**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Обработка BMP-файла

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6304 |  | Прозорова А. Д. |
| Преподаватель |  | Берленко Т. А. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| **Студентка** Прозорова Анастасия Дмитриевна | | |
| **Группа** 6304 | | |
| **Тема работы:** Обработка BMP-файла | | |
| **Исходные данные:**  Дано bmp изображение глубины цвета 24 бит. Требуется написать программу, которое вырезает изображение, заданное квадратной областью, в bmp-файле 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:** | | |
| Студентка |  | Прозорова А. Д. |
| Преподаватель |  | Берленко Т. А. |

1. **АННОТАЦИЯ**

Данная работа направлена на создание программы для обработки bmp-изображения с помощью языка программирования С, в частности, с использованием структур, указателей и работы с динамической памятью. Процесс написания программы содержит открытие файла, извлечение данных из файла, редактирование изображения, создание и сохранение нового файла. В работе использовались файлы исходного кода(\*.c), заголовочные файлы(\*.h), makefile.

ОГЛАВЛЕНИЕ

[АННОТАЦИЯ 3](#__RefHeading__5436_189884073)

[ВВЕДЕНИЕ 5](#__RefHeading___Toc470274170)

[Цель работы 5](#__RefHeading___Toc470274171)

[Формулировка задачи 5](#__RefHeading___Toc470274172)

[ТЕОРИЯ 7](#__RefHeading___Toc470274174)

[РЕШЕНИЕ ЗАДАЧИ 8](#__RefHeading___Toc470274175)

[1. Описание структур bmp-файла 8](#__RefHeading___Toc470274176)

[2. Описание функции error\_checking 9](#__RefHeading___Toc470274177)

[3. Описание функции get\_bmp\_file\_info 9](#__RefHeading___Toc470274179)

[4. Описание функции create\_new\_bmp 1](#__RefHeading___Toc470274180)

[5. Описание функции main 2](#__RefHeading___Toc470274181)

[3. Создание makefile 3](#__RefHeading___Toc470274187)

[4. Тестирование программы 4](#__RefHeading__3041_2137485679)

[ЗАКЛЮЧЕНИЕ 1](#__RefHeading___Toc470274190)

[СПИСОК ИСТОЧНИКОВ 2](#__RefHeading__7345_189884073)

[ПРИЛОЖЕНИЕ 3](#__RefHeading__5450_189884073)

[Файл main.c 3](#__RefHeading___Toc470274192)

[Файл bmp\_func.c 4](#__RefHeading___Toc470274193)

[Файл bmp\_structs.h 5](#__RefHeading__5452_189884073)

[Файл bmp\_func.h 6](#__RefHeading___Toc470274195)

1. **ВВЕДЕНИЕ**
   1. **Цель работы**

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

* 1. Формулировка задачи

Требуется написать программу, которая вырезает изображение, заданное квадратной областью в BMP-файле и сохраняет результат в новом файле.

Программа получает параметры из входного потока и должна проверить их корректность.

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 isn't correctly defined”.

*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).

1. **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.

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).

Структура BitmapFileHeader занимает 14 байт, структура BitmapInfoHeader — 40 байт.

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.

1. **РЕШЕНИЕ ЗАДАЧИ**
   1. ***1. Описание структур bmp-файла***

*1.1. Заголовок bmp-файла*

**typedef struct** bitmap\_file\_header

{

**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;

*1.2. Структура bmp-файла*

**typedef struct** bitmap\_info\_header

{

**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;

*1.3. Структура для описания одного пикселя (3 байта определяют три компонента цвета)*

**typedef struct** rgb\_triple

{

**char**  rgb\_blue;

**char**  rgb\_green;

**char** rgb\_red;

} rgb\_triple;

Из-за выравния структуры bitmap\_file\_header и bitmap\_info\_header занимают большее количество байт, чем необходимо. Для установки выравнивания в 1 байт применяется директива #pragma pack

* 1. ***2. Описание функции error\_checking***

Функция проверяет корректность введенных координат: является ли введенная область квадратом и положительны ли координаты, - также проверяет, удалось ли открыть нужный файл.

**int** error\_checking(**int** x0, **int** x1, **int** y0, **int** y1, **FILE**\* bmp\_file)

{

**if** (bmp\_file == **NULL**)

{

printf("**Error(fail** **with** **input** **file)**");

**return** 0;

}

**if** (x0 < 0 || x1 < 0 || y0 < 0 || y1 < 0 || (y0 - y1) != (x1 - x0) || (y0 - y1) < 0 || (x1 - x0) < 0)

{

printf("**Error(invalid** **coordinates)**");

**return** 0;

}

**else** **return** 1;

}

* 1. ***3. Описание функции get\_bmp\_file\_info***

Функция, с помощью которой считывается информация из заголовка bmp-файла и из структуры его изображения, в отдельный двумерный массив построчно записывается растр изображения.

**char\*\*** get\_bmp\_file\_info(**FILE\*** bmp\_file, **bitmap\_file\_header\*** bfh, **bitmap\_info\_header\*** bih)

{

*//получаем размер изображения в байтах*

**fseek**(bmp\_file, 0, SEEK\_END);

**int** art\_size = **ftell**(bmp\_file); *//ftell возвращает значение указателя текущего положения потока, для бинарных потоков возвращает кол-во байт от начала файла (в данном случае возвращает размер изображения)*

**fseek**(bmp\_file, 0, SEEK\_SET);

*//считываем файл в отдельную строку*

**char\*** arr = (**char\***)malloc(sizeof(**char**) **\*** art\_size);

**char\*** copy\_arr = arr;

**fread**(arr, sizeof(char), art\_size, bmp\_file);

*//считываем в отдельные переменные информацию о заголовке файла и информацию об изображении*

\*bfh **=** \*((bitmap\_file\_header\*)arr);

arr = arr **+** sizeof(bitmap\_file\_header);

\*bih = \*((bitmap\_info\_header\*)arr);

arr = arr **-** sizeof(bitmap\_file\_header);

*//сдвигаем указатель на начало строки пикселей в файле и считываем ее в двумерный массив*

arr = arr + bfh->bf\_off\_bits;

**char\*\*** pixels = (**char\*\***)malloc(sizeof(**char\***) \* bih->bi\_height);

**int** str\_len = bih->bi\_width **\*** 3 **+** bih->bi\_width **%** 4; *//размер одной строки пикселей + кол-во байт для корректного выравнивания*

**int** i, j, k = 0;

**for**( i = 0; i **<** bih->bi\_height; i**++**)

{

pixels[i] = (**char\***)malloc(sizeof(**char**) **\*** str\_len);

**for** (j = 0; j < str\_len; j**++**)

{

pixels[i][j] = arr[k];

k**++**;

}

}

**free**(copy\_arr);

***return*** pixels;

}

* 1. ***4. Описание функции create\_new\_bmp***

Функция, создающая новый bmp-файл, записывающая в него структуры данного bmp-файла с изменененными полями bi\_width и bi\_height структуры bitmap\_info\_header и построчно матрицу пикселей, обрезанную по заданным координатам.

**void** create\_new\_bmp(**char\*** name, **char\*\*** pixels, **bitmap\_file\_header\*** bfh, **bitmap\_info\_header**\* bih, **int** x0, **int** y0, **int** x1, **int** y1)

{

*//создаем новый файл*

\*(strchr(name, '.')) = '**\0'**;

**strcat**(name, "**\_new.bmp**");

**FILE\*** new\_bmp\_file = fopen(name, "**wb**");

*//записываем в новый файл информацию о заголовке и изображении*

**fwrite**(bfh, sizeof(bitmap\_file\_header), 1, new\_bmp\_file);

**fwrite**(bih, sizeof(bitmap\_info\_header), 1, new\_bmp\_file);

*//записываем в новый файл строку пикселей*

**int** str\_len\_rubbish = bih->bi\_width **%** 4; *//кол-во байт для корректного выравнивания*

**int** i, j;

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

{

**for** (j = (x0)**\***3; j **<** x1\*3; j**++**)

**fputc**(pixels[i][j], new\_bmp\_file);

**for** (j = 0; j < str\_len\_rubbish; j**++**)

**fputc**(0, new\_bmp\_file);

}

**fclose**(new\_bmp\_file);

}

* 1. ***5. Описание функции main***

Функция, в которой производится открытие bmp-файла, считывание координат области, которую нужно будет вырезать из изображения, вызывается функция проверки введенных данных на корректность, вызывается функция считывания информации и массива пикселей из открытого bmp-файла и затем функция, которая создает новый bmp-файл с вырезанным изображением из старого bmp-файла.

**int** main()

{

*//выделяем память под имя файла, вводим имя файла и угловые координаты области, которую нужно будет вырезать из изображения*

**char\*** bmp\_file\_name = (**char\***)malloc(sizeof(**char**) \* 100);

**printf**("**Write** **name\**n");

**fgets** (bmp\_file\_name, 100, stdin);

\*(strchr(bmp\_file\_name, '\**n**')) = **'\0';**

**int** x0, x1, y0, y1, i;

**printf**("**Write** **top** **left** **coordinates\n**");

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

**printf**("**Write bottom right coordinates\n**");

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

*//открываем двоичный файл для чтения и проверяем корректность введенных данных*

**FILE\*** bmp\_file = **fopen**(bmp\_file\_name, "**rb**");

**if** (error\_checking(x0, x1, y0, y1, bmp\_file) == 0)

{

**free**(bmp\_file\_name);

**fclose**(bmp\_file);

**return** 0;

}

*//считываем информацию о файле и закрываем его*

bitmap\_file\_header bfh; // заголовок bmp файла

bitmap\_info\_header bih; // структура изображения

**char\*\*** pixels = **get\_bmp\_file\_info**(bmp\_file, &bfh, &bih); //матрица пикселей

**fclose**(bmp\_file);

*//проверяем, не вылазят ли введенные координаты за рамки изображения*

**if** (bih.bi\_width <= x1-1 || bih.bi\_height < y0-1)

{

**printf**("**Error(size of picture)\n**");

**free**(bmp\_file\_name);

**return** 0;

}

*//создаем новый файл с обрезанным изображением*

bih.bi\_height = y0-y1;

bih.bi\_width = x1-x0;

**create\_new\_bmp**(bmp\_file\_name, pixels, &bfh, &bih, x0, y0, x1, y1);

**printf**("**New bmp\_file %s created\n**", bmp\_file\_name);

**free**(bmp\_file\_name);

**return** 0;

}

* 1. ***3. Создание makefile***

Создаем объектные файлы функций, после компилируем их вместе, затем очищаем объектные файлы после сборки проекта. В отдельные файлы вынесем функции для обработки bmp-файла (bmp\_func.c) и главную функцию main, открывающую bmp-файл и вызывающую функции для его обработки (main.c). Для каждого из этих файлов создадим заголовочные файлы \*.h c прототипами соответствующих функций и заголовочный файл с описанием структур bmp-файла.

*#запись результата в исполняемый файл name*

**main**: bmp\_func.o main.o

**gcc** main.o -o main bmp\_func.o

#очистка

**rm** \*.o

*#cоздание объектного файла main.o*

**main.o** : main.c bmp\_func.h bmp\_structs.h

**gcc** -c main.c

*#cоздание объектного файла bmp\_func.o*

**bmp\_func.o:** bmp\_func.c bmp\_func.h bmp\_structs.h

gcc -c bmp\_func.c

*4. Тестирование программы*

С помощью команды make проект собирается в исполняемый файл. На вход подяется bmp-изображение размером 700 х 525 пикселей.

1. На вход подаются координаты (-5, 300) и (300, -5). Выход: файл не создан, сообщение об ошибке «Error(invalid coordinates)».
2. На вход подаются координаты (34, 342) и (348, 30). Выход: файл не создан, сообщение об ошибке «Error(invalid coordinates)».
3. На вход подаются координаты (50, 600) и (600, 50). Выход: файл не создан, сообщение об ошибке «Error(size of picture)».
4. На вход подаются координаты (0, 525) и (525, 0). Выход: файл 525x525 создан, сообщение «New bmp\_file file\_new.bmp created».

*Рис. 1. Исходное изображение Рис. 2. Итоговое изображение*

5) На вход подаются координаты (300, 400) и (305, 395). Выход: файл 5х5 создан, сообщение «New bmp\_file file\_new.bmp created».

1. **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 обрезать исходное изображение и записать его в новый файл. В работе использовались файлы исходного кода(\*.c), заголовочные файлы(\*.h), makefile.

1. **СПИСОК ИСТОЧНИКОВ**
2. Керниган Б., Ритчи Д. Язык программирования СИ. СПб.: Издательство "Невский Диалект", 2001. - 352 с.
3. Wicipedia. Bmp. 2016. URL: https://ru.wikipedia.org/wiki/BMP
4. **ПРИЛОЖЕНИЕ**
   1. ***Файл main.c***

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "bmp\_structs.h"

#include "bmp\_func.h"

int main()

{

//выделяем память под имя файла, вводим имя файла и угловые координаты области, которую нужно будет вырезать из изображения

char\* bmp\_file\_name = (char\*)malloc(sizeof(char) \* 100);

printf("Write name\n");

fgets (bmp\_file\_name, 100, stdin);

\*(strchr(bmp\_file\_name, '\n')) = '\0';

int x0, x1, y0, y1, i;

printf("Write top left coordinates\n");

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

printf("Write bottom right coordinates\n");

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

//открываем двоичный файл для чтения и проверяем корректность введенных данных

FILE\* bmp\_file = fopen(bmp\_file\_name, "rb");

if (error\_checking(x0, x1, y0, y1, bmp\_file) == 0)

{

free(bmp\_file\_name);

fclose(bmp\_file);

return 0;

}

//считываем информацию о файле и закрываем его

bitmap\_file\_header bfh; // заголовок bmp файла

bitmap\_info\_header bih; // структура изображения

char\*\* pixels = get\_bmp\_file\_info(bmp\_file, &bfh, &bih); //матрица пикселей

fclose(bmp\_file);

//проверяем, не вылазят ли введенные координаты за рамки изображения

if (bih.bi\_width <= x1-1 || bih.bi\_height < y0-1)

{

printf("Error(size of picture)\n");

free(bmp\_file\_name);

return 0;

}

//создаем новый файл с обрезанным изображением

bih.bi\_height = y0-y1;

bih.bi\_width = x1-x0;

create\_new\_bmp(bmp\_file\_name, pixels, &bfh, &bih, x0, y0, x1, y1);

printf("New bmp\_file %s created\n", bmp\_file\_name);

free(bmp\_file\_name);

return 0;

}

* 1. ***Файл bmp\_func.c***

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "bmp\_structs.h"

// Функция, которая проверяет корректность введенных данных

int error\_checking(int x0, int x1, int y0, int y1, FILE\* bmp\_file)

{

if (bmp\_file == NULL)

{

printf("Error(fail with input file)");

return 0;

}

if (x0 < 0 || x1 < 0 || y0 < 0 || y1 < 0 || (y0 - y1) != (x1 - x0) || (y0 - y1) < 0 || (x1 - x0) < 0)

{

printf("Error(invalid coordinates)");

return 0;

}

else return 1;

}

//Функция, считывающая информацию о bmp файле

char\*\* get\_bmp\_file\_info(FILE\* bmp\_file, bitmap\_file\_header\* bfh, bitmap\_info\_header\* bih)

{

//получаем размер изображения в байтах

fseek(bmp\_file, 0, SEEK\_END);

int art\_size = ftell(bmp\_file); //ftell возвращает значение указателя текущего положения потока, для бинарных потоков возвращает кол-во байт от начала файла (в данном случае возвращает размер изображения)

fseek(bmp\_file, 0, SEEK\_SET);

//считываем файл в отдельную строку

char\* arr = (char\*)malloc(sizeof(char) \* art\_size);

char\* copy\_arr = arr;

fread(arr, sizeof(char), art\_size, bmp\_file);

//считываем в отдельные переменные информацию о заголовке файла и информацию об изображении

\*bfh = \*((bitmap\_file\_header\*)arr);

arr = arr + sizeof(bitmap\_file\_header);

\*bih = \*((bitmap\_info\_header\*)arr);

arr = arr - sizeof(bitmap\_file\_header);

//сдвигаем указатель на начало строки пикселей в файле и считываем ее в двумерный массив

arr = arr + bfh->bf\_off\_bits;

char\*\* pixels = (char\*\*)malloc(sizeof(char\*) \* bih->bi\_height);

int str\_len = bih->bi\_width \* 3 + bih->bi\_width % 4; //размер одной строки пикселей + кол-во байт для корректного выравнивания

int i, j, k = 0;

for( i = 0; i < bih->bi\_height; i++)

{

pixels[i] = (char\*)malloc(sizeof(char) \* str\_len);

for (j = 0; j < str\_len; j++)

{

pixels[i][j] = arr[k];

k++;

}

}

free(copy\_arr);

return pixels;

}

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

void create\_new\_bmp(char\* name, char\*\* pixels, bitmap\_file\_header\* bfh, bitmap\_info\_header\* bih, int x0, int y0, int x1, int y1)

{

//создаем новый файл

\*(strchr(name, '.')) = '\0';

strcat(name, "\_new.bmp");

FILE\* new\_bmp\_file = fopen(name, "wb");

//записываем в новый файл информацию о заголовке и изображении

fwrite(bfh, sizeof(bitmap\_file\_header), 1, new\_bmp\_file);

fwrite(bih, sizeof(bitmap\_info\_header), 1, new\_bmp\_file);

//записываем в новый файл строку пикселей

int str\_len\_rubbish = bih->bi\_width % 4; //кол-во байт для корректного выравнивания

int i, j;

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

{

for (j = (x0)\*3; j < x1\*3; j++)

fputc(pixels[i][j], new\_bmp\_file);

for (j = 0; j < str\_len\_rubbish; j++)

fputc(0, new\_bmp\_file);

}

fclose(new\_bmp\_file);

}

* 1. *Файл bmp\_structs.h*

#pragma pack(push, 1)

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;

* 1. *Файл bmp\_func.h*

// Функция, которая проверяет корректность введенных данных

int error\_checking(int x0, int x1, int y0, int y1, FILE\* bmp\_file);

//Функция, считывающая информацию о bmp файле

char\*\* get\_bmp\_file\_info(FILE\* bmp\_file, bitmap\_file\_header\* bfh, bitmap\_info\_header\* bih);

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

void create\_new\_bmp(char\* name, char\*\* pixels, bitmap\_file\_header\* bfh, bitmap\_info\_header\* bih, int x0, int y0, int x1, int y1);