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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка BMP-файла на языке Си.

Студентка гр. 0382 _____ Михайлова О.Д.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студентка Михайлова О.Д.

Группа 0382

Тема работы: обработка BMP-файла на языке Си

Исходные данные:
Программе на вход подается изображение формата BMP, которое необходимо считать, изменить в соответствии с указанными условиями и записать на диск. Функционал реализуется с использованием CLI.

Содержание пояснительной записки: «Аннотация», «Содержание», «Задание», «Ход работы», «Тестирование», «Заключение», «Список использованных источник», «Исходный код программы».

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 05.04.2021

Дата сдачи реферата: 28.05.2021

Дата защиты реферата: 31.05.2021

Студентка _____ Михайлова О.Д.

Преподаватель _____ Берленко Т.А.

АННОТАЦИЯ

Разработана программа на языке Си, которая обрабатывает указанный пользователем BMP-файл со следующими свойствами:

- 24 бита на цвет;
- без сжатия;
- не содержит таблицу цветов.

Функции для редактирования изображения определяются ключами, которые вводит пользователь. Для этого реализован интерфейс CLI, для использования которого подключена библиотека <getopt.h>. Программа проверяет входные данные и в случае неправильного ввода выводит сообщения о соответствующих ошибках. Результат обработки изображения записывается в другой BMP-файл, также указанный пользователем.

Исходный код программы приведен в приложении А.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: написать программу для работы с изображением формата BMP, выбранным пользователем, с использование CLI.

Результатом является программа, выполняющая считывание и обработку изображения формата BMP, выбранного пользователем, и запись измененного файла. Программа также проверяет входные данные и в случае неправильного ввода выводит сообщения о соответствующих ошибках.

Программа разработана для операционных систем на базе Linux.

1. ЗАДАНИЕ

Вариант 10

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 весь следующий функционал по обработке bmp-файла:

1. Отражение заданной области. Этот функционал определяется:

- Выбором оси относительно которой отражать (горизонтальная или вертикальная)
- Координатами левого верхнего угла области
- Координатами правого нижнего угла области

2. Копирование заданной области. Функционал определяется:

- Координатами левого верхнего угла области-источника
- Координатами правого нижнего угла области-источника
- Координатами левого верхнего угла области-назначения

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

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

4. 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 какой значение ее требуется изменить

2. ХОД РАБОТЫ

2.1. Структуры

Для считывания и записи BMP-файла создаются структуры BitmapFileHeader, BitmapInfoHeader и Rgb со следующими полями:

```
typedef struct BitmapFileHeader{
    unsigned short signature;
    unsigned int filesize;
    unsigned short reserved1;
    unsigned short reserved2;
    unsigned int pixelArrOffset;
}BitmapFileHeader;
```

Рисунок 1 - Структура BitmapFileHeader

```
typedef struct BitmapInfoHeader{
    unsigned int headerSize;
    unsigned int width;
    unsigned int height;
    unsigned short planes;
    unsigned short bitsPerPixel;
    unsigned int compression;
    unsigned int imageSize;
    unsigned int xPixelsPerMeter;
    unsigned int yPixelsPerMeter;
    unsigned int colorsInColorTable;
    unsigned int importantColorCount;
}BitmapInfoHeader;
```

Рисунок 2 - Структура BitmapInfoHeader

```
typedef struct Rgb{
    unsigned char b;
    unsigned char g;
    unsigned char r;
}Rgb;
```

Рисунок 3 - Структура Rgb

2.2. Считывание BMP-файла

Считывание BMP-файла реализуется в функции main. С помощью функции fopen для чтения открывается файл, указанный пользователем. Если при открытии произошла ошибка, программа выводит сообщение на экран и прекращает работу. Далее создаются объекты bmfh типа BitmapFileHeader и bmih типа BitmapInfoHeader, и в их поля записывается информация о файле с помощью функции fread. После этого в переменные H и W типа unsigned int записываются значения полей width(ширина) и height(высота) структуры BitmapInfoHeader соответственно и в двумерный массив arr типа Rgb с помощью цикла for и функции fread записывается информация о каждом пикселе. Для массива память выделяется динамически с использованием функции malloc, для которой подключена библиотека <stdlib.h>. С помощью функции fclose файл закрывается.

При считывании файла учитывается выравнивание.

2.3. Запись BMP-файла

Измененное изображение записывается в новый BMP-файл. Запись реализована в функции main.

С помощью функции fopen создается или открывается (если уже существует) для записи пустой BMP-файл, указанный пользователем. Далее функция fwrite записывает в этот файл информацию об изображении и массив пикселей, и с помощью функции fclose файл закрывается.

При записи файла учитывается выравнивание.

2.4. Функция отражения заданной области.

Функция flip_area принимает на вход следующие данные:

- указатель на строку s, содержащую ось, относительно которой нужно отражать;
- координаты x1 и y1 левого верхнего угла области;
- координаты x2 и y2 нижнего правого угла области;

- значения переменных W и H;
- указатель на массив пикселей arr.

Сначала проверяются входные данные и, если данные некорректны, на экран выводится соответствующее сообщение об ошибке и функция заканчивает работу. С помощью функции strcmp, для которой подключена библиотека <string.h>, определяется ось, относительно которой нужно отражать изображение. Далее во вложенном цикле for на всей заданной области пиксели, симметричные относительно заданной оси, меняются местами, т.е. меняются элементы массива. Так как осей, относительно которых можно отразить изображение, две, то в функции рассматриваются две аналогичные ситуации. Функция ничего не возвращает.

2.5. Функция копирования заданной области

Функция copy_area принимает на вход следующие данные:

- координаты x1 и y1 левого верхнего угла области-источника;
- координаты x2 и y2 правого нижнего угла области-источника;
- координаты x3 и y3 левого верхнего угла области-назначения;
- значения переменных W и H;
- указатель на массив пикселей arr.

Сначала проверяются входные данные и, если данные некорректны, на экран выводится соответствующее сообщение об ошибке и функция заканчивает работу. Далее в переменные h и w типа int записываются значения высоты и ширины области-источника соответственно и создается двумерный массив copy типа Rgb, для которого функцией malloc динамически выделяется память. Во вложенном цикле for в массив copy записываются элементы из массива arr, образующие область, которую необходимо скопировать. В следующем вложенном цикле for в массиве arr те элементы, которые образуют область-назначения, перезаписываются на элементы массива copy. Функция ничего не возвращает.

2.6. Функция замены всех пикселей одного заданного цвета на другой цвет

Функция pixels_color_change принимает на вход следующие данные:

- компоненты r1, g1, b1 цвета, которых необходимо заменить;
- компоненты r2, g2, b2 цвета, на который необходимо заменить;
- значения переменных W и H;
- указатель на массив пикселей arr.

Сначала проверяются входные данные и, если данные некорректны, на экран выводится соответствующее сообщение об ошибке и функция заканчивает работу. Далее во вложенном цикле проверяются все элементы массива и, если поля r, b, g элемента совпадают с r1, g1, b1 соответственно, то этим полям присваиваются значения r2, g2, b2. Также в функции создается переменная flag типа int для проверки наличия введенного цвета. Если значение этой переменной после завершения циклов остается неизменным, на экран выводится сообщением о том, что изображение не содержит выбранный цвет. Функция ничего не возвращает.

2.7. Функция фильтр rgb-компонент

Функция rgb_filter позволяет для всего изображения либо установить в 0, либо установить в 255 значение заданной компоненты, принимает на вход следующие данные:

- указатель на строку s, содержащую название компоненты, которой необходимо изменить;
- значение c, в которое нужно установить заданную компоненту;
- значения переменных W и H;
- указатель на массив пикселей arr.

Сначала проверяются входные данные и, если данные некорректны, на экран выводится соответствующее сообщение об ошибке и функция заканчивает работу. Далее во вложенном цикле для каждого элемента массива полю заданной компоненты устанавливается значение c. Поле,

которое необходимо изменить определяется с помощью функции strcmp. Функция ничего не возвращает.

2.8. Функции вывода информации о файле

Функции printFileHeader(bmfh) и printInfoHeader(bmih) выводят информацию об исходном файле: значения всех полей обоих структур в восьмеричной и десятичной системах счисления.

2.9. Функция main и реализация интерфейса CLI

Функция main содержит два аргумента: int argc – количество параметров, передаваемых функции; char* argv[] – массив параметров.

Для реализации CLI используется библиотека <getopt.h>. Создается массив с короткими ключами char *opts и структура с длинными ключами struct option longOpts[]. Далее для обработки параметров используется функция getopt_long, которая поддерживает одновременно и короткие, и длинные ключи. Возвращаемое значение функции записывается в переменную int opt. Если это значение равно -1, значит параметров нет и на экран выводится инструкция.

Первый элемент массива argv, который содержит введенное пользователем название обрабатываемого файла, копируется в строку char* filename с помощью функции strcpy. Последний элемент содержит название файла, в который нужно записать результат, и копируется в строку char * out_filename.

Далее происходит считывание файла и обработка некоторых входных данных:

- Если исходный файл не был введен при вызове ключей -h/--help, то на экран выводит инструкция;
- Если исходный файл не BMP-формата, программа выводит сообщение об ошибке и прекращает работу;

- Если в исходном файле используется таблица цветов, программа выводит сообщение об ошибке и прекращает работу;
- Если изображение сжато, программа выводит сообщение об ошибке и прекращает работу;

После этого завершается считывание файла и начинается обработка ключей с помощью оператора switch и цикла while, который продолжается до тех пор, пока opt не станет равен -1. Каждый ключ соответствует вызову определенной функции. В случае неправильно введенного ключа, на экран выводится сообщение об ошибке.

Присваивание входных данных переменным для реализации выбранной функции осуществляется с помощью функции sscanf, которая возвращает количество полей, значения которых действительно были присвоены переменным. На основе этого в каждом случае case осуществляется проверка, на наличие всех аргументов, нужных для реализации функции.

Далее происходит проверка на то, введено ли название результирующего файла: если out_filename не совпадает с последним элементов массива argv, измененного после окончания цикла while, то на экран выводится сообщение о том, что результирующий файл не введен, происходит очистка памяти массива и программа завершает работу. В ином случае, происходит запись результирующего файла и очистка памяти.

3. ТЕСТИРОВАНИЕ

Исходное изображение:

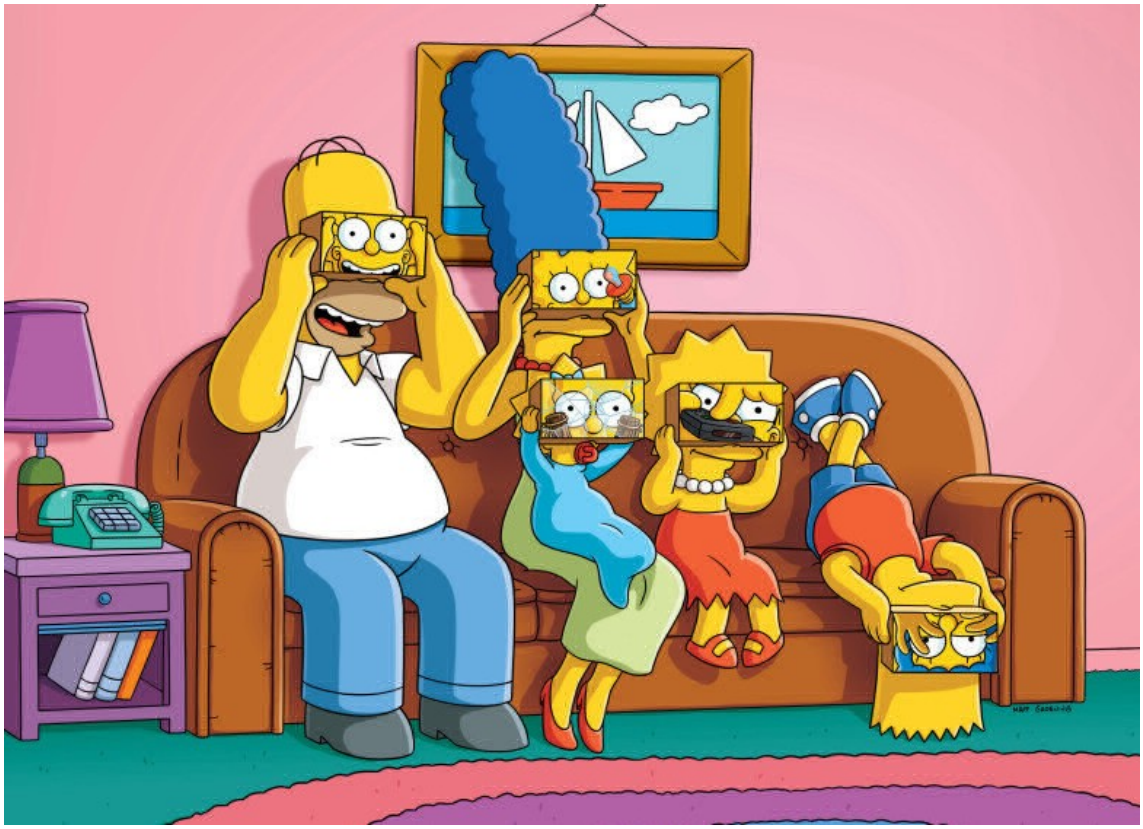

Рисунок 4 - Исходное изображение

3.1. Вывод инструкции

```
oksana@oksana:~/Документы$ ./a.out --help
Программа с CLI по обработке bmp-файла со следующими сведениями:
        24 бита на цвет;
        без сжатия;
        не используется таблица цветов.
Первым аргументом введите название файла (путь), который нужно обработать.
Последним аргументом введите название файла (путь), в который нужно записать результат.
Ключи для вызова доступных функций и описание входных данных:
1) -h/--help: Вывод меню.
2) -f/--flip: Отражение заданной области.
        После ключа введите через запятую координаты левого верхнего угла области, координаты правого н
ижнего угла области и ось, относительно которой отражать('x' - горизонтальная, 'y' - вертикальная).
3) -c/--copy: Копирование заданной области.
        После ключа введите через запятую координаты левого верхнего угла области-источника, координаты
 правого нижнего угла области-источника и координаты левого верхнего угла области-назначения.
4) -p/--pcolor: Заменяет все пиксели одного заданного цвета на другой цвет.
        После ключа введите через запятую цвет (значения rgb-компонент), который требуется заменить, и
цвет, на который требуется заменить.
5) -r/--rgb: Фильтр rgb-компонент. Позволяет для всего изображения либо установить в 0 либо установить
в 255 значение заданной компоненты.
        После ключа введите через запятую значение, в которое нужно установить (0 или 255), и rgb-компо
ненту ('r', 'g', 'b').
6) -i/--info: Вывод информации об исходном файле.
```

Рисунок 5 - Вывод инструкции

3.2. Отражение заданной области

Входные данные: ./a.out simpsonsvr.bmp -f 215,462,576,123,x out.bmp

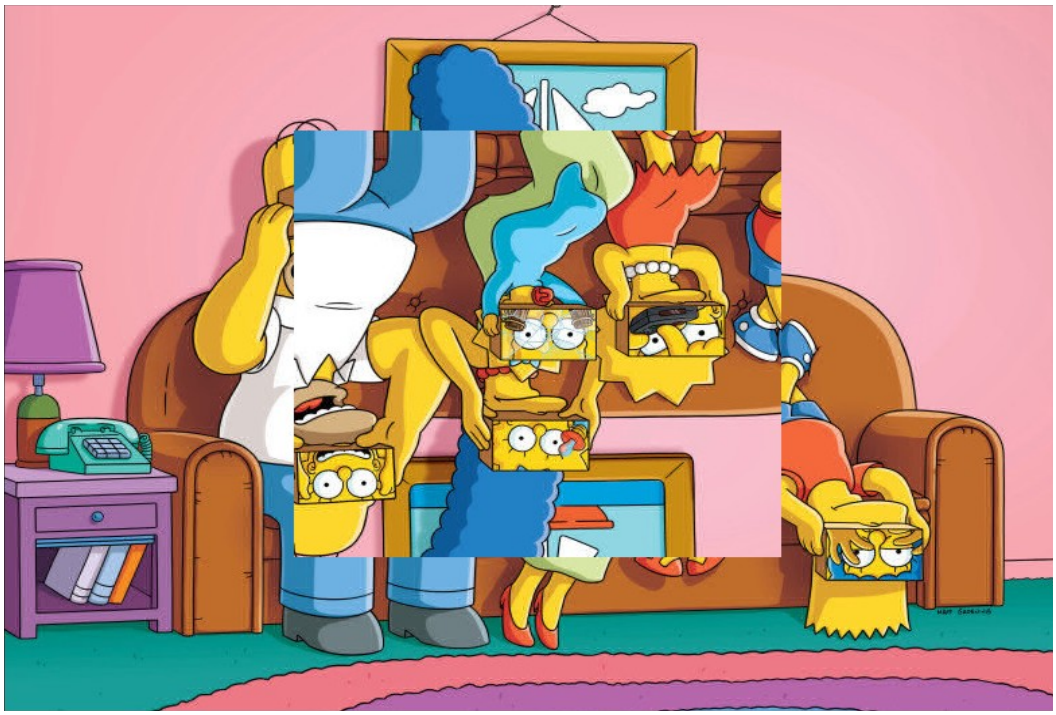

Рисунок 6 - Отражение заданной области

3.3. Копирование заданной области

Входные данные: ./a.out simpsonsvr.bmp --copy 215,462,576,123,130,350 out.bmp

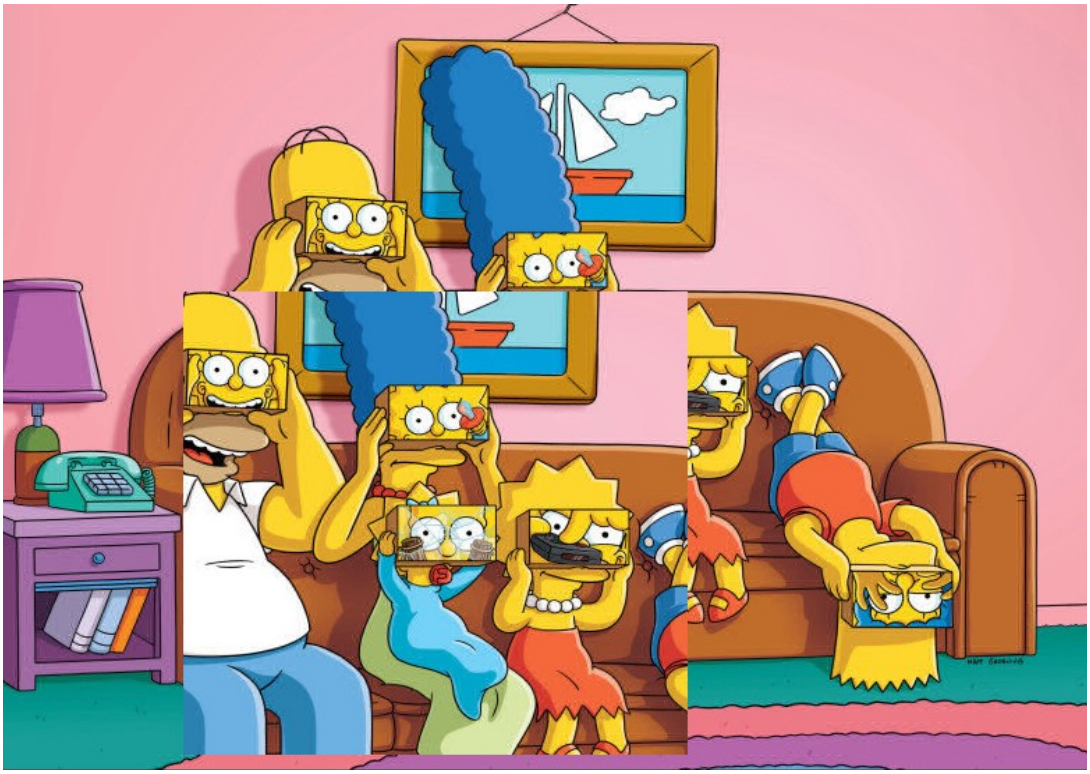

Рисунок 7 - Копирование заданной области

3.4. Замена всех пикселей одного заданного цвета на другой цвет

Входные данные: ./a.out simpsonsvr.bmp -p 255,255,255,255,0,0 out.bmp

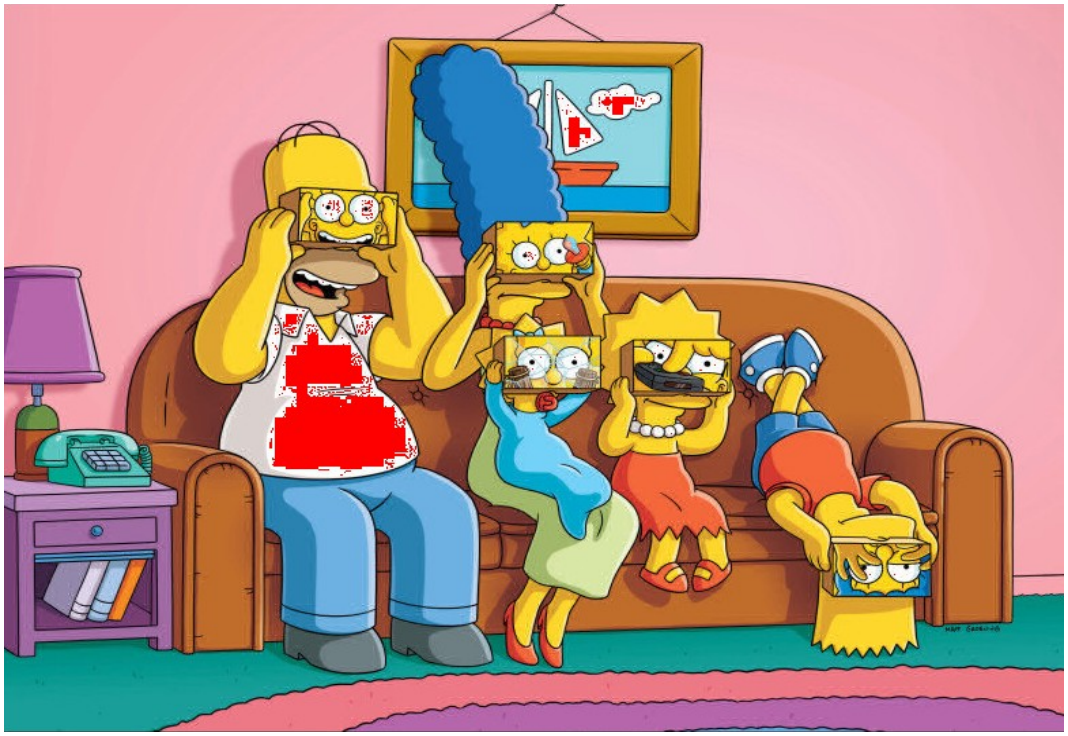

Рисунок 8 - Замена всех пикселей одного заданного цвета на другой цвет

3.5. Фильтр rgb-компонент

Входные данные: ./a.out simpsonsvr.bmp --rgb 255,r out.bmp

Рисунок 9 - Фильтр rgb-компонент

3.6. Вывод информации о файле

```
oksana@oksana:~/Документы$ ./a.out simpsonsvr.bmp -i out.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)
```

Рисунок 10 - Вывод информации о файле

3.7. Примеры обработок ошибок

```
oksana@oksana:~/Документы$ ./a.out simpsonsvr.bmp --copy 215,462,576,123 out.bmp
Ошибка (-c/--copy): не хватает аргументов для выполнения функции.
```

Рисунок 11 - Ошибка: недостаточно аргументов для выполнения функции

```
oksana@oksana:~/Документы$ ./a.out simpsonsvr.bmp --flip 645,462,576,123,x out.bmp
Ошибка (-f/--flip): координата x правого нижнего угла должна быть больше координаты x левого верхнего у
гла).
```

Рисунок 12 - Ошибка: введенные координаты некорректны

```
oksana@oksana:~/Документы$ ./a.out simpsonsvr.bmp --pcolor 255,255,255,255,0,0
Не введено название результирующего файла.
```

Рисунок 13 - Ошибка: не введено название результирующего файла

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была создана программа для обработки BMP-файла с использованием интерфейса CLI. Программа может обрабатывать изображения со следующими свойствами:

- 24 бита на цвет;
- без сжатия;
- не содержит таблицу цветов.

Программа реализует функции в соответствии с введенными пользователем ключами:

1) -h/--help – вывод меню;
2) f/--flip – отражение заданной области;
3) -c/--copy – копирование заданной области;
4) -p/--pcolor – заменяет все пиксели одного заданного цвета на другой цвет;
5) -r/--rgb – фильтр rgb-компонент. Позволяет для всего изображения либо установить в 0, либо установить в 255 значение заданной компоненты;
6) -i/--info – вывод информации об исходном файле.

Программа поддерживает длинные и короткие ключи.

Программа была успешно протестирована на работоспособность.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ciplusplus URL: http://cplusplus.com
2. https://www.gnu.org/software/libc/manual/html_node/Getopt.html

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: pr_cw_2.c

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

#pragma pack(push, 1)
typedef struct BitmapFileHeader{
    unsigned short signature;
    unsigned int filesize;
    unsigned short reserved1;
    unsigned short reserved2;
    unsigned int pixelArrOffset;
}BitmapFileHeader;

typedef struct BitmapInfoHeader{
    unsigned int headerSize;
    unsigned int width;
    unsigned int height;
    unsigned short planes;
    unsigned short bitsPerPixel;
    unsigned int compression;
    unsigned int imageSize;
    unsigned int xPixelsPerMeter;
    unsigned int yPixelsPerMeter;
    unsigned int colorsInColorTable;
    unsigned int importantColorCount;
}BitmapInfoHeader;
#pragma pack(pop)

void printFileHeader(BitmapFileHeader header){
                printf("signature:\t%x    (%hu)\n",    header.signature,
header.signature);
    printf("filesize:\t%x (%u)\n", header.filesize, header.filesize);
                printf("reserved1:\t%x    (%hu)\n",    header.reserved1,
header.reserved1);
                printf("reserved2:\t%x    (%hu)\n",    header.reserved2,
header.reserved2);
          printf("pixelArrOffset:\t%x  (%u)\n",  header.pixelArrOffset,
header.pixelArrOffset);
}

void printInfoHeader(BitmapInfoHeader header){
               printf("headerSize:\t%x    (%u)\n",    header.headerSize,
header.headerSize);
    printf("width:\t%x (%u)\n", header.width, header.width);
    printf("height:\t%x (%u)\n", header.height, header.height);
    printf("planes:\t%x (%hu)\n", header.planes, header.planes);
            printf("bitsPerPixel:\t%x   (%hu)\n",   header.bitsPerPixel,
header.bitsPerPixel);
              printf("compression:\t%x   (%u)\n",    header.compression,
header.compression);
```

```
                printf("imageSize:\t%x    (%u)\n",    header.imageSize,
header.imageSize);
        printf("xPixelsPerMeter:\t%x  (%u)\n",  header.xPixelsPerMeter,
header.xPixelsPerMeter);
        printf("yPixelsPerMeter:\t%x  (%u)\n",  header.yPixelsPerMeter,
header.yPixelsPerMeter);
                          printf("colorsInColorTable:\t%x      (%u)\n",
header.colorsInColorTable, header.colorsInColorTable);
                         printf("importantColorCount:\t%x      (%u)\n",
header.importantColorCount, header.importantColorCount);
}

typedef struct Rgb{
    unsigned char b;
    unsigned char g;
    unsigned char r;
}Rgb;

void print_help(){
      printf("Программа с CLI по обработке bmp-файла со следующими
сведениями:\n");
    printf("\t24 бита на цвет;\n");
    printf("\tбез сжатия;\n");
    printf("\tне используется таблица цветов.\n");
     printf("Первым аргументом введите название файла (путь), который
нужно обработать.\n");
      printf("Последним аргументом введите название файла (путь), в
который нужно записать результат.\n");
     printf ("Ключи для вызова доступных функций и описание входных
данных:\n");
    printf("1) -h/--help: Вывод меню.\n");
    printf("2) -f/--flip: Отражение заданной области.\n"
             "\tПосле ключа введите через запятую координаты левого
верхнего угла области, координаты правого нижнего угла области и ось,
относительно   которой   отражать('x'   -   горизонтальная,   'y'   -
вертикальная).\n");
    printf("3) -c/--copy: Копирование заданной области.\n"
             "\tПосле ключа введите через запятую координаты левого
верхнего угла области-источника, координаты правого нижнего угла
области-источника и координаты левого верхнего угла области-
назначения.\n");
      printf("4) -p/--pcolor: Заменяет все пиксели одного заданного
цвета на другой цвет.\n"
            "\tПосле ключа введите через запятую цвет (значения rgb-
компонент), который требуется заменить, и цвет, на который требуется
заменить.\n");
     printf("5) -r/--rgb: Фильтр rgb-компонент. Позволяет для всего
изображения либо установить в 0 либо установить в 255 значение
заданной компоненты.\n"
            "\tПосле ключа введите через запятую значение, в которое
нужно установить (0 или 255), и rgb-компоненту ('r', 'g', 'b').\n");
    printf("6) -i/--info: Вывод информации об исходном файле.\n");
}

void flip_area(char* s, int x1, int y1, int x2, int y2, unsigned int
W, unsigned int H, Rgb **arr){

    if (x1 > W || y1 > H || x1<0 || y1<0){
```

```
            printf("Ошибка (-f/--flip): выбранные координаты левого
верхнего угла находятся за пределами изображения.\n");
        return;
    }
    if (x2 > W || y2 > H || x2<0 || y2<0){
            printf("Ошибка (-f/--flip): выбранные координаты правого
нижнего угла находятся за пределами изображения.\n");
        return;
    }
    if (x2 < x1){
         printf("Ошибка (-f/--flip): координата x правого нижнего угла
должна быть больше координаты x левого верхнего угла).\n");
        return;
    }
    if (y2 > y1){
         printf("Ошибка (-f/--flip): координата y правого нижнего угла
должна быть меньше координаты y левого верхнего угла).\n");
        return;
    }
    if (strcmp(s, "x")!=0 && strcmp(s, "y")!=0){
        printf("Ошибка (-f/--flip): выбранная ось недоступна.\n");
        return;
    }

    Rgb a;
    if (strcmp(s, "x")==0) {
        for (int i = y2; i <= (y1+y2)/2; i++) {
            for (int j=x1; j<=x2; j++) {
                a = arr[i][j];
                arr[i][j] = arr[y2+y1-i][j];
                arr[y2+y1-i][j] = a;
            }
        }
    }
    else if (strcmp(s, "y")==0){
        for (int i = y2; i < y1; i++) {
            for (int j = x1; j <= (x1 + x2) / 2; j++) {
                a = arr[i][j];
                arr[i][j] = arr[i][x1 + x2 - j];
                arr[i][x1 + x2 - j] = a;
            }
        }
    }
}

void copy_area(int x1, int y1, int x2, int y2, int x3, int y3,
unsigned int W, unsigned int H, Rgb **arr){

    if (x1 > W || y1 > H || x1<0 || y1<0){
            printf("Ошибка (-c/--copy): выбранные координаты левого
верхнего угла находятся за пределами изображения.\n");
        return;
    }
    if (x2 > W || y2 > H || x2<0 || y2<0){
            printf("Ошибка (-c/--copy): выбранные координаты правого
нижнего угла находятся за пределами изображения.\n");
        return;
    }
```

```
    if (x2 < x1){
         printf("Ошибка (-c/--copy): координата x правого нижнего угла
должна быть больше координаты x левого верхнего угла).\n");
        return;
    }
    if (y2 > y1){
         printf("Ошибка (-c/--copy): координата y правого нижнего угла
должна быть меньше координаты y левого верхнего угла).\n");
        return;
    }
    if (x3 > W || y3 > H || x3<0 || y3<0){
            printf("Ошибка (-c/--copy): выбранные координаты левого
верхнего угла области-назначения находятся за пределами изображения.\
n");
        return;
    }
    if ((x2-x1)>W-x3 || (y1-y2)>y3){
            printf("Ошибка (-c/--copy): выбранная область-назначения
меньше области-источника.\n");
        return;
    }

    int h = y1-y2+1;
    int w = x2-x1+1;
    Rgb** copy = malloc(h*sizeof(Rgb*));
    for (int i=0; i<h; i++){
        copy[i] = malloc(w*sizeof(Rgb));
    }
    int x = 0;
    int y = 0;
    for (int i=y2; i<=y1; i++){
        x=0;
        for (int j=x1; j<=x2; j++){
            copy[y][x] = arr[i][j];
            x++;
        }
        y++;
    }
    int m = 0;
    int n;
    for (int i=y3-y+1; i<=y3; i++) {
        n = 0;
        for (int j = x3; j < x3 + x; j++) {
            arr[i][j] = copy[m][n];
            n++;
        }
        m++;
    }
}

void pixels_color_change(int r1, int g1, int b1, int r2, int g2, int
b2, unsigned int H, unsigned int W, Rgb** arr){
    if (r1>255 || r1<0 || g1>255 || g1<0 || b1>255 || b1<0){
            printf("Ошбика (-p/--pcolor): выбранные параметры цвета,
который нужно заменить, находятся вне допустимых границ.\n");
        return;
    }
    if (r2>255 || r2<0 || g2>255 || g2<0 || b2>255 || b2<0){
```

```
          printf("Ошбика (-p/--pcolor): выбранные параметры цвета, на
который нужно заменить, находятся вне допустимых границ.\n");
        return;
    }
    int flag = 1;
    for (int i = 0; i<H; i++){
        for (int j = 0; j<W; j++){
             if (arr[i][j].r == r1 && arr[i][j].g == g1 && arr[i][j].b
== b1){
                arr[i][j].r = r2;
                arr[i][j].g = g2;
                arr[i][j].b = b2;
                flag = 0;
            }
        }
    }
    if (flag){
          printf("Ошбика (-p/--pcolor): данное изображение не содержит
выбранного цвета.\n");
    }
}

void rgb_filter(char* s, int c, unsigned int H, unsigned int W, Rgb**
arr){
    if (c>255 || c<0){
         printf("Ошибка (-r/--rgb): выбранные параметры цвета находятся
вне допустимых границ.\n");
        return;
    }
    if (strcmp(s, "r")!=0 && strcmp(s, "g")!=0 && strcmp(s, "b")!=0){
          printf("Ошибка (-r/--rgb): выбранный цвет отсутствует в rgb-
компоненте.\n");
        return;
    }
    for (unsigned int i=0; i<H; i++){
        for (unsigned int j=0; j<W; j++){
            if (strcmp(s, "r")==0) {
                arr[i][j].r = c;
            }
            else if (strcmp(s, "g")==0) {
                arr[i][j].g = c;
            }
            else if (strcmp(s, "b")==0) {
                arr[i][j].b = c;
            }
        }
    }

}


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

    char *opts = "f:c:p:r:ih";
    struct option longOpts[]={
            {"flip", required_argument, NULL, 'f'},
            {"copy", required_argument, NULL, 'c'},
            {"pcolor", required_argument, NULL, 'p'},
```

```
            {"rgb", required_argument, NULL, 'r'},
            {"info", no_argument, NULL, 'i'},
            {"help", no_argument, NULL, 'h'},
            {NULL, 0, NULL, 0}
    };

    int opt;
    int longIndex;
    opt = getopt_long(argc, argv, opts, longOpts, &longIndex);

    if (opt == -1){
        print_help();
        return 0;
    }

    char* filename = malloc((strlen(argv[1]))*sizeof(char));
    strcpy(filename, argv[1]);
    char * out_filename = malloc((strlen(argv[argc-1]))*sizeof(char));
    strcpy(out_filename, argv[argc-1]);

    FILE *f = fopen(filename, "rb");
    if (!f){
        if (opt == 'h') {
            print_help();
            return 0;
        }
        else {
            printf("Ошибка при чтении исходного файла.\n");
            return 0;
        }
    }
    BitmapFileHeader bmfh;
    BitmapInfoHeader bmih;
    fread(&bmfh, 1, sizeof(BitmapFileHeader), f);
    fread(&bmih, 1, sizeof(BitmapInfoHeader), f);
    if (bmfh.signature != 0x4d42){
         printf("Файл не соотвествует формату bmp. Пожалуйста, выберите
другой файл.\n");
        return 0;
    }
    if (bmih.bitsPerPixel != 24){
           printf("Глубина цвета данного файла не 24 пикселя на бит.
Пожалуйста, выберите другой файл.\n");
        return 0;
    }
    if (bmih.colorsInColorTable != 0){
             printf("Для данного файла используется таблица цветов.
Пожалуйста,выберите другой файл.\n");
        return 0;
    }
    if (bmih.compression != 0){
         printf("Изображение сжатое. Пожалуйста, выберите другой файл.\
n");
        return 0;
    }

    unsigned int H = bmih.height;
    unsigned int W = bmih.width;
```

```
    Rgb **arr = malloc(H*sizeof(Rgb*));
    for( int i=0; i<H; i++){
        arr[i] = malloc(W*sizeof(Rgb)+(W*3)%4);
        fread(arr[i], 1, W*sizeof(Rgb)+(W*3)%4, f);
    }
    fclose(f);


    while(opt!=-1){
        switch(opt){
            case 'f': {
                char s[50];
                int x1, y1, x2, y2;
                 int count = sscanf(optarg, "%d,%d,%d,%d,%s", &x1, &y1,
&x2, &y2, s);
                if (count<5){
                      printf("Ошибка (-f/--flip): не хватает аргументов
для выполнения функции.\n");
                    break;
                }
                s[1] = '\0';

                flip_area(s, x1, y1, x2, y2, W, H, arr);
                break;
            }
            case 'c': {
                int x1, y1, x2, y2, x3, y3;
                  int count = sscanf(optarg, "%d,%d,%d,%d,%d,%d", &x1,
&y1, &x2, &y2, &x3, &y3);
                if (count<6){
                      printf("Ошибка (-c/--copy): не хватает аргументов
для выполнения функции.\n");
                    break;
                }
                copy_area(x1, y1, x2, y2, x3, y3, W, H, arr);
                break;
            }
            case 'p': {
                int r1, g1, b1, r2, g2, b2;
                  int count = sscanf(optarg, "%d,%d,%d,%d,%d,%d", &r1,
&g1, &b1, &r2, &g2, &b2);
                if (count<6){
                                printf("Ошибка (-p/--pcolor): не хватает
аргументов для выполнения функции.\n");
                    break;
                }
                    pixels_color_change(r1, g1, b1, r2, g2, b2, H, W,
arr);
                break;
            }
            case 'r': {
                int c;
                char s[50];
                int count = sscanf(optarg, "%d,%s", &c, s);
                if (count<2){
                       printf("Ошибка (-r/--rgb): не хватает аргументов
для выполнения функции.\n");
```

```
                    break;
                }
                s[1] = '\0';
                rgb_filter(s, c, H, W, arr);
                break;
            }
            case 'i': {
                printFileHeader(bmfh);
                printInfoHeader(bmih);
                break;
            }
            case 'h':{
                print_help();
                break;
            }
            default:{
                printf("Нет такого ключа.\n");
                break;
            }
        }
        opt = getopt_long(argc, argv, opts, longOpts, &longIndex);
    }


    if (strcmp(out_filename, argv[argc-1])!=0){
        printf("Не введено название результирующего файла.\n");
        for (int i=0; i<H; i++){
            free(arr[i]);
        }
        free(arr);
        return 0;
    }

    FILE *ff = fopen(out_filename, "wb");
    fwrite(&bmfh, 1, sizeof(BitmapFileHeader), ff);
    fwrite(&bmih, 1, sizeof(BitmapInfoHeader), ff);
    unsigned int w = W*sizeof(Rgb)+(W*3)%4;
    for (int i=0; i<H; i++){
        fwrite(arr[i], 1, w, ff);
    }
    fclose(ff);
    for (int i=0; i<H; i++){
        free(arr[i]);
    }
    free(arr);

    return 0;
}
```