNew bmp fie \"%s\" created\n\n", bmp_file_name);

	free(bmp_file_name); //Память, выделенная под строку, освобождается

	return 0;
}
```

Работа с “make” и сборка проекта

Для удобства дальнейшей работы с проектом рационально разбить его на несколько файлов:

- main.c – ввод данных, вызов основных функций;
- functions.c – реализация необходимых функций;
- functions.h – объявления прототипов функций из “functions.c”;
- bmp_structs.h – объявление необходимых структур.

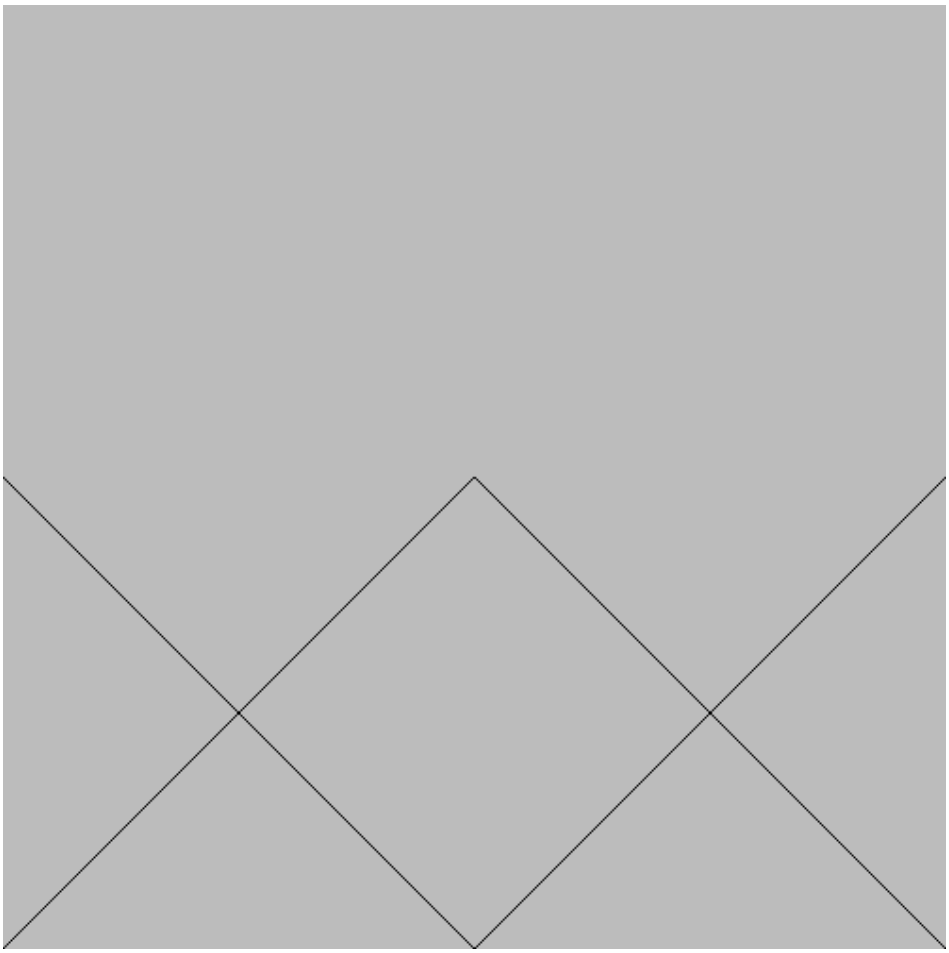
Для быстрой компиляции проекта удобно создать makefile:

```
course: main.o functions.o
	g++ main.o -o course.out functions.o
	rm *.o

main.o: main.c bmp_structs.h functions.h
	g++ -c main.c

functions.o: functions.c functions.h bmp_structs.h
	g++ -c functions.c
```

Тестирование программы

1. *Основной функционал*
    1. Командой **make course** проект собирается в исполняемый файл.
    2. На вход подается одноцветное изображение 500x500 пикселей examp_1.bmp
    3. Затем передаются координаты (0, 249) и (249, 0) – крест в левом нижнем углу.

4. Далее полученному файлу передаются координаты (250, 249) и (499, 0) – крест в правом нижнем углу.
5. В итоге получен правильный BMP файл с двумя крестами внизу:



2. *Сохранность заголовков и «мусорные» байты*

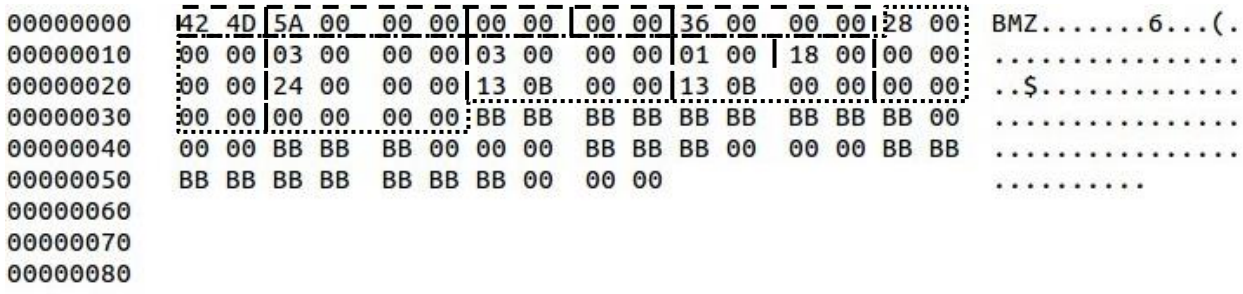
Необходимым условием является заполнение «мусорных», необходимых для выравнивания, байтов нулями.

Также необходимо проверить, что структуры полученного файла сохранены корректно.

Проверку для этого лучше провести на маленьком изображении. В данном случае на одноцветном изображении размером 3x3, в котором в каждой строке в конце должны содержаться 3 лишних байта. Цвет исходного изображения (BB, BB, BB) выбран для удобства отличия нулевых байтов.

1. После открытия файлов в программу посылаются координаты (1, 1) и (1, 1). Тем самым в центре рисуется 1 черная точка.

2. Далее с помощью утилиты **hexedit** проверяется, что содержит каждый байт полученного файла.



Пунктирными линиями отмечены заголовки BMP файла, вертикальными линиями разделены поля структур. Все поля структуры содержат корректную информацию.

Далее идут 9 байтов (BB), то есть 3 пикселя. Далее идут те самые «мусорные» байты, которые успешно обнулены. В других строках происходит то же самое.

Заключение

В ходе выполнения данной работы, были закреплены на практике знания о структурах и файлах, посредством создания функций для работы с ними. Выполнено это было на примере программы, обрабатывающей файл формата BMP, в котором используются специальные структуры данных.

Также было реализовано разбиение проекта на отдельные файлы, что в дальнейшем позволит удобнее добавлять и отлаживать новый функционал.

Список использованных источников

1. Шилдт Г. – Полный справочник по C. – М.: Вильямс, 2004. – 752 с.
2. Wikipedia. – BMP. – 2016. URL: https://ru.wikipedia.org/wiki/BMP
3. Raymond E. - The Lost Art of C Structure Packing. – 2016. URL: http://www.catb.org/esr/structure-packing/

Приложение А. Файлы проекта

Файл bmp_structs.h

```
#pragma pack(push, 1)
typedef struct BITMAPFILEHEADER
{
	unsigned short	bfType;
	unsigned int bfSize;
	unsigned short	bfReserved1;
	unsigned short	bfReserved2;
	unsigned int bfOffBits;
} BITMAPFILEHEADER;

typedef struct BITMAPINFOHEADER
{
	unsigned int	biSize;
	unsigned int	biWidth;
	unsigned int	biHeight;
	unsigned short	biPlanes;
	unsigned short	biBitCount;
	unsigned int	biCompression;
	unsigned int	biSizeImage;
	unsigned int	biXPelsPerMeter;
	unsigned int	biYPelsPerMeter;
	unsigned int	biClrUsed;
	unsigned int	biClrImportant;
} BITMAPINFOHEADER;
#pragma pack(pop)

typedef struct RGBTRIPLE
{
	char  rgbBlue;
	char  rgbGreen;
	char  rgbRed;
} RGBTRIPLE;
```

Файл functions.c

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

#include "bmp_structs.h"
#include "functions.h"

int input_errors_checker(FILE* bmp_file, int x0, int y0, int x1, int y1)
```

```
{
/*Если файл не открыт, то указатель обнуляется.
В этом случае возвращается 0 (критерий ошибки)*/
	if (bmp_file == NULL)
	{
		printf("Fail with input file\n");
		return 0;
	};
/*Проверки на корректность введенных координат.
В случае обнаружения ошибки, возращается 0, а также
закрывается открытый файл.*/
	if (x0<0 || x1<0 || y0<0 || y1<0)
	{
		printf("Fail with coordinates\n");
		fclose(bmp_file);
		return 0;
	};
	if ((y0-y1)!=(x1-x0) || (y0-y1) < 0 || (x1-x0) < 0)
	{
		printf("Fail with area\n");
		fclose(bmp_file);
		return 0;
	}
	return 1;
}

char** get_headers_and_raster(FILE* bmp_file, BITMAPFILEHEADER* file_info,
                                                BITMAPINFOHEADER* picture_info)
{
/*Определяется размер файла и выделяется память под строку, куда скопируется файл,
для дальнейшей с ним работы.*/
	fseek(bmp_file,0,SEEK_END);
	int picture_size = ftell(bmp_file);
	fseek(bmp_file,0, SEEK_SET);

	char* array_for_extract = (char*)malloc(sizeof(char)*picture_size);
	char* copy_of_arr = array_for_extract;

/*Файл считывается в выделенную память.*/
	fread(array_for_extract, sizeof(char), picture_size, bmp_file);

/*Заголовок с информацией о файле считывается в отдельную переменную.
Указатель в строке сдвигается на размер заголовка.*/
	*file_info = *((BITMAPFILEHEADER*)array_for_extract);
	array_for_extract += sizeof(BITMAPFILEHEADER);

/*Аналогично считывается информация об изображении. Затем указатель перемещается
```

```
на начало самого изображения (растра).*/
	*picture_info = *((BITMAPINFOHEADER*)array_for_extract);

	array_for_extract -= sizeof(BITMAPFILEHEADER);
	array_for_extract+=file_info->bfOffBits;

/*Согласно с полученной из заголовков информацией, выделяется память под
строки (каждый пиксель кодируется 3 байтами) с непосредственной информацией
о пикселях. Также в каждой строке выделяется память, которая необходима для
корректного выравнивания.*/
	int string_len = 3*picture_info->biWidth + (picture_info->biWidth%4);
	char** raster = (char**)malloc(sizeof(char*)*picture_info->biHeight);
	int k = 0;
	/*Двумерный массив заполняется информацией*/
	for(int i = 0; i < picture_info->biHeight; i++)
	{
		raster[i] = (char*)malloc(sizeof(char)*string_len);
		for (int e = 0; e < string_len; e++)
		{
			raster[i][e] = array_for_extract[k++];
		}
	}
	free(copy_of_arr); //Освобождается память для строки
	return raster;
}

char** write_cross(char** raster, int x0, int y0, int x1, int y1)
{
	int black_point_a = x0;
	int black_point_b = x1;
	RGBTRIPLE* string_for_work = NULL; //Указатель на пиксель (строку из пикселей)
	RGBTRIPLE black_pixel = {0,0,0};//Пиксель черного цвета
/*Цикл "бегает" только в пределах введеных координат*/
	for (int i = y1; i <= y0; i++)
	{
/*Указатель на строку из символов приводится к указателю на строку из пикселей*/
		string_for_work = (RGBTRIPLE*)raster[i];

/*Цикл ставит две точки в каждой строке и тем самым
рисует крест*/
		for(int k = x0; k <= x1; k++)
		{
			if(k==black_point_a || k==black_point_b)
			string_for_work[k] = black_pixel;
		}
			black_point_a++;
			black_point_b--;
```

```
	}
	return raster;

}

void create_new_bmp(char* old_name, char** raster, BITMAPFILEHEADER* file_info,
                                BITMAPINFOHEADER* picture_info)
{
/*К старому имени файла прибавляется строчка "with_cross".
Создается новый файл, с новым именем.*/
	*(strchr(old_name, '.')) = '\0';
	strcat(old_name, "_with_cross.bmp");
	FILE* new_bmp_file = fopen(old_name,"wb");

/*В новый файл записываются заголовки, переданные функции*/
	fwrite(file_info,sizeof(BITMAPFILEHEADER), 1, new_bmp_file);
	fwrite(picture_info,sizeof(BITMAPINFOHEADER),1,new_bmp_file);

/*Определяется число байт в строчке, которые содержат информацию
о пикселях изображения. Также определяется количество байт, необходимых
для выравнивания. Данные байты должны содержать нули.*/
	int part_with_pixels = 3*picture_info->biWidth;
	int part_with_garbage = picture_info->biWidth%4;

/*В новый файл построчно записывается растр изображения.
Также в каждой строке в конце дописывается необходимое число
нулевых байт, необходимых для выравнивания.*/
	for(int i = 0; i < picture_info->biHeight; i++)
	{
		fwrite(raster[i], sizeof(char),part_with_pixels, new_bmp_file);
		for (int k = 0; k < part_with_garbage; k++)
			fputc(0, new_bmp_file);
	}
	fclose(new_bmp_file); //Новый файл закрывается.
}
```

Файл functions.h

```
int input_errors_checker(FILE*, int, int, int, int);

char** get_headers_and_raster(FILE*, BITMAPFILEHEADER*,BITMAPINFOHEADER*);

char** write_cross(char**, int, int, int, int);

void create_new_bmp(char*, char**, BITMAPFILEHEADER*,BITMAPINFOHEADER*);
```

Файл main.c

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

#include "bmp_structs.h" //Включение заголовочных файлов с объявлением
#include "functions.h"   //структур и функций.
#define NAME_SIZE 100

int main()
{
	system("clear");

/*Выделяется память под строку и считывается имя файла и
координаты левого верхнего и правого нижнего углов,
в пределах которых будет нарисован крест.*/
	char* bmp_file_name = (char*)malloc(sizeof(char)*NAME_SIZE);
	printf("Write name of an existing bmp file: ");
	fgets (bmp_file_name, 100, stdin);
	*(strchr(bmp_file_name, '\n')) = '\0';

	int x0,x1,y0,y1;
	x0 = x1 = y0 = y1 = -1;
	printf("Top left coordinates (x, y): ");
	scanf("%d %d", &x0, &y0);
	printf("Bottom right coordinates (x, y): ");
	scanf("%d %d", &x1, &y1);

/*Специальной функцией проверяется корректность введенных данных
(реализация в "functions.c")*/
	FILE* bmp_file = fopen(bmp_file_name, "rb");
	if (input_errors_checker(bmp_file, x0,y0,x1,y1) == 0)
	{
		free(bmp_file_name);
		return 0;
	}

/*Создается двумерный массив символов и заполняется информацией о пикселях
bmp картинки. Также выделяются заголовки с информацией о файле и изображении.
Реализация функции в "functions.c"*/
	BITMAPFILEHEADER file_info;
	BITMAPINFOHEADER picture_info;
	char** raster = get_headers_and_raster(bmp_file, &file_info, &picture_info);
	fclose(bmp_file); //Закрытие файла.


/*Дополнительная проверка на основании полученных данных об изображении.
```

```
Проверка на то, вмещается ли введенная область в рамки изображения.*/
	if (picture_info.biWidth < x1+1 || picture_info.biHeight < y0+1)
	{
		printf("Fail with size of picture\n");
		free(bmp_file_name);
		return 0;
	}

/*Специальная функция, которая рисует крест, в заданной области.
Реализация в "functions.c"*/
	raster = write_cross(raster, x0, y0, x1,y1);

/*Функция создает новый bmp файл с нарисованным крестом.
Сохраняет его под именем "'старое имя'_with_cross.bmp".
Реализация функции в "functions.c"*/
	create_new_bmp(bmp_file_name,raster, &file_info, &picture_info);
	printf("\nNew bmp fie \"%s\" created\n\n", bmp_file_name);

	free(bmp_file_name); //Память, выделенная под строку, освобождается

	return 0;
}
```

Файл makefile

```
course: main.o functions.o
	g++ main.o -o course.out functions.o
	rm *.o

main.o: main.c bmp_structs.h functions.h
	g++ -c main.c

functions.o: functions.c functions.h bmp_structs.h
	g++ -c functions.c
```