**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**Курсовая работа**

**по дисциплине «Программирование»**

**Тема: Обработка изображений в формате BMP.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Киреев К.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2019

|  |  |  |
| --- | --- | --- |
| Студент Киреев К.А. | | |
| Группа 8381 | | |
| Тема работы: Обработка изображений в формате BMP.  Исходные данные: программу требуется реализовать в виде терминального интерфейса. Программа должна принимать аргументы, давать возможность сохранить измененную картинку в другой файл, выполнять функционал по обработке изображений, указанный в задании. | | |
|  | | |
| Содержание пояснительной записки:  «Содержание», «Введение», «Задание работы», «Ход выполнения работы» «Заключение», «Список использованных источников». | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц. | | |
| Дата выдачи задания: 27.02.2019 | | |
| Дата сдачи реферата: | | |
| Дата защиты реферата: | | |
| Студент |  | Киреев К.А. |
| Преподаватель |  | Берленко Т.А. |

**Аннотация**

В ходе выполнения курсовой работы была написана программа, которая редактирует и сохраняет изображение в формате BMP. Обработка осуществляется в соответствии с заданием. Функционал определяется рисованием окружности, заданной двумя способами, ее заливкой, толщиной линий, фильтром rgb-компоненты, который выставляет выбранную компоненту в 0 или 255, а также разделение изображения на n\*m, дополнительными линиями. В работе реализован интерфейс для взаимодействия с пользователем.

**Summary**

In the course of the course work was written a program that edits and saves the image in BMP format. Processing is carried out in accordance with the task. The functional is determined by drawing a circle defined in two ways, its fill, the thickness of the lines, the rgb-component filter, which sets the selected component to 0 or 255, as well as splitting the image into n \* m, additional lines. In work the interface for user interaction is implemented.

**СОДЕРЖАНИЕ**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
| 1. | Задание | 6 |
| 2. | Ход выполнения работы | 7 |
| 2.1. | Структура программы | 7 |
| 2.2. | Считывание картинки | 7 |
| 2.3. | Хранение пикселей картинки | 7 |
| 2.4. | Выравнивание | 7 |
| 2.5. | Создание функции для взаимодействия с пользователем | 8 |
| 2.6. | Создание функций обработки изображения | 9 |
| 2.6.1. | Создание функции для рисования окружности | 9 |
| 2.6.2. | Создание функции для rgb-фильтра | 12 |
| 2.6.3. | Создание функции для разделения изображения n\*m | 13 |
|  | Заключение | 14 |
|  | Список использованных источников | 20 |
|  | Приложение А. Исходный код программы | 21 |
|  |  |  |
|  |  |  |

**ВВЕДЕНИЕ**

**Цель работы**

Написать на языке Си программу, которая реализует различные функции для обработки изображений формата BMP.

**Основные задачи**

Реализация консольного интерфейса с использованием getopt.

Обеспечение стабильной работы с различными форматами BMP, обработка исключительных случаев.

Реализация сохранения изображения в новом файле.

**Методы решения**

Разработка программы велась с помощью операционной системы Windows 10 и с использованием компилятора CodeBlocks. Отладка также производилась при помощи CodeBlocks.

**1. ЗАДАНИЕ**

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 окружности. Окружность определяется:
  + **либо** координатами левого верхнего и правого нижнего угла квадрата, в который она вписана, **либо** координатами ее центра и радиусом
  + толщиной линии окружности
  + цветом линии окружности
  + окружность может быть залитой или нет
  + цветом, которым залита сама окружность, если пользователем выбрана залитая окружность
* (2) Фильтр rgb-компонент. Этот инструмент должен позволять для всего изображения либо установить в 0 либо установить в 255 значение заданной компоненты. Функционал определяется
  + Какую компоненту требуется изменить
  + В какое значение ее требуется изменить
* (3) Разделяет изображение на N\*M частей. Реализация: либо провести линии заданной толщины, тем самым разделив изображение либо сохранение каждой части в отдельный файл. Функционал определяется:
  + Количество частей по “оси” Y
  + Количество частей по “оси” X
  + Толщина линии
  + Цвет линии
  + Либо путь куда сохранить кусочки

# 2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

**2.1. Структура программы**

Вначале прописывается первичное устройство программы. Объявляются заголовочные файлы стандартной библиотеки Си, такие как <stdio.h>, <string.h>, <stdlib.h>, include <unistd.h>, include <stdint.h>.

**2.2. Считывание картинки**

Объявляются структуры, отвечающие за хранение данных об изображении:

RGB \*\*arr;

RGB \*pad;

BITMAPFILEHEADER bmfh;

BITMAPINFOHEADER bmih;

Далее происходит считывание данных с помощью функции fread(), принимающей адрес структуры, в которую надо записать данные, ее размер, кол-во считываемых полей и сам файл, из которого надо производить считывание.

**2.3. Хранение пикселей картинки**

Для выполнения данной задачи создается двумерный массив arr структуры RGB. Сама структура состоит из 3-х элементов, отвечающих за хранение синего, зеленого и красного цвета.

**2.4. Выравнивание**

Для этого создается переменная padding, в которую записывается кол-во пикселей, которые необходимо «дозаполнить», высчитывается переменная по формуле 4 - (длина\_картинки \* 3) % 4.

**2.5. Создание функции для взаимодействия с пользователем**

Когда метод main() вызывается кодом инициализации программы, обработка команд, передающихся в этот метод, уже выполнена. Параметр argc содержит число, соответствующее числу параметров или аргументов, и argv содержит массив указателей на эти аргументы. Под аргументами библиотека подразумевает название программы и все остальные параметры, что следуют за именем команды и отделяются друг от друга пробелом.

|  |
| --- |
| int getopt( int argc, char \*const argv[], const char \*optstring ); |

Получая на вход число параметров команды (argc), массив указателей на эти параметры (argv) и строку с опциями (optstring), getopt() возвращает первый параметр и задает некоторые глобальные переменные. При повторном вызове с теми же аргументами функция вернет следующий параметр и задаст глобальную переменную. Если больше не будет найдено опций, которые надо распознать, то функция вернет -1, что означает завершение обработки команды.

Сначала объявляется строка с опциями optString, которая состоит из одного символа для каждого аргумента командной строки(static const char \*optString = " i?c:y:p:q:b:";).

Ниже приводится полное описание каждой опции:

"-? : Print available commands";

"-i : Print image file information";

"c [b] [g] [r] : Set the color";

"-y [k] [n(0/255)] : Rgb change to 0 or 255";

"-p [n] [m] [thickness] : Divides the image into parts, n \* m";

"-q [ctrX] [ctrY] [rad] [thickness] [filling(1/0)] [b] [g] [r]: Draw a circle - first way; use -c to change the color of the line of this circle";

"-b [X1] [Y1] [X2] [Y2] [thickness] [filling(1/0)] [b] [g] [r]: Draw a circle - second way; use -c to change the color of the line of this circle");

**2.6. Создание функций обработки изображения**

**2.6.1. Создание функции для рисования окружности.**

Функция рисования окружности может быть вызвана двумя способами. При обоих случаях используется функция drawfirstcircle(), которая принимает координаты центра, радиус окружности, толщину контура, и цвет заливки, если таковая необходима. Эти переменные высчитываются при любом способе задания данных об окружности, где могут быть поданы и координаты углов квадрата, в который вписана окружность. Осуществляется проверка корректности цвета. Далее рисуется окружность с помощью функции drawCircle. Затем при необходимости заливки происходит перезапись цвета в каждом пикселе внутри окружности на заданный ранее цвет. Проверка происходит с помощью формулы рисования окружности (x-a)^2 + (y-b)^2 = r^2.

Функция рисования окружности drawCircle работает через алгоритм Брезенхэма для рисования окружностей. Сначала происходит проверка на корректные координаты. По методу построения алгоритм Брезенхэма похож на рисование линии. В этом алгоритме строится дуга окружности для первого квадранта, а координаты точек окружности для остальных квадрантов получаются симметрично. На каждом шаге алгоритма рассматриваются три пикселя, и из них выбирается наиболее подходящий путём сравнения расстояний от центра до выбранного пикселя с радиусом окружности.

// R - радиус, X1, Y1 - координаты центра

int x := 0

int y := R

int delta := 1 - 2 \* R

int error := 0

while (y >= 0)

drawpixel(X1 + x, Y1 + y)

drawpixel(X1 + x, Y1 - y)

drawpixel(X1 - x, Y1 + y)

drawpixel(X1 - x, Y1 - y)

error = 2 \* (delta + y) - 1

if ((delta < 0) && (error <= 0))

delta += 2 \* ++x + 1

continue

if ((delta > 0) && (error > 0))

delta -= 2 \* --y + 1

Алгоритм рисования линии:

Отрезок проводится между двумя точками — (x0, y0) и (x1,y1), где в этих парах указаны столбец и строка соответственно, номера которых растут вправо и вниз. Сначала мы будем предполагать, что наша линия идёт вправо и вниз, причём горизонтальное расстояние x1 – x0 превосходит вертикальное y1 – y0, то есть наклон линии от горизонтали — менее 45°. Наша цель состоит в том, чтобы для каждого столбца x между x0 и x1 определить, какая строка y ближе всего к линии, и нарисовать точку (x, y).

**2.6.2. Создание функции для rgb-фильтра.**

В функцию filter() подается цвет, который нужно изменить, и значение. Для каждого цвета используется одинаковый алгоритм. В двух циклах проходится каждый пиксель изображения и в каждом пикселе меняется заданная компонента на заданное ранее значение[0 или 255]. То есть при значении 0 заданная компонента будет отсутствовать на изображении, что можно легко проверить. И при значении 255 наооборот..

if(strcmp(cv, "r") == 0){

for(int i = 0; i < bmih.biHeight; i++)

for(int j = 0; j < bmih.biWidth; j++)

arr[i][j].blue = (unsigned char)zn;

printf("red value was changed to %u\n", (unsigned char)zn);

}

**2.6.3. Создание функции для разделения изображения n\*m.**

Функция bmpforpieces() принимает количество частей по горизонтали, количество частей по вертикали и толщину контура. Сначала происходит проверка на корректность введенных данных. Проверяется можно ли разделить изображение на целые части с заданными параметрами, иначе выводится соответствующее сообщение. В двух циклах проверяется каждый пиксель на корректность для вызова функции рисования линии. При значении оси x, которое является делителем значения ширины, деленной на количество частей, рисуется линия до верхнего пикселя на той же координате x. Если ширина больше единицы также рисуются дополнительные линии, параллельные проведенной. При значении оси y, которое является делителем значения высоты, деленной на количество частей, рисуется линия до правого пикселя на той же координате y.

for(int i = 0; i < bmih.biHeight; i++){

for(int j = 0; j < bmih.biWidth; j++){

if((j != 0) && (j % (bmih.biWidth/m) == 0)){

drawLine(j, i, j, bmih.biHeight-1, r, g, b);

for(q = 1; q <= h/2; q++){

drawLine(j+q, i, j+q, bmih.biHeight-1, r, g, b);

drawLine(j-q, i, j-q, bmih.biHeight-1, r, g, b);

}

}

if((i != 0) && (i % (bmih.biHeight/n) == 0)){

drawLine(j, i, bmih.biWidth-1, i, r, g, b);

for(qq = 1; qq <= h/2; qq++){

drawLine(j, i+qq, bmih.biWidth-1, i+qq, r, g, b);

drawLine(j, i-qq, bmih.biWidth-1, i-qq, r, g, b);

}

}

}

}

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы было создано приложение с консольным интерфейсом для обработки изображений в формате BMP. Для программы было написано полное описание всех функций, системы защиты от разных ситуаций (например, неправильно введенные данные пользователем), созданы структуры для хранения изображения и функции для его обработки.

**Демонстрация работоспособности программы**

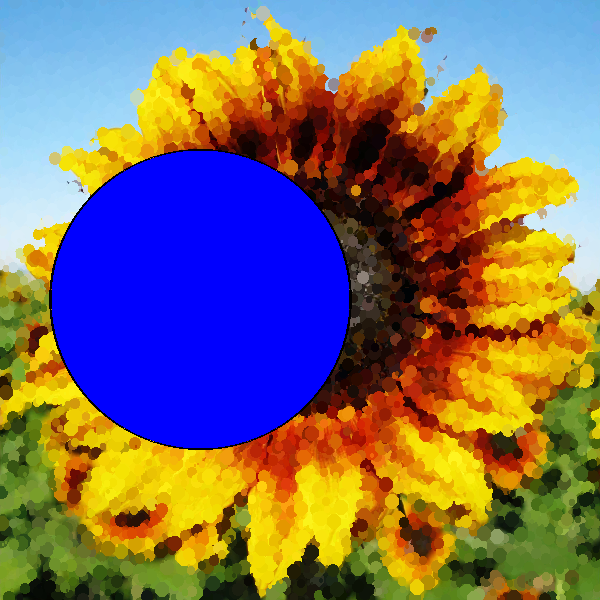
1 задание:



2 задание:



3 задание:



Исходное изображение:

****

**список использованных источников**

1. Керниган Б. В., Ритчи Д. М. Язык программирования Си: Пер. с англ.

— 3-е изд. — СПб.: Невский Диалект, 2001. — 352 с.

# 2. Майк МакГрат Программирование на С для начинающих : Пер. с англ.- СПб.: Эксмо, 2016 – 193 c.

# 3. Лафоре Р.Объектно-ориентированное программирование в С++ : Питер, 4 издание 2004. – 928 с.

4. <http://algolist.manual.ru/graphics/painting/circle.php>

5. Шилдт Герберт. "C" для профессиональных программистов: Пер. с англ.

**ПРИЛОЖЕНИЕ А.**

**ИСХОДЫЙ КОД ПРОГРАММЫ.**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <stdint.h>

#pragma pack(push, 1)

typedef struct{

unsigned int biSize;

unsigned int biWidth;

unsigned int biHeight;

uint16\_t biPlanes;

uint16\_t biBitCount;

unsigned int biCompression;

unsigned int biSizeImage;

int biXPelsPerMeter;

int biYPelsPerMeter;

unsigned int biClrUsed;

unsigned int biClrImportant;

}BITMAPINFOHEADER;

typedef struct{

uint16\_t bfType;

unsigned int bfSize;

uint16\_t bfReserved1;

uint16\_t bfReserved2;

unsigned int bfOffBits;

}BITMAPFILEHEADER;

typedef struct{

unsigned char red;

unsigned char green;

unsigned char blue;

}RGB;

#pragma pack(pop)

RGB \*\*arr, \*pad;

BITMAPFILEHEADER bmfh;

BITMAPINFOHEADER bmih;

FILE \*f, \*ff;

size\_t padding;

unsigned char r = 0, g = 0, b = 0;

char \*inputFile, \*outputFile = "out.bmp";

static const char\* optString = "i?c:y:p:q:b:";

void filter(char\* cv, int zn);

void bmpforpieces(int m, int n, int h);

void drawfirstcircle(int ctrX, int ctrY, int rad, int h, int fill, RGB clr);

void drawCircle(int ctrX, int ctrY, int rad);

RGB tmpColor(unsigned char r, unsigned char g, unsigned char b);

int cmp(RGB a, RGB b);

void drawLine(int x1, int y1, int x2, int y2, unsigned char r, unsigned char g, unsigned char b);

void flood(int xx, int yy, unsigned char r, unsigned char g, unsigned char b);

void outFile();

int isallright();

void imageOut(BITMAPFILEHEADER bfh, BITMAPINFOHEADER bih);

void printAppInfo();

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

inputFile = malloc(strlen(argv[argc-1]) \* sizeof(char));

strcpy(inputFile, argv[argc-1]);

int opt = 0;

opt = getopt(argc, argv, optString);

f = fopen(inputFile, "r+b");

fread(&bmfh, sizeof(bmfh), 1, f);

fread(&bmih, sizeof(bmih), 1, f);

arr = (RGB\*\*) malloc(bmih.biHeight \* sizeof(RGB\*));

for(int i = 0; i < bmih.biHeight; i++)

arr[i] = (RGB\*)malloc(bmih.biWidth \* sizeof(RGB));

pad = (RGB\*)malloc(4 \* sizeof(RGB));

padding = 0;

if((bmih.biWidth \* 3) % 4)

padding = 4 - (bmih.biWidth \* 3) % 4;

for(int i = 0; i < bmih.biHeight; i++){

for (int j = 0; j < bmih.biWidth; j++)

fread(&(arr[i][j]), sizeof(RGB),1, f);

if(padding != 0)

fread(pad, padding, 1, f);

}

fclose(f);

while(opt != -1){

switch(opt){

case 'i':

imageOut(bmfh, bmih);

break;

case '?':

printAppInfo();

break;

case 'c':{

int rrr[3];

for (int k = -1; k < 2; ++k)

rrr[k + 1] = atoi(argv[optind + k]);

r = (unsigned char) rrr[2];

g = (unsigned char) rrr[1];

b = (unsigned char) rrr[0];

break;

}

case 'y':{

char cl[10];

strcpy(cl, argv[optind-1]);

int z = atoi(argv[optind]);

if((strcmp(cl, "r") == 0 || strcmp(cl, "g") == 0 || strcmp(cl, "b") == 0) && (z == 0 || z == 255))

filter(cl, z);

else

puts("Color must be r, g or b and value must be 0 or 255");

break;

}

case 'p':{

int ppp[3];

for (int k = -1; k < 2; ++k)

ppp[k + 1] = atoi(argv[optind + k]);

if((ppp[2] > bmih.biWidth/ppp[1]) || (ppp[2] > bmih.biHeight/ppp[2]))

puts("such thickness is too big for this separation");

else

bmpforpieces(ppp[0], ppp[1], ppp[2]);

break;

}

case 'q':{

int crr[8];

for (int k = -1; k < 7; ++k)

crr[k+1] = atoi(argv[optind + k]);

RGB cc = tmpColor((unsigned char) crr[7], (unsigned char) crr[6], (unsigned char) crr[5]);

drawfirstcircle(crr[0], crr[1], crr[2], crr[3], crr[4], cc);

break;

}

case 'b':{

int crr1[9];

for (int k = -1; k < 8; ++k)

crr1[k+1] = atoi(argv[optind + k]);

RGB cc = tmpColor((unsigned char) crr1[8], (unsigned char) crr1[7], (unsigned char) crr1[6]);

int rad = (crr1[1]-crr1[0])/2;

int ctrX1 = crr1[0]+rad;

int ctrY1 = crr1[1]-rad;

drawfirstcircle(ctrX1, ctrY1, rad, crr1[4], crr1[5], cc);

break;

}

default:

break;

}

opt = getopt(argc, argv, optString);

}

return 0;

}

void imageOut(BITMAPFILEHEADER bfh, BITMAPINFOHEADER bih){

printf("BITMAPFILEHEADER:\n");

printf("The file type: %x (%hu)\n", bfh.bfType, bfh.bfType);

printf("The size, in bytes, of the bitmap file: %i\n", bfh.bfSize);

printf("Reserved: %i %i\n", bfh.bfReserved1, bfh.bfReserved2);

printf("The offset, in bytes, from the beginning of the BITMAPFILEHEADER structure to the bitmap bits: %i\n\n", bfh.bfOffBits);

printf("BITMAPINFOHEADER:\n");

printf("The number of bytes required by the structure: %i\n", bih.biSize);

printf("The width of the bitmap, in pixels: %i\n", bih.biWidth);

printf("The height of the bitmap, in pixels: %i\n", bih.biHeight);

printf("The number of planes for the target device: %i\n", bih.biPlanes);

printf("The number of bits-per-pixel: %i\n", bih.biBitCount);

printf("The type of compression for a compressed bottom-up bitmap: %i\n", bih.biCompression);

printf("The size, in bytes, of the image: %i\n", bih.biSizeImage);

printf("The number of color indexes in the color table that are actually used by the bitmap: %i\n", bih.biClrUsed);

printf("The horizontal resolution, in pixels-per-meter, of the target device for the bitmap: %i\n", bih.biWidth);

printf("The vertical resolution, in pixels-per-meter, of the target device for the bitmap: %i\n", bih.biHeight);

printf("The number of color indexes that are required for displaying the bitmap: %i\n", bih.biClrImportant);

}

void printAppInfo(){

printf("\n-? : Print available commands\n");

printf("-i : Print image file information\n");

printf("-c [b] [g] [r] : Set the color\n\n");

printf("-y [k] [n(0/255)] : Rgb change to 0 or 255\n");

printf("-p [n] [m] [thickness] : Divides the image into parts, n \* m\n");

printf("-q [ctrX] [ctrY] [rad] [thickness] [filling(1/0)] [b] [g] [r]: Draw a circle - first way;\n use -c to change the color of the line of this circle\n");

printf("-b [X1] [Y1] [X2] [Y2] [thickness] [filling(1/0)] [b] [g] [r]: Draw a circle - second way;\n use -c to change the color of the line of this circle\n\n");

}

void filter(char\* cv, int zn){

if(isallright())

return;

if(strcmp(cv, "r") == 0){

for(int i = 0; i < bmih.biHeight; i++)

for(int j = 0; j < bmih.biWidth; j++)

arr[i][j].blue = (unsigned char)zn;

printf("red value was changed to %u\n", (unsigned char)zn);

}

if(strcmp(cv, "g") == 0){

for(int i = 0; i < bmih.biHeight; i++)

for(int j = 0; j < bmih.biWidth; j++)

arr[i][j].green = (unsigned char)zn;

printf("green value was changed to %u\n", (unsigned char)zn);

}

if(strcmp(cv, "b") == 0){

for(int i = 0; i < bmih.biHeight; i++)

for(int j = 0; j < bmih.biWidth; j++)

arr[i][j].red = (unsigned char)zn;

printf("blue value was changed to %u\n", (unsigned char)zn);

}

outFile();

}

void bmpforpieces(int n, int m, int h){

if(isallright())

return;

if(bmih.biHeight % n != 0){

puts("Height can't be divided by integer");

return;

}

else

printf("Num of pieces on y = %d\n", bmih.biHeight/n);

if(bmih.biWidth % m != 0){

puts("Width can't be divided by integer");

return;

}

else

printf("Num of pieces on x = %d\n", bmih.biWidth/m);

int q, qq;

for(int i = 0; i < bmih.biHeight; i++){

for(int j = 0; j < bmih.biWidth; j++){

if((j != 0) && (j % (bmih.biWidth/m) == 0)){

drawLine(j, i, j, bmih.biHeight-1, r, g, b);

for(q = 1; q <= h/2; q++){

drawLine(j+q, i, j+q, bmih.biHeight-1, r, g, b);

drawLine(j-q, i, j-q, bmih.biHeight-1, r, g, b);

}

}

if((i != 0) && (i % (bmih.biHeight/n) == 0)){

drawLine(j, i, bmih.biWidth-1, i, r, g, b);

for(qq = 1; qq <= h/2; qq++){

drawLine(j, i+qq, bmih.biWidth-1, i+qq, r, g, b);

drawLine(j, i-qq, bmih.biWidth-1, i-qq, r, g, b);

}

}

}

}

outFile();

}

void drawfirstcircle(int ctrX, int ctrY, int rad, int h, int fill, RGB clr){

if((clr.red < 0) || (clr.green > 255) || (clr.blue < 0) || (clr.red > 255) || (clr.green < 0) || (clr.blue > 255)){

printf("Error\n");

return;

}

for(int dd = 0; dd < h; dd++)

drawCircle(ctrX, ctrY, rad-dd);

if(fill){

if(ctrX < 0 || ctrX - rad < 0 || ctrY < 0 || ctrY - rad < 0 || ctrX > bmih.biWidth || ctrX + rad > bmih.biWidth || ctrY > bmih.biHeight || ctrY + rad > bmih.biHeight){

printf("Error\n");

return;

}

drawCircle(ctrX, ctrY, rad-h);

for(int i = 0; i < bmih.biHeight; i++)

for(int j = 0; j < bmih.biWidth; j++){

if((j-ctrX)\*(j-ctrX) + (i-ctrY)\*(i-ctrY) < (rad-h+1)\*(rad-h+1))

arr[i][j] = tmpColor(clr.red, clr.green, clr.blue);

}

}

outFile();

}

void drawCircle(int ctrX, int ctrY, int rad){

if(isallright())

return;

if(ctrX < 0 || ctrX - rad < 0 || ctrY < 0 || ctrY - rad < 0 || ctrX > bmih.biWidth || ctrX + rad > bmih.biWidth || ctrY > bmih.biHeight || ctrY + rad > bmih.biHeight){

printf("Error\n");

return;

}

int x = 0;

int y = rad;

int delta = 1 - 2 \* rad;

int error = 0;

int dd;

while(y >= 0){

arr[ctrY + y][ctrX + x] = tmpColor(r, g, b);

arr[ctrY - y][ctrX + x] = tmpColor(r, g, b);

arr[ctrY + y][ctrX - x] = tmpColor(r, g, b);

arr[ctrY - y][ctrX - x] = tmpColor(r, g, b);

error = 2 \* (delta + y) - 1;

if((delta < 0) && (error <= 0)){

delta += 2 \* ++x + 1;

continue;

}

if((delta > 0) && (error > 0)){

delta -= 2 \* --y + 1;

continue;

}

delta += 2 \* (++x - y--);

}

}

void drawLine(int x1, int y1, int x2, int y2, unsigned char r, unsigned char g, unsigned char b){

const int deltaX = abs(x2 - x1);

const int deltaY = abs(y2 - y1);

const int signX = x1 < x2 ? 1 : -1;

const int signY = y1 < y2 ? 1 : -1;

int error = deltaX - deltaY;

arr[y2][x2] = tmpColor(r, g, b);

while(x1 != x2 || y1 != y2){

arr[y1][x1] = tmpColor(r, g, b);

const int error2 = error \* 2;

if(error2 > -deltaY){

error -= deltaY;

x1 += signX;

}

if(error2 < deltaX){

error += deltaX;

y1 += signY;

}

}

}

int isallright(){

if(bmih.biCompression != 0){

printf("Unknown compression method\n");

return 1;

}

if(bmih.biClrUsed != 0){

printf("Version not supported\n");

return 1;

}

return 0;

}

RGB tmpColor(unsigned char r, unsigned char g, unsigned char b){

RGB tmp;

tmp.blue = b;

tmp.green = g;

tmp.red = r;

return tmp;

}

int cmp(RGB a, RGB b){

if((a.blue == b.blue) && (a.green == b.green) && (a.red == b.red))

return 1;

else

return 0;

}

void outFile(){

ff = fopen(outputFile, "w+b");

fwrite(&bmfh, sizeof(bmfh), 1, ff);

fwrite(&bmih, sizeof(bmih), 1, ff);

for(int i = 0; i < bmih.biHeight; i++){

for(int j = 0; j < bmih.biWidth; j++)

fwrite(&(arr[i][j]), sizeof(RGB), 1, ff);

if(padding != 0)

fwrite(pad, padding, 1, ff);

}

fclose(ff);

}