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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка BMP файлов

Студент гр. 0382 _____ Афанасьев Н. С.

Преподаватели Чайка К. В.
_____ Шевская Н.В.

Санкт-Петербург

2021

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент: Афанасьев Н. С.

Группа 0382

Тема работы: Обработка BMP файлов

Вариант 10

Исходные данные:

Программа должна иметь CLI или GUI.

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) Заменяет все пиксели одного заданного цвета на другой цвет. Функционал определяется:

- Цвет, который требуется заменить
- Цвет на который требуется заменить

2) Сделать рамку в виде узора. Рамка определяется:
   - Узором (должно быть несколько на выбор. Красивый узор можно получить используя фракталы)
   - Цветом
   - Шириной
3) Поиск всех залитых прямоугольников заданного цвета. Требуется найти все прямоугольники заданного цвета и обвести их линией. Функционал определяется:
   - Цветом искомых прямоугольников
   - Цветом линии для обводки
   - Толщиной линии для обводки

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 05.04.2021

Дата сдачи реферата: 16.05.2021

Дата защиты реферата: 18.05.2021

Студент гр. 0382 _____ Афанасьев Н. С.

Преподаватели _____ Чайка К. В.
Шевская Н.В.

АННОТАЦИЯ

В процессе выполнения курсовой работы создавалась программа на языке C для обработки BMP файла. Программа имеет CLI (Command Line Interface) с возможностью вывода справки о программе и о всех возможных командах и ключах. Программа поддерживает только BMP Version 3 (Microsoft Windows 3.x) с BITMAPINFOHEADER в 40 байт, глубиной 24 бита, без сжатия. Разработка велась на операционной системе Windows 10 Home x64 в редакторе исходного кода Visual Studio Code с использованием компилятора MinGW.

Инструкция по запуску приложения: скомпилировать файл bmpedit.c в файл с названием bmpedit

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – создать приложение на языке C с CLI для обработки изображения формата BMP согласно запросам пользователя.

Для выполнения работы необходимо выполнить следующие задачи:

- Обработка запросов пользователя
- Создание структур для работы с файлами
- Чтение и запись BMP файла
- Изменение исходного изображения
- Обработка возможных ошибок

Для первой задачи, для реализации интерфейса, используется библиотека *getopt.h*

Для третьей задачи, для чтения и записи файлов, используются методы библиотеки stdio.h.

Для четвёртой задачи, используется работа с двумерным массивом пикселей изображения.

2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1. Структуры

Для начала все структуры оборачиваем в *#pragma pack(push, 1)* и *#pragma pack(pop)*. Первое устанавливает размер выравнивания в 1 байт, второе возвращает предыдущую настройку. Без этого размер структур в памяти будет варьироваться в зависимости от компилятора.

Далее создаются структуры **BitmapFileHeader** и **BitmapInfoHeader** с полями, соответствующими выбранной версии формата BMP. Структура **BGR** определяет один пиксель – его координатами являются значения голубого, зелёного и красного оттенков (от 0 до 255). Структура **BMP** содержит информацию о файле: *BitmapFileHeader* и *BitmapInfoHeader*, массив пикселей *pixelArray* и переменную *bytesPerRow*, хранящую количество байт в одном ряду изображения.

С помощью ключевого слова *typedef* этим структурам назначаются их имена.

2.2. Интерфейс командной строки и выбор операции

Формат ввода запросов представляет из себя строку:
./bmpedit [команда] [имя исходного файла] -[ключ1]/--[ключ1] [аргумент1] ...

Для обработки запросов в функции *int* **main***(int argc, char* argv[])* сначала создаются переменные для аргументов со значениями по умолчанию (*image, output, borderWidth, style, color1, color2*), далее создаются объекты структуры *option* из библиотеки *getopt.h*, определяющие короткие и соответствующие им длинные ключи.

Если аргументы, помимо имени программы, не переданы, или второй аргумент (далее, под вторым аргументом будет пониматься выбранная команда) равен “*help*”, то функцией *void* **printHelp()** выводится справка. Если аргументы переданы, но их меньше трёх, то согласно формату запроса не передано имя файла, вследствие чего выводится ошибка. Если файл введён, то

производится попытка его чтения, и если в процессе возникла ошибка, то она выводится на экран (процесс чтения описан ниже).

Когда файл считан, определяется введённая команда с помощью ступенчатых условий и *strcmp*. Если команда не найдена, выводится ошибка. Если команда – "*info*", то с помощью функции *void* **printFileInfo***(BMP image)* выводится информация о считанном файле. Для каждой остальной команды определяется массив возможных ключей, и затем начинается считывание ключей с помощью функции *getopt_long* библиотеки *getopt.h*. Проверка ключа определяется с помощью функции *int* **findKey***(...)*: в зависимости от ключа присваивается значение соответствующей переменной. Если ключ для заданной команды не опознан, то выводится ошибка. Отдельно стоит сказать, что для определения введённого цвета используется функция *int* **text2BGR***(char* str, BGR* color)*, которая в зависимости от введённого формата: HEX (.ffffff) или RGB (255.255.255) – создаёт объект BGR с помощью функции *BGR* **createBGR***(uint8_t blue, uint8_t green, uint8_t red)*. Также, если у выходного файла отсутствует расширение "*.bmp*", то оно добавляется. Далее в зависимости от выбранной операции запускается соответствующая функция.

2.3. Чтение файла

Считывание файла происходит с помощью функции *int* **readBMP***(char* path, BMP* image)*. Если файл не удалось открыть с помощью функции *fopen*, выводится ошибка. Далее с помощью *fread* считывается часть файла и записывается в объект *BitmapFileHeader*. Здесь проверяется сигнатура файла: если это не "*0x4d42*", то файл не формата BMP, и выводится ошибка. Далее, по такому же принципу записывается объект *BitmapInfoHeader*. Здесь проверяются версия BMP (у поддерживаемой размер *InfoHeader* – 40 байт), глубина изображения (поддерживается 24 бита на пиксель) и отсутствие сжатия (так как не поддерживается). Далее рассчитывается количество байт на один ряд (должно делиться на 4) и считывается массив пикселей с учётом их расположения в BMP файле. Затем файл закрывается.

2.4. Первая операция

Первая операция представляет из себя поиск пикселей одного цвета и замена их на другой цвет. Это осуществляется с помощью функции *void* **colorSwap***(BMP* image, BGR colorToChange, BGR newColor)*. Функция проходит все пиксели и заменяет пиксели одного цвета на другой.

2.5. Вторая операция

Вторая операция заключается в отрисовке рамки в виде узора с заданной шириной и цветом. Это осуществляется с помощью функции *void* **frame***(BMP* image, uint16_t padding, BGR color, uint8_t pattern, BGR patternColor)*. Сначала пересчитываются значения высоты и ширины изображения и количество байт в ряду. Далее создаётся новый массив пикселей (изначально с нулями), которые закрашиваются в зависимости от положения: если они располагаются на узоре, то закрашиваются цветом узора; иначе, если они находятся на территории рамки, то закрашиваются цветом рамки; иначе берутся и присваиваются (с учётом ширины рамки) значения пикселей из исходной картинки. Всего определены 3 узора: если в качестве узора указано 0 (по умолчанию), то будет отрисовываться рамка без узора; узор 1 представляет из себя чередование полос двух цветов (цвета рамки и узора), узор 2 представляет из себя волны, а узор 3 представляет из себя кривые линии. Последние два узора выполнены с помощью функции синуса. Затем в изначальном массиве пикселей освобождается память и присваивается уже новый массив.

2.6. Третья операция

Третья операция заключается в поиске прямоугольников одного цвета и их обводка. Это осуществляется с помощью функции *void* **findRectangles***(BMP* image, BGR color, BGR borderColor, uint16_t borderWidth)*. Сначала функция проходит по всем пикселям и ищет нижний левый угол предполагаемого прямоугольника, то есть сверху и справа должны быть пиксели того же цвета, а слева и снизу – нет. Далее функция идет вправо пока пиксели одного цвета,

попутно проверяя отсутствие пикселей того же цвета снизу - так получается нижнее основание (ширина) предполагаемого прямоугольника. Далее для каждой точки получившегося основания, считается высота: функция проходит вверх, пока пиксели одного цвета. Для первого и последнего столбца проверяется отсутствие пикселей того же цвета снаружи. Если высота для одной точки не совпадает с высотой для предыдущей точки, то исследуемая фигура – не прямоугольник. Так, функция получает координаты левого нижнего и правого верхнего угла прямоугольника, которые используются для отрисовки рамки с помощью функции *void* **drawRectangleFrame***(BMP* image, int16_t x1, int16_t y1, uint16_t x2, uint16_t y2, uint8_t borderWidth, BGR color)*, которая просто закрашивает пиксели, удовлетворяющих положению рамки заданной ширины.

2.7. Запись в файл

Файл для вывода является по умолчанию исходным файлом, если пользователь не указал иначе через специальный ключ. Если файл указан без расширения “.bmp”, то он добавляется автоматически.

Запись в файл происходит с помощью функции *void* **writeBMP***(char* path, BMP* image)*. Сначала открывается файл для вывода изображения. Далее с помощью функции *fwrite* записываются сначала объекты *BitmapFileHeader* и *BitmapInfoHeader*, а затем и каждая строка в массиве пикселей. Затем файл закрывается.

Разработанный программный код см. в приложении В.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была создана программа на языке C с CLI для обработки изображений формата BMP. Пользователь вводит необходимую команду, имя файла, и аргументы через ключи, чтобы выполнить одну из следующих операций: замена всех пикселей одного цвета на другой, создание рамки с узором определённых цветов и размера, поиск прямоугольников определённого цвета и обводка их рамкой заданного цвета и ширины, вывод информации о файле, вывод справочной информации. Программа обрабатывает возможные ошибки вместо того, чтобы просто прекращать работу. Можно сделать вывод о соответствии полученного результата поставленной цели.

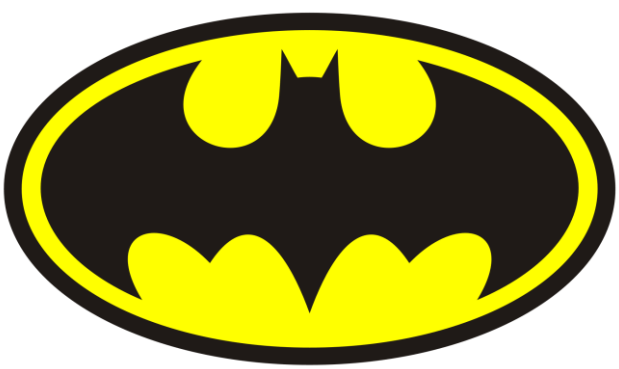
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

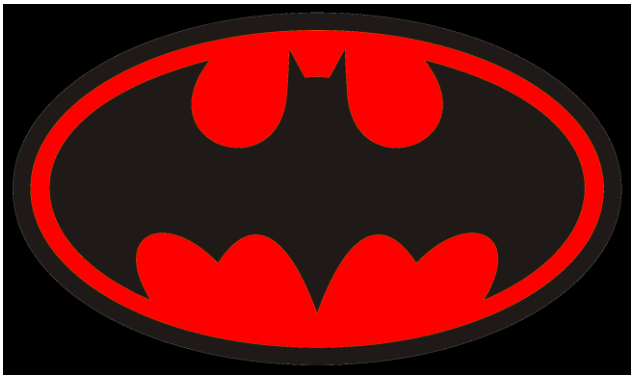
1. Сайт (онлайн-справочник) www.c-cpp.ru
2. Сайт en.wikipedia.org/wiki/BMP_file_format – общая информация
3. Сайт www.fileformat.info/format/bmp/egff.htm - версии BMP формата
4. Сайт https://firststeps.ru/linux/r.php?10 – информация о getopt.h

ПРИЛОЖЕНИЕ А
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

```
D:\a>bmpedit swap image.bmp -c .FFFF00 -C 255.0.0 -o test1

D:\a>bmpedit swap test1.bmp --color1 255.255.255 --color2 0.0.0
```





Пример 1 – Первая операция

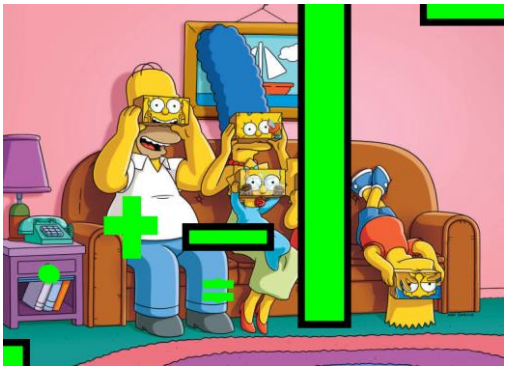
```
D:\a>bmpedit frame image2.bmp --color1 .9400d3 --color2 .ffff00 --width 41 --style 3 -o ./test2

D:\a>bmpedit frame test2.bmp -c .4b0082 -C .00FF00 -w 19 -s 2

D:\a>bmpedit frame test2.bmp -c .ffffff -w 11 -s 1
```

Пример 2 – Вторая операция

```
D:\a>bmpedit findrect rectangles.bmp -c 0.255.0 -C 0.0.0 --width 10 -o test3
```



Пример 3 – Третья операция

```
D:\a>bmpedit

        bmpedit - программа с CLI для редактирования bmp файлов
Поддерживаются только файлы BMP Version 3 (Microsoft Windows 3.x / 40 bytes BITMAPINFOHEADER)
с глубиной изображения - 24 бита и без сжатия.

Формат ввода: ./bmpedit [команда] [имя исходного файла] -[ключ1]/--[ключ1] [аргумент1] ...

        Комманды:
        findrect [имя файла] - поиск и обводка всех залитых прямоугольников заданного цвета.
                -c/--color1 [.fffffff/255.255.255] - искомый цвет (HEX или RGB)
                -w/--width  [число] - ширина обводки (по умолчанию 0)
                -C/--color2 [.fffffff/255.255.255] - цвет обводки (HEX или RGB) (по умолчанию чёрный)
                -o/--output [путь] - файл для вывода (по умолчанию исходный файл)
        frame [имя файла] - создание рамки в виде узора.
                -c/--color1 [.fffffff/255.255.255] - цвет рамки (HEX или RGB)
                -w/--width  [число] - ширина рамки (по умолчанию 0)
                -s/--style  [0-3] - узор (0 - (по умолчанию) отсутствует, 1-полосы, 2-волны, 3-кривые линии)
                -C/--color2 [.fffffff/255.255.255] - цвет узора (HEX или RGB) (по умолчанию чёрный)
                -o/--output [путь] - файл для вывода (по умолчанию исходный файл)
        help - вывод справки о работе программы.
        info [имя файла] - вывод информации об изображении.
        swap [имя файла] - замена всех пикселей одного заданного цвета на другой цвет.
                -c/--color1 [.fffffff/255.255.255] - искомый цвет (HEX или RGB)
                -C/--color2 [.fffffff/255.255.255] - цвет для замены (HEX или RGB) (по умолчанию чёрный)
                -o/--output [путь] - файл для вывода (по умолчанию исходный файл)
```

Пример 4 – Вывод справки

```
D:\a>bmpedit info simpsonsvr.bmp
signature:      4d42 (19778)
fileSize:       141a62 (1317474)
reserved1:      0 (0)
reserved2:      0 (0)
pixelArrOffset: 36 (54)
headerSize:     28 (40)
width:          30c (780)
height:         233 (563)
planes:         1 (1)
bitsPerPixel:   18 (24)
compression:    0 (0)
imageSize:      0 (0)
xPixelsPerMeter:        2e23 (11811)
yPixelsPerMeter:        2e23 (11811)
colorsInColorTable:     0 (0)
importantColorCount:    0 (0)
```

Пример 5 – Вывод информации о файле

```
D:\a>bmpedit drawSomething simpsonsvr.bmp
Команды drawSomething не существует

D:\a>bmpedit info
Не введено имя файла с изображением

D:\a>bmpedit swap 1px.bmp -c .000000 -C .000000 -a aaaaaaaa
Найден неопознанный ключ для комманды "swap"

D:\a>bmpedit swap 1px.bmp -c .jjjjjj -C 1000.-1000.1000 -a aaaaaaaa
Введён некорректный цвет

D:\a>bmpedit findrect rectangles.bmp -c 0.255.0 -C 0.0.0 --width -10 -o test3
Значение размера рамки не может быть меньше 0

D:\a>bmpedit info bmpv5.bmp
Неподдерживаемая версия BMP. Должен быть BMP Version 3 (Microsoft Windows 3.x / 40 bytes BITMAPINFOHEADER)

D:\a>bmpedit info a.txt
Неподдерживаемый тип файла
```

Пример 6 – Возможные ошибки

ПРИЛОЖЕНИЕ В
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл bmpedit.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <math.h>
#include <locale.h>
#include <getopt.h>

#pragma pack (push, 1)
    typedef struct{
        uint16_t signature; // BM BA CI CP IC PT
        uint32_t fileSize;
        uint16_t reserved1;
        uint16_t reserved2;
        uint32_t pixelArrOffset;
    } BitmapFileHeader;

    typedef struct{
        uint32_t headerSize; // 28(40) for BITMAPINFOHEADER
        uint32_t width;
        uint32_t height;
        uint16_t planes; // must be 1
        uint16_t bitsPerPixel; // 18(24) bits (BGR)
        uint32_t compression; // 0 (no compression)
        uint32_t imageSize;
        uint32_t xPixelsPerMeter;
        uint32_t yPixelsPerMeter;
        uint32_t colorsUsed;
        uint32_t importantColors;
    } BitmapInfoHeader;

    typedef struct{
        uint8_t blue;
```

```
        uint8_t green;
        uint8_t red;
    } BGR;

    typedef struct{
        BitmapFileHeader fileHeader;
        BitmapInfoHeader infoHeader;
        BGR** pixelArray;
        uint32_t bytesPerRow;
    } BMP;

#pragma pack(pop)

int readBMP(char* path, BMP* image){
    FILE* inputStream = fopen(path, "rb");
    if(!inputStream){ wprintf(L"Не удаётся открыть файл %hs\n", path); return
0; }

    fread(&(image->fileHeader), 1, sizeof(BitmapFileHeader), inputStream);
    if(image->fileHeader.signature != 0x4d42){ wprintf(L"Неподдерживаемый тип
файла\n"); return 0; }

    fread(&(image->infoHeader), 1, sizeof(BitmapInfoHeader), inputStream);
    if(image->infoHeader.headerSize != 40){ wprintf(L"Неподдерживаемая версия
BMP. Должен быть BMP Version 3 (Microsoft Windows 3.x / 40 bytes
BITMAPINFOHEADER)\n"); return 0; }
    if(image->infoHeader.bitsPerPixel != 24) { wprintf(L"Неподдерживаемая
глубина изображения. Должная быть 24 бита\n"); return 0; }
    if(image->infoHeader.compression != 0) { wprintf(L"Сжатие изображения не
поддерживается\n"); return 0; }

    uint32_t height = image->infoHeader.height;
    uint32_t width = image->infoHeader.width;
    if(height > 65535 || width > 65535) { wprintf(L"Слишком большое
изображение\n"); return 0; }
    uint32_t bytesPerRow = width*sizeof(BGR) + (4 - (width*sizeof(BGR))%4)%4;

    image->bytesPerRow = bytesPerRow;
```

```
    image->pixelArray = malloc(height*sizeof(BGR*));
    for(int i = 0; i < height; i++){
        image->pixelArray[i] = malloc(bytesPerRow);
        fread(image->pixelArray[i], 1, bytesPerRow, inputStream);
    }
    fclose(inputStream);
    return 1;
}

BGR createBGR(uint8_t blue, uint8_t green, uint8_t red){
    BGR bgr = {blue, green, red};
    return bgr;
}

void drawRectangleFrame(BMP* image, int16_t x1, int16_t y1, uint16_t x2,
uint16_t y2, uint8_t borderWidth, BGR color){
    for(int x = x1 - borderWidth; x <= x2 + borderWidth; x++)
        for(int y = y1 - borderWidth; y <= y2 + borderWidth; y++)
            if(x >= 0 && y >= 0 && x < image->infoHeader.width && y < image-
>infoHeader.height &&
                ((x <= x1 || x >= x2) || (y <= y1 || y >= y2))) image-
>pixelArray[y][x] = color;
}

void colorSwap(BMP* image, BGR colorToChange, BGR newColor){
    for(int i = 0; i < image->infoHeader.height; i++)
        for(int j = 0; j < image->infoHeader.width; j++)
            if(!memcmp(&(image->pixelArray[i][j]), &colorToChange,
sizeof(BGR))) image->pixelArray[i][j] = newColor;
}

void frame(BMP* image, uint16_t padding, BGR color, uint8_t pattern, BGR
patternColor){
    uint32_t newHeight = image->infoHeader.height + 2*padding;
    uint32_t newWidth = image->infoHeader.width + 2*padding;
    uint32_t newBytesPerRow = newWidth*sizeof(BGR) + (4-
(newWidth*sizeof(BGR))%4)%4;
```

```
    BGR** newPixArr = malloc(newHeight*sizeof(BGR*));

    short a;
    for(int i = 0; i < newHeight; i++){
        newPixArr[i] = calloc(1, newBytesPerRow);
        for(int j = 0; j < newWidth; j++){
            if(i < padding || i >= newHeight-padding || j < padding || j >=
newWidth-padding){
                switch (pattern){
                    case 1: //horizaontal lines
                        if(((i > padding && i < newHeight-padding) && (i %
padding*2 < padding )) ||
                            ((j > padding && j < newWidth-padding) && (j %
padding*2 < padding ))) newPixArr[i][j] = patternColor;
                        else newPixArr[i][j] = color;
                        break;
                    case 2: //waves
                        if((j < padding || j > newWidth - padding) && (i <
padding || i > newHeight - padding)){ newPixArr[i][j] = patternColor; break; }
                        a = round((sin((double)j/10) * padding / 2)+padding/2);
                        if((i < a) || (i > newHeight - a)){ newPixArr[i][j] =
patternColor; break; }
                        a = round((sin((double)i/10) * padding / 2)+padding/2);
                        if((j < a) || (j > newWidth - a)){ newPixArr[i][j] =
patternColor; break; }
                        newPixArr[i][j] = color;
                        break;
                    case 3: //curved lines
                        a = round((sin((double)j/10) * padding / 2)+padding/2);
                        if((j > padding && j < newWidth-padding) && ((abs(i -
a) < 3) || (abs(newHeight - i - a) < 3))) newPixArr[i][j] = patternColor;
                        else{
                            a = round((sin((double)i/10) * padding /
2)+padding/2);
                            if((i > padding && i < newHeight-padding) &&
((abs(j - a) < 3) || (abs(newWidth - j - a) < 3))) newPixArr[i][j] =
patternColor;
                            else newPixArr[i][j] = color;
```

```
                        }
                        break;
                    default:
                        newPixArr[i][j] = color;
                        break;
                }
            }
            else{
                newPixArr[i][j] = image->pixelArray[i-padding][j-padding];
            }
        }
    }
    for(int i = 0; i < image->infoHeader.height; i++) free(image-
>pixelArray[i]);
    free(image->pixelArray);

    image->infoHeader.height = newHeight;
    image->infoHeader.width = newWidth;
    image->bytesPerRow = newBytesPerRow;
    image->pixelArray = newPixArr;
}

void findRectangles(BMP* image, BGR color, BGR borderColor, uint16_t
borderWidth){
    BGR** arr = image->pixelArray;
    int count = 0, maxCount = 10;
    uint16_t* coordinates = malloc(maxCount*4*sizeof(uint16_t));
    uint16_t x2, y2; uint8_t sameHeight;
    for(int x = 0; x < image->infoHeader.width-1; x++){
        for(int y = 0; y < image->infoHeader.height-1; y++){
            if(!memcmp(&arr[y][x], &color, sizeof(BGR)) &&
                !memcmp(&arr[y+1][x], &color, sizeof(BGR)) &&
!memcmp(&arr[y][x+1], &color, sizeof(BGR)) &&
                (x== 0 || memcmp(&arr[y][x-1], &color, sizeof(BGR))) && (y == 0
|| memcmp(&arr[y-1][x], &color, sizeof(BGR)))){
                    int width;
                    for(width = x+1; (width != image->infoHeader.width) &&
!memcmp(&arr[y][width], &color, sizeof(BGR)); width++){
```

```
                        if(y > 0 && !memcmp(&arr[y-1][width], &color,
sizeof(BGR))){ sameHeight=0; break; }
                    }
                    x2=width-1;
                    sameHeight = 1; y2 = 0;
                    for(int i = x; i <= x2; i++){
                        int height;
                        for(height = y+1; height != image->infoHeader.height &&
!memcmp(&arr[height][i], &color, sizeof(BGR)); height++){
                            if((i==x && x > 0 && !memcmp(&arr[height][x-1],
&color, sizeof(BGR)))||
                                (i==x2 && x2 + 1 < image->infoHeader.width &&
!memcmp(&arr[height][x2+1], &color, sizeof(BGR)))){ sameHeight=0; break; }
                        }
                        if(!sameHeight) break;
                        if(!y2) y2 = height-1;
                        else if(height-1 != y2){ sameHeight = 0; break; }
                    }
                    if(sameHeight){
                        coordinates[count*4] = x;
                        coordinates[count*4+1] = y;
                        coordinates[count*4+2] = x2;
                        coordinates[count*4+3] = y2;
                        if(count++==maxCount){ maxCount+=10;
realloc(&coordinates, maxCount*4*sizeof(uint16_t)); }
                    }
                }
            }
        }
        for(int i = 0; i < count; i++){
            drawRectangleFrame(image, coordinates[4*i]-1, coordinates[4*i+1]-1,
coordinates[4*i+2]+1, coordinates[4*i+3]+1, borderWidth, borderColor);
        }
        free(coordinates);
}

void writeBMP(char* path, BMP* image){
    FILE* outputStream = fopen(path, "wb");
```

```
    fwrite(&(image->fileHeader), 1, sizeof(BitmapFileHeader), outputStream);
    fwrite(&(image->infoHeader), 1, sizeof(BitmapInfoHeader), outputStream);
    for(int i=0; i<image->infoHeader.height; i++) fwrite(image->pixelArray[i],
1, image->bytesPerRow, outputStream);

    fclose(outputStream);
}

void printFileInfo(BMP image){
    BitmapFileHeader fileHeader = image.fileHeader;
 printf("signature:\t%x (%hu)\n", fileHeader.signature, fileHeader.signature);
 printf("fileSize:\t%x (%u)\n", fileHeader.fileSize, fileHeader.fileSize);
 printf("reserved1:\t%x (%hu)\n", fileHeader.reserved1, fileHeader.reserved1);
 printf("reserved2:\t%x (%hu)\n", fileHeader.reserved2, fileHeader.reserved2);
 printf("pixelArrOffset:\t%x (%u)\n", fileHeader.pixelArrOffset,
fileHeader.pixelArrOffset);
    BitmapInfoHeader infoHeader = image.infoHeader;
    printf("headerSize:\t%x (%u)\n", infoHeader.headerSize,
infoHeader.headerSize);
 printf("width:     \t%x (%u)\n", infoHeader.width, infoHeader.width);
 printf("height:    \t%x (%u)\n", infoHeader.height, infoHeader.height);
 printf("planes:    \t%x (%hu)\n", infoHeader.planes, infoHeader.planes);
 printf("bitsPerPixel:\t%x (%hu)\n", infoHeader.bitsPerPixel,
infoHeader.bitsPerPixel);
 printf("compression:\t%x (%u)\n", infoHeader.compression,
infoHeader.compression);
 printf("imageSize:\t%x (%u)\n", infoHeader.imageSize, infoHeader.imageSize);
 printf("xPixelsPerMeter:\t%x (%u)\n", infoHeader.xPixelsPerMeter,
infoHeader.xPixelsPerMeter);
 printf("yPixelsPerMeter:\t%x (%u)\n", infoHeader.yPixelsPerMeter,
infoHeader.yPixelsPerMeter);
 printf("colorsInColorTable:\t%x (%u)\n", infoHeader.colorsUsed,
infoHeader.colorsUsed);
 printf("importantColorCount:\t%x (%u)\n", infoHeader.importantColors,
infoHeader.importantColors);
}
```

```
void printHelp(){
    wprintf(L"\n\tbmpedit - программа с CLI для редактирования bmp файлов\n");
    wprintf(L"Поддерживаются только файлы BMP Version 3 (Microsoft Windows 3.x / 40 bytes BITMAPINFOHEADER)\n");
    wprintf(L"с глубиной изображения - 24 бита и без сжатия.\n\n");
 wprintf(L"Формат ввода: ./bmpedit [команда] [имя исходного файла] -[ключ1]/--[ключ1] [аргумент1] ...\n\n");
    wprintf(L"\tКомманды:\n");
    wprintf(L"\tfindrect [имя файла] - поиск и обводка всех залитых прямоугольников заданного цвета.\n");
        wprintf(L"\t\t -c/--color1 [.fffffff/255.255.255] - искомый цвет (HEX или RGB)\n");
        wprintf(L"\t\t -w/--width  [число] - ширина обводки (по умолчанию 0)\n");
        wprintf(L"\t\t -C/--color2 [.fffffff/255.255.255] - цвет обводки (HEX или RGB) (по умолчанию чёрный)\n");
        wprintf(L"\t\t -o/--output [путь] - файл для вывода (по умолчанию исходный файл)\n");
    wprintf(L"\tframe [имя файла] - создание рамки в виде узора.\n");
        wprintf(L"\t\t -c/--color1 [.fffffff/255.255.255] - цвет рамки (HEX или RGB)\n");
        wprintf(L"\t\t -w/--width  [число] - ширина рамки (по умолчанию 0)\n");
        wprintf(L"\t\t -s/--style  [0-3] - узор (0 - (по умолчанию) отсутствует, 1-полосы, 2-волны, 3-кривые линии)\n");
        wprintf(L"\t\t -C/--color2 [.fffffff/255.255.255] - цвет узора (HEX или RGB) (по умолчанию чёрный)\n");
        wprintf(L"\t\t -o/--output [путь] - файл для вывода (по умолчанию исходный файл)\n");
    wprintf(L"\thelp - вывод справки о работе программы.\n");
    wprintf(L"\tinfo [имя файла] - вывод информации об изображении.\n");
    wprintf(L"\tswap [имя файла] - замена всех пикселей одного заданного цвета на другой цвет.\n");
        wprintf(L"\t\t -c/--color1 [.fffffff/255.255.255] - искомый цвет (HEX или RGB)\n");
        wprintf(L"\t\t -C/--color2 [.fffffff/255.255.255] - цвет для замены (HEX или RGB) (по умолчанию чёрный)\n");
        wprintf(L"\t\t -o/--output [путь] - файл для вывода (по умолчанию исходный файл)\n");
```

```
}

int text2BGR(char* str, BGR* color){
    int16_t red = -1, green = -1, blue = -1;
    if(str[0] == '.' && strlen(str) == 7 && strspn(&str[1],
"0123456789aAbBcCdDeEfF") == 6){
        sscanf(str, ".%2x%2x%2x", &red, &green, &blue);
        sscanf(str, ".%2x%2x", &red, &green);
        sscanf(str, ".%2x", &red);
    }else if(strlen(str) >= 5 && strlen(str) <= 11){
        sscanf(str, "%d.%d.%d", &red, &green, &blue);
        sscanf(str, "%d.%d", &red, &green);
        sscanf(str, "%d", &red);
    }
    if(red < 0 || red > 255 || green < 0 || green > 255 || blue < 0 || blue >
255){
        wprintf(L"Введён некорректный цвет\n"); return 0;
    }else *color = createBGR(blue, green, red);
    return 1;
}

int findKey(char* command, int key, BMP* image, char** output, uint16_t*
borderWidth, uint8_t* style, BGR* color1, BGR* color2){
    switch (key){
        case 'c':
            if(!text2BGR(optarg, color1)) return 0;
            break;
        case 'C':
            if(!text2BGR(optarg, color2)) return 0;
            break;
        case 'w':
            if(atoi(optarg) > 65535){ wprintf(L"Задан слишком большой размер
рамки\n"); return 0; }
            if(atoi(optarg) < 0){ wprintf(L"Значение размера рамки не может
быть меньше 0\n"); return 0; }
            else *borderWidth = atoi(optarg);
            break;
        case 's':
```

```
            if(atoi(optarg) > 4 || atoi(optarg) < 0){ wprintf(L"Выбранного
стиля не существует\n"); return 0; }
            else *style = atoi(optarg);
            break;
        case 'o':
            if(strlen(optarg) < 4 || memcmp(".bmp", optarg + strlen(optarg) -
4, 4)) strcat(optarg, ".bmp");
            *output = optarg;
            break;
        default:
            wprintf(L"Найден неопознанный ключ для комманды \"%hs\"\n",
command); return 0;
            break;
    }
    return 1;
}

int main(int argc, char* argv[]){
    setlocale(LC_ALL, "");

    BMP image;
    char* output;
    uint16_t borderWidth = 0;
    uint8_t style = 0;
    BGR color1 = createBGR(0, 0, 0);
    BGR color2 = createBGR(0, 0, 0);

    const struct option borderStruct = {"width", required_argument, NULL, 'w'};
    const struct option styleStruct  = {"style", required_argument, NULL, 's'};
    const struct option color1Struct = {"color1", required_argument, NULL,
'c'};
    const struct option color2Struct = {"color2", required_argument, NULL,
'C'};
    const struct option outputStruct = {"output", required_argument, NULL,
'o'};

    if(argc == 1 || !strcmp(argv[1], "help")) { printHelp(); return 0; }
```

```
    else if(argc < 3) { wprintf(L"Не введено имя файла с изображением\n");
return -1; }
    else if(!readBMP(argv[2], &image)) return -1;
    else{
        output = argv[2];
        int key, index; opterr = 0;
        if(!strcmp(argv[1], "info")) printFileInfo(image);
        else if(!strcmp(argv[1], "swap")){
            struct option opts[] = {color1Struct, color2Struct, outputStruct};
            while((key = getopt_long(argc, argv, "c:C:o:", opts, &index)) != -
1)
                if(!findKey("swap", key, &image, &output, &borderWidth, &style,
&color1, &color2)) return -1;
            colorSwap(&image, color1, color2);
        }
        else if(!strcmp(argv[1], "frame")){
            struct option opts[] = {borderStruct, color1Struct, styleStruct,
color2Struct, outputStruct};
            while((key = getopt_long(argc, argv, "c:C:w:s:o:", opts, &index))
!= -1)
                if(!findKey("frame", key, &image, &output, &borderWidth,
&style, &color1, &color2)) return -1;
            frame(&image, borderWidth, color1, style, color2);
        }
        else if(!strcmp(argv[1], "findrect")){
            struct option opts[] = {borderStruct, color1Struct, color2Struct,
outputStruct};
            while((key = getopt_long(argc, argv, "c:C:w:o:", opts, &index)) !=
-1)
                if(!findKey("findrect", key, &image, &output, &borderWidth,
&style, &color1, &color2)) return -1;
            findRectangles(&image, color1, color2, borderWidth);
        }
        else{ wprintf(L"Команды %hs не существует\n", argv[1]); return -1; }
    }

    writeBMP(output, &image);
    for(int i = 0; i < image.infoHeader.height; i++) free(image.pixelArray[i]);
```

```
    free(image.pixelArray);
    return 0;
}
```