**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 |  | Тимофеев А.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**ЗАДАНИЕ**

**на курсовую работу**

Студент: Тимофеев А.А.

Группа 6304

Тема работы: Обработка BMP-файла

Содержание пояснительной записки:

* Аннотация
* Содержание
* Введение
* Ход работы
* Заключение
* Список литературы
* Приложение

Предполагаемый объем пояснительной записки: не менее 20 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тимофеев А.А.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Берленко Т.А.

# Аннотация

Целью данной работы было написание на языке С программы для обработки BMP-изображения с использованием структур, функций стандартных библиотек и работы с динамической памятью. В программе считывается BMP-файл, проверяется корректность введенных данных, поворот заданной области на 90 градусов по часовой стрелке. Результат работы программы записывается в новый файл.

Оглавление

[Аннотация 3](#_Toc484139878)

[Введение 5](#_Toc484139880)

[Ход работы 6](#_Toc484139881)

[1. Структура BMP-файла 6](#_Toc484139882)

[1.1. Заголовок BMFileHeader 6](#_Toc484139883)

[1.2. Структура BMInfoHeader 6](#_Toc484139884)

[1.3. Структура для описания единичного пикселя 7](#_Toc484139885)

[2. Функции программы 7](#_Toc484139886)

[2.1. Проверка корректности введенных данных 7](#_Toc484139887)

[2.2. Считывание BMP-файла 8](#_Toc484139888)

[2.3. Функция поворота заданной области 9](#_Toc484139889)

[2.4. Функция создания нового BMP-файла 10](#_Toc484139890)

[2.5. Функция Main 11](#_Toc484139891)

[3. Работа с Makefile 12](#_Toc484139892)

[4. Тестирование программы 14](#_Toc484139893)

[4.1. Тест №1: 14](#_Toc484139894)

[4.2. Тест №2: 15](#_Toc484139895)

[4.3. Тест №3: 16](#_Toc484139896)

[4.4. Тест №4: 16](#_Toc484139897)

[Заключение 17](#_Toc484139898)

[Список источников 18](#_Toc484139899)

[Приложение 19](#_Toc484139900)

[Исходный код 19](#_Toc484139901)

[main.c 19](#_Toc484139902)

[functions.h 20](#_Toc484139903)

[functions.c 20](#_Toc484139904)

[BmpStructures.h 24](#_Toc484139905)

[Makefile 25](#_Toc484139906)

# Введение

Требуется написать программу, которая поворачивает квадратную заданную область BMP-файла по часовой стрелке на 90 градусов и сохраняет результат в новом файле.

Программа получает параметры из входного потока и должна проверить их корректность. Параметры: input\_file, x0, y0, x1, y1, где

* 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 файла
* выводится сообщение об ошибке “Fail with <имя параметра>”.

Общие сведения:

* глубина – 24 бита
* 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.
* обратить 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

## Структура BMP-файла

BMP (от англ. Bitmap Picture) — формат хранения растровых изображений, разработанный компанией Microsoft. Файлы формата BMP могут иметь расширения .bmp, .dib и .rle.

Структура BMP-файла в программе описывается следующим образом:

### Заголовок BMFileHeader

|  |
| --- |
| typedef struct BMFileHeader |
| { |  |
| unsigned short btype; //тип файла |  |
| unsigned int bsize; //размер файла в байтах |  |
| unsigned int breserved12; //зарезервирован и должен быть нулем |  |
| unsigned int boffset; //Определяет смещение от начала битов точечного рисунка в структуре BMFileHeader, в байтах |  |
| }BMFileHeader; |  |

### Структура BMInfoHeader

|  |  |
| --- | --- |
| typedef struct BMInfoHeader | |
| { | |  |
| unsigned int size; //размер структуры | |  |
| unsigned int width; //ширина изображения в пикселях | |  |
| unsigned int height; //высота изображения в пикселях | |  |
| unsigned short planes; //Количество плоскостей. Всегда равно 1 | |  |
| unsigned short bitCount; //Глубина цвета в битах на пиксель | |  |
| unsigned int compression; //Тип сжатия. Если компрессия не используется, то флаг имеет значенине BI\_RGB | |  |
| unsigned int sizelmage; //Размер изображения в байтах | |  |
| unsigned int xPelsPerMeter; //Горизонтальное и вертикальное разрешение (в пикселях на метр) соответственно | |  |
| unsigned int yPelsPerMeter; | |  |
| unsigned int clrUsed; //Количество используемых цветов кодовой таблицы | |  |
| unsigned int clrImportant; /\*Количество основных цветов. | |  |
| Определяет число цветов, необходимых для отображения изображения. | |  |
| Если значение поля равно 0, то используются все цвета.\*/ | |  |
|  |
| }BMInfoHeader; | |  |

### Структура для описания единичного пикселя

|  |
| --- |
| typedef struct RGB |
| { |  |
| char red; |  |
| char green; |  |
| char blue; |  |
| } RGB; |  |

Для установки выравнивания в 1 байт применяем директиву:

#pragma pack(push, 1)

После описания структуры BMP-файла возвращаем исходные настройки:

#pragma pack(pop)

## Функции программы

### Проверка корректности введенных данных

Данная функция проверяет введенный файл и координаты. Файл должен быть корректно открыт и не должен быть пустым. Координаты не должны быть отрицательными, а также заданная ими область должна быть квадратной.

|  |
| --- |
| int CheckingErrors(FILE\* input\_file,int x0,int y0,int x1,int y1) |
| { |  |
| if (input\_file == NULL) //Если файл пустой |  |
| { |  |
| printf("Fail with openning input\_file\n"); |  |
| return 0; |  |
| } |  |
| if (x0 < 0 || y0 < 0 || x1 < 0 || y1 < 0 || x1 - x0 < 0 || y0- y1 < 0){ /\*Если координаты отрицательны |  |
| или заданы не правильно\*/ |  |
| printf("Fail with coordinates\n"); |  |
| fclose(input\_file); //Закрываем открытый файл |  |
| return 0; |  |
| } |  |
| if ((x1 - x0) != (y0 - y1)) //Если область не квадратная |  |
| { |  |
| printf("Fail with field\n"); |  |
| fclose(input\_file); //Закрываем открытый файл |  |
| return 0; |  |
| } |  |
| } |  |

### Считывание BMP-файла

Данная функция считывает информацию о файле, находящуюся в структурах, описанных в пункте 1. Чтобы корректно выделить память под файл, узнается его размер. Далее считывается файл, и извлекаются заголовки.

Затем заполняется растр.

|  |
| --- |
| char\*\* LoadBmp(FILE\* input\_file, BMFileHeader\* FileInfo, BMInfoHeader\* ImageInfo) |
| { int i,j,k; |  |
| fseek(input\_file,0,SEEK\_END);//устанавливаем позицию в конец файла |  |
| int BmpSize = ftell(input\_file);//размер файла (Для бинарных потоков, возвращается значение соответствующее количеству байт от начала файла) |  |
| fseek(input\_file,0,SEEK\_SET);//перемещение указателя в начало |  |
|  |  |
| char\* buffer = (char\*)malloc(sizeof(char)\*BmpSize); //Выделяем память под строку для работы с файлом |  |
|  |  |
| fread(buffer,sizeof(char),BmpSize,input\_file); //Считываем файл |  |
| \*FileInfo = \*((BMFileHeader\*)buffer); //Cчитываем заголовок с информацией о файле |  |
| buffer += sizeof(BMFileHeader); //Смещаем указатель |  |
| \*ImageInfo = \*((BMInfoHeader\*)buffer); //Считываем информацию об изображении |  |
| buffer -= sizeof(BMFileHeader); |  |
| buffer += FileInfo -> boffset; //Смещаем указатель на начало изображение |  |
|  |  |
| int len = 3\*ImageInfo->width + (ImageInfo->width%4); //Пиксели кодируются 3мя байтами, выравнивание |  |
| char\*\* raster = (char\*\*)malloc(sizeof(char\*)\*ImageInfo->height); //Заполняем массив |  |
| for(i=0; i < ImageInfo->height; i++) |  |
| { |  |
| raster[i] = (char\*)malloc(sizeof(char)\*len); |  |
| for(j=0; j < len; j++) |  |
| raster[i][j] = buffer[k++]; |  |
| } |  |
| return raster; |  |
| } Функция поворота заданной области Функция принимает заполненный растр и координаты заданной области. После чего поворачивает заданную область на 90 градусов по часовой стрелке. И возвращает измененный растр.   |  | | --- | | char\*\* AreaRotate(char\*\* raster, int x0, int y0, int x1, int y1) | | { |  | | RGB\* str = NULL; |  | | int i,j,k; |  | | RGB\*\* area = (RGB\*\*)malloc(sizeof(RGB\*)\*(y0-y1+1)); //Выделяем память под массив строк |  | | for ( i = 0; i <=y0-y1; i++) //Заполняем двумерный массив |  | | { |  | | area[i] = (RGB\*)malloc(sizeof(RGB)\*(x1-x0+1)); |  | | memmove(area[i], raster[y1+i]+x0\*sizeof(RGB), sizeof(RGB)\*(x1-x0+1)); |  | | } |  | | for ( j = y1; j <= y0; j++) //Ограничиваем область заданными координатами |  | | { |  | | str = (RGB\*)raster[j]; |  | | for( k = x0; k <=x1; k++) |  | | { |  | | str[k]=area[k-x0][y0-j]; //Поворачиваем область на 90 градусов |  | | } |  | | } |  | | free(area); |  | | return raster; |  | | } |  | |  |

### Функция создания нового BMP-файла

Функция создает новый BMP-файл с измененным изображением.

|  |
| --- |
| void NewBmp(char\* input\_file, char\*\* raster, BMFileHeader\* FileInfo, BMInfoHeader\* ImageInfo) |
| { |  |
| int i,j ; |  |
| \*(strchr(input\_file, '.')) = '\0'; |  |
| strcat(input\_file, "rotated.bmp"); |  |
| FILE\* output\_file = fopen(input\_file,"wb"); |  |
| //запись заголовков |  |
| fwrite(FileInfo,sizeof(BMFileHeader), 1, output\_file); |  |
| fwrite(ImageInfo,sizeof(BMInfoHeader),1, output\_file); |  |
| int Pixels = 3\*ImageInfo->width; //Байты под пикслеи |  |
| int Alignment = ImageInfo->width%4; //Байты под выравнивание |  |
| for(i=0; i < ImageInfo->height; i++) |  |
| { |  |
| fwrite(raster[i], sizeof(char), Pixels, output\_file); |  |
| for (j=0; j < Alignment; j++) |  |
| fputc(0, output\_file); |  |
| } |  |
| fclose(output\_file); |  |
| } |  |

### 

### Функция Main

В данной функции реализуется интерфейс для удобства работы с программой, и вызываются другие функции для выполнения задачи программы.

|  |
| --- |
| int main() |
| { |  |
| char\* BmpName = (char\*)malloc(sizeof(char)\*100); |  |
| printf("Enter name of the BMP file: "); |  |
| fgets(BmpName,100,stdin); |  |
| \*(strchr(BmpName,'\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); |  |
| FILE\* input\_file = fopen(BmpName, "rb"); |  |
| if (CheckingErrors(input\_file, x0,y0,x1,y1) == 0) |  |
| { |  |
| free(BmpName); |  |
| return 0; |  |
| } |  |
| BMFileHeader FileInfo; |  |
| BMInfoHeader ImageInfo; |  |
| char\*\* buffer = LoadBmp(input\_file, &FileInfo, &ImageInfo); |  |
| fclose(input\_file); |  |
| if (ImageInfo.width < x1+1 || ImageInfo.height < y0+1) |  |
| { |  |
| printf("Fail with area\n"); |  |
| free(BmpName); |  |
| return 0; |  |
| } |  |
| buffer = AreaRotate(buffer, x0, y0,x1,y1); |  |
| NewBmp(BmpName,buffer, &FileInfo, &ImageInfo); |  |
| printf("Rotated BMP file \"%s\" created\n", BmpName); |  |
| free(BmpName); |  |
| return 0; |  |
| } |  |

## Работа с Makefile

Программа была разбита на следующие файлы:

* main.c – основная функция программы
* BmpStructures.h – структуры для работы с BMP-файлами
* functions.c – функции для работы с BMP-файлами
* functions.h – прототипы функций для работы с BMP-файлами

Был создан следующий Makefile:

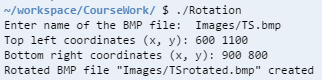
|  |
| --- |
| Rotation: main.o functions.o |
| gcc main.o -o Rotation functions.o |  |
| rm \*.o |  |
| main.o: main.c BmpStructures.h functions.h |  |
| gcc -c main.c |  |
| functions.o: functions.c functions.h BmpStructures.h |  |
| gcc -c functions.c |  |

## Тестирование программы

### Тест №1:



Ввод/Вывод:

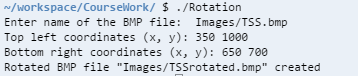


Полученное изображение:



### Тест №2:



Ввод/Вывод:

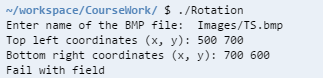
Полученное изображение:



### Тест №3:

Было использовано изображение из теста №1.

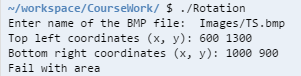
Ввод/Вывод:



(Заданная область вышла за пределы разрешения изображения).

### Тест №4:

Было использовано изображение из теста №1.

Ввод/Вывод:

(Заданная область не является квадратной).

# Заключение

В ходе выполнения данной работы на практике были закреплены умения работать с BMP-файлами, на примере программы, поворачивающей определенную область изображения на 90 градусов по часовой стрелке. Также были использованы навыки создания Makefile-а, полученные в ходе предыдущих работ.

# Список источников

* Википедия. «BMP».

URL: <https://ru.wikipedia.org/wiki/BMP>

01.12.2016г.

* «Работа с файлами в формате BMP»

URL: <http://aco.ifmo.ru/el_books/image_processing/3_06.html>

* Керниган Б., Ритчи Д. Язык программирования Си.\Пер. с англ., 3-е изд., испр. – СПб.: “Невский Диалект”, 2001. – 352 с.

# Приложение

## Исходный код

### main.c

|  |
| --- |
| int main() |
| { |
| char\* BmpName = (char\*)malloc(sizeof(char)\*100); |
| printf("Enter name of the BMP file: "); |
| fgets(BmpName,100,stdin); |
| \*(strchr(BmpName,'\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); |
| FILE\* input\_file = fopen(BmpName, "rb"); |
| if (CheckingErrors(input\_file, x0,y0,x1,y1) == 0) |
| { |
| free(BmpName); |
| return 0; |
| } |
| BMFileHeader FileInfo; |
| BMInfoHeader ImageInfo; |
| char\*\* buffer = LoadBmp(input\_file, &FileInfo, &ImageInfo); |
| fclose(input\_file); |
| if (ImageInfo.width < x1+1 || ImageInfo.height < y0+1) |
| { |
| printf("Fail with area\n"); |
| free(BmpName); |
| return 0; |
| } |
| buffer = AreaRotate(buffer, x0, y0,x1,y1); |
| NewBmp(BmpName,buffer, &FileInfo, &ImageInfo); |
| printf("Rotated BMP file \"%s\" created\n", BmpName); |
| free(BmpName); |
| return 0; |
| } |

### functions.h

|  |
| --- |
| int CheckingErrors(FILE\* input\_file,int x0,int y0,int x1,int y1); |
| char\*\* LoadBmp(FILE\* input\_file, BMFileHeader\* FileInfo, BMInfoHeader\* ImageInfo); |  |
| char\*\* AreaRotate(char\*\* raster, int x0, int y0, int x1, int y1); |  |
| void NewBmp(char\* input\_file, char\*\* raster, BMFileHeader\* FileInfo, BMInfoHeader\* ImageInfo); |  |

### functions.c

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include "BmpStructures.h" |
|  | #include "functions.h" |
|  |  |
|  | int CheckingErrors(FILE\* input\_file,int x0,int y0,int x1,int y1) |
|  | { |
|  | if (input\_file == NULL) //Если файл пустой |
|  | { |
|  | printf("Fail with openning input\_file\n"); |
|  | return 0; |
|  | } |
|  | if (x0 < 0 || y0 < 0 || x1 < 0 || y1 < 0 || x1 - x0 < 0 || y0-y1 < 0){ /\*Если координаты отрицательны |
|  | или заданы не правильно\*/ |
|  | printf("Fail with coordinates\n"); |
|  | fclose(input\_file); //Закрываем открытый файл |
|  | return 0; |
|  | } |
|  | if ((x1 - x0) != (y0 - y1)) //Если область не квадратная |
|  | { |
|  | printf("Fail with field\n"); |
|  | fclose(input\_file); //Закрываем открытый файл |
|  | return 0; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | char\*\* LoadBmp(FILE\* input\_file, BMFileHeader\* FileInfo, BMInfoHeader\* ImageInfo) |
|  | { int i,j,k; |
|  | fseek(input\_file,0,SEEK\_END);//устанавливаем позицию в конец файла |
|  | int BmpSize = ftell(input\_file);//размер файла (Для бинарных потоков, возвращается значение соответствующее количеству байт от начала файла) |
|  | fseek(input\_file,0,SEEK\_SET);//перемещение указателя в начало |
|  |  |
|  | char\* buffer = (char\*)malloc(sizeof(char)\*BmpSize); //Выделяем память под строку для работы с файлом |
|  |  |
|  | fread(buffer,sizeof(char),BmpSize,input\_file); //Считываем файл |
|  | \*FileInfo = \*((BMFileHeader\*)buffer); //Cчитываем заголовок с информацией о файле |
|  | buffer += sizeof(BMFileHeader); //Смещаем указатель |
|  | \*ImageInfo = \*((BMInfoHeader\*)buffer); //Считываем информацию об изображении |
|  | buffer -= sizeof(BMFileHeader); |
|  | buffer += FileInfo -> boffset; //Смещаем указатель на начало изображение |
|  |  |
|  | int len = 3\*ImageInfo->width + (ImageInfo->width%4); //Пиксели кодируются 3мя байтами, выравнивание |
|  | char\*\* raster = (char\*\*)malloc(sizeof(char\*)\*ImageInfo->height); //Заполняем массив |
|  | for(i=0; i < ImageInfo->height; i++) |
|  | { |
|  | raster[i] = (char\*)malloc(sizeof(char)\*len); |
|  | for(j=0; j < len; j++) |
|  | raster[i][j] = buffer[k++]; |
|  | } |
|  | return raster; |
|  | } |
|  |  |
|  | char\*\* AreaRotate(char\*\* raster, int x0, int y0, int x1, int y1) |
|  | { |
|  | RGB\* str = NULL; |
|  | int i,j,k; |
|  | RGB\*\* area = (RGB\*\*)malloc(sizeof(RGB\*)\*(y0-y1+1)); //Выделяем память под массив строк |
|  | for ( i = 0; i <=y0-y1; i++) //Заполняем двумерный массив |
|  | { |
|  | area[i] = (RGB\*)malloc(sizeof(RGB)\*(x1-x0+1)); |
|  | memmove(area[i], raster[y1+i]+x0\*sizeof(RGB), sizeof(RGB)\*(x1-x0+1)); |
|  | } |
|  | for ( j = y1; j <= y0; j++) //Ограничиваем область заданными координатами |
|  | { |
|  | str = (RGB\*)raster[j]; |
|  | for( k = x0; k <=x1; k++) |
|  | { |
|  | str[k]=area[k-x0][y0-j]; //Поворачиваем область на 90 градусов |
|  | } |
|  | } |
|  | free(area); |
|  | return raster; |
|  | } |
|  |  |
|  | void NewBmp(char\* input\_file, char\*\* raster, BMFileHeader\* FileInfo, BMInfoHeader\* ImageInfo) |
|  | { |
|  | int i,j ; |
|  | \*(strchr(input\_file, '.')) = '\0'; |
|  | strcat(input\_file, "rotated.bmp"); |
|  | FILE\* output\_file = fopen(input\_file,"wb"); |
|  | //запись заголовков |
|  | fwrite(FileInfo,sizeof(BMFileHeader), 1, output\_file); |
|  | fwrite(ImageInfo,sizeof(BMInfoHeader),1, output\_file); |
|  | int Pixels = 3\*ImageInfo->width; //Байты под пикслеи |
|  | int Alignment = ImageInfo->width%4; //Байты под выравнивание |
|  | for(i=0; i < ImageInfo->height; i++) |
|  | { |
|  | fwrite(raster[i], sizeof(char), Pixels, output\_file); |
|  | for (j=0; j < Alignment; j++) |
|  | fputc(0, output\_file); |
|  | } |
|  | fclose(output\_file); |
|  | **}** BmpStructures.h  |  | | --- | | typedef struct BMFileHeader | | { | | unsigned short btype; //тип файла | | unsigned int bsize; //размер файла в байтах | | unsigned int breserved12; //зарезервирован и должен быть нулем | | unsigned int boffset; //Определяет смещение от начала битов точечного рисунка в структуре BMFileHeader, в байтах | | }BMFileHeader; | |  | | typedef struct BMInfoHeader | | { | | unsigned int size; //размер структуры | | unsigned int width; //ширина изображения в пикселях | | unsigned int height; //высота изображения в пикселях | | unsigned short planes; //Количество плоскостей. Всегда равно 1 | | unsigned short bitCount; //Глубина цвета в битах на пиксель | | unsigned int compression; //Тип сжатия. Если компрессия не используется, то флаг имеет значенине BI\_RGB | | unsigned int sizelmage; //Размер изображения в байтах | | unsigned int xPelsPerMeter; //Горизонтальное и вертикальное разрешение (в пикселях на метр) соответственно | | unsigned int yPelsPerMeter; | | unsigned int clrUsed; //Количество используемых цветов кодовой таблицы | | unsigned int clrImportant; /\*Количество основных цветов. | | Определяет число цветов, необходимых для отображения изображения. | | Если значение поля равно 0, то используются все цвета.\*/ | | }BMInfoHeader; | | #pragma pack(pop) | | typedef struct RGB | | { | | char red; | | char green; | | char blue; | | } RGB; | |
|  |  |

### 

### Makefile

|  |
| --- |
| Rotation: main.o functions.o |
| gcc main.o -o Rotation functions.o |  |
| rm \*.o |  |
| main.o: main.c BmpStructures.h functions.h |  |
| gcc -c main.c |  |
| functions.o: functions.c functions.h BmpStructures.h |  |
| gcc -c functions.c |  |