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 _____ Чегодаева Е.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2021

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студентка Чегодаева Е.А.

Группа 0382.

Тема работы: Обработка BMP-файла на языке Си.

Вариант 11.

Программа **должна** иметь CLI или GUI.
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 какой 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. -- по желанию студента (можно и оба варианта). Функционал определяется:
    - Количество частей по “оси” Y
    - Количество частей по “оси” X
    - Толщина линии
    - Цвет линии
    - Либо путь куда сохранить кусочки

Содержание пояснительной записки:

«Содержание», «Введение», «Задание работы», «Ход выполнения работы», «Тестирование», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 05.04.2020

Дата сдачи реферата: 22.05.2020

Дата защиты реферата: 24.05.2020

Студентка гр. 0382 _____ Чегодаева Е.А.

Преподаватель _____ Берленко Т.А.

АННОТАЦИЯ

В курсовой работе была реализована программа для обработки BMP-изображения в зависимости от команд пользователя.

Программа получает на вход изображение и после выполнения задач создаёт новый BMP-файл (название может задать пользователь). Функционал определяется рисованием круга (с возможной заливкой), заменой параметра одного из цветов, делением изображения на N*M частей.

Осуществлён интерфейс CLI, предусмотрены сообщения о возможных ошибках при неверном вводе, формате и т.д.

Пример работы программы приведён в приложении А.
Код приведён в приложении В.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

Цель работы: создать программу, получающую изображение BMP-формата и обрабатывающую его в соответствии с командами пользователя.

Задачи:

1. Реализовать корректное считывание и хранение BMP-файла;
2. Считать информацию о формате и данных изображения;
3. Реализовать оператор switch{};
4. Произвести операции:

   4.1. 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

   4.2. 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

   4.3. Разделяет изображение на N*M частей. Реализация: либо провести линии заданной толщины, тем самым разделив изображение. Функционал определяется:
   - Количество частей по “оси” Y
   - Количество частей по “оси” X
   - Толщина линии
   - Цвет линии

5. Реализовать корректный вывод: создать изображение, повторяющее первоначальное, но с изменениями в зависимости от пользовательских команд;

6. Реализовать интерфейс общения с пользователем: Произвести корректное считывание команд в “коротком” и “длинном” форматах;
7. Оформить сообщения о различных ошибках, которые могут возникнуть в ходе работе программы;

Программа поддерживает BMP-изображения глубиной цвета 24 бита, без сжатия. Для обработки необходимо, чтобы изображение хранилось в одной директории с программой. Интерфейс предполагает знание пользователя о порядке и формате ввода данных (для ознакомления существует команда --help (-h) с подробным описанием всех возможных функций). При возникновении ошибок различного типа программа определяет этот тип и печатает пользователю подсказки для их устранения (ошибки при вводе файла неподдерживаемого формата, ошибки при вводе данных для обработки). Обработка, связанная с цветом, основана на аддитивной цветовой модели (RGB). При инициализации переменных использовались целочисленные типы данных фиксированного размера.

Считывание (отсчёт) пикселей осуществляется с левого нижнего угла изображения — это стоит учитывать при вводе координат в различных командах.

1. ЗАДАНИЕ

Программа **должна** иметь CLI или GUI.

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 какой 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. -- по желанию студента (можно и оба варианта). Функционал определяется:
    - Количество частей по “оси” Y
    - Количество частей по “оси” X
    - Толщина линии
    - Цвет линии
    - Либо путь куда сохранить кусочки

3. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

3.1. Структуры и глобальные переменные, реализованные в программе:

- BitmapFileHeader — информация о формате файле;
- BitmapInfoHeader — информация о данных файле;
- Rgb — цветовая палитра (красный, зелёный, синий);
- option long_options[] — хранит команды, используемые в интерфейсе взаимодействия с пользователем в “коротком” и “длинном” форматах;
- Rgb** arr — массив пикселей, с которым происходят все взаимодействия;
- unsigned int H — поле структуры BitmapFileHeader, хранит высоту изображения;
- unsigned int W— поле структуры BitmapFileHeader, хранит ширину изображения;

3.2. Считывание изображения

Предполагается, что изображение для обработки записывается последним параметром в командной строке. Исходя из этого, создаётся строка *inputFile,* в которой храниться название файла. Посредствам функции *fopen()* происходит считывание изображения. Далее так же осуществляется считывание данных об изображении в соответствующие поля структур. Каждый пиксель записывается элементом массива *arr[]* посредством двух циклов *for(),* на каждой интеграции происходит выправные.

3.3. Создание файла для вывода результата

Посредством функции *fopen()* создаётся файл с пользовательским названием (по умолчанию - out.bmp). Файл полностью повторяет значения

высоты и ширины изначального изображения. Далее в новый файл записываются значения каждого пикселя обрабатываемого изображения после выполнения всех подзадач.

3.4. Создание интерфейса

CLI - Command Line Interface:

В функцию *main()* поступают *argc* — количество аргументов, *argv* — строка с аргументами. Инициализируется переменная *option_index,* которая хранит индексы каждой команды, для дальнейшей обработки. Далее посредством *getopt_long()* в программу поступает определённая команда (с возможными необходимыми данными). Строка " static const char *optString = "hip:o:s:r:c:l:"; " обозначает какие команды предусматривают ввод дополнительных данных, а какие нет. Программа поддерживает корректное считывание команд в "коротком" и "длинном" форматах.

3.5. Выбор команды

С помощью оператора switch{}, реализован выбор команды пользователя. Оператор хранит как технические составляющие (возможность вывода информации о возможностях программы и правила ввода, возможность вывода информации об изображении, возможность создания файла для вывода с названием, которое задаёт пользователь), так и команды, основанные на задачах работы (рисование круга, замена RGB-компоненты, деление изображения). Оператор реализован через получение команд и данных через строку в терминале. Завершение считывания осуществляется при помощи цикла *while().* После каждой итерации осуществляется получение следующей команды, если такая существует и обнуление *option_index* для корректной обработки следующих аргументов. Осуществлена возможность ввода нескольких команд с соответствующими параметрами друг за другом.

3.6. Операции команды '-h'

Команда служит для ознакомления пользователя с программой и её функционалом. Данная команда может быть вызвана как в ручною, так и при компиляции программы без аргументов.

3.7. Операции команды '-o'

Команда получает название файла для вывода. В дальнейшем этот файл будет создан в данной директории. Файл будет повторять исходные данные обкатываемого изображения с изменениями, внесёнными в ходе выполнения других команд.

3.8. Операции команды '-i'

Команда для получения пользователем информации об изображении. Для этого реализованы для функции *printFileHeader()* и *printInfoHeader(bmif).*

3.9. Операции команды '-c'

Команда для замены RGB-компоненты. Команда получает наименование цвета *colorName*, который необходимо изменить и значение *Value*, в которое этот цвет необходимо установить. Далее вызывается функция *ChangeColor().*

Функция ChangeColor():

Посредствам двух циклов *for()* и условного оператора *if()* осуществляется поиск необходимого цвета и устанавливает его цвета в заданное значение.

3.10. Операции команды '-l'

Команда для рисования на изображении N*M линий заданного цвета и толщины, тем самым деления изображения на части.

Считывание происходит в массив *inletL[]*. Далее каждый элемент этого массива присваивается соответствующей переменной для удобства дальнейшей работы. Затем вызывается функция *DrawLines().*

Функция DrawLines():

Внутри функции посредствам двух циклов *for()* сначала рисуются прямые заданной толщины и цвета по вертикальной оси и затем аналогичным алгоритмом по вертикальной.

3.11. Операции команды '-r'

Команда для рисования круга по заданному центру и радиусу. Так же задаётся толщина и цвет окружности.

Считывание происходит в массив *inletR[]*. Далее каждый элемент этого массива присваивается соответствующей переменной для удобства дальнейшей работы. Затем вызывается функция *DrawCircle().*

Функция DrawCircle():

Внутри функции посредствам двух циклов *for() и* арифметического уравнения окружности - рисуется круг заданной толщины путем замены цвета пикселей по данному параметру.

3.12. Операции команды '-s'

Команда для рисования вписанной окружности в квадрат, по заданным координатам: левого верхнего и нижнего правого углов. Так же задаётся толщина и цвет окружности.

Считывание происходит в массив *inletS[]*. Далее каждый элемент этого массива присваивается соответствующей переменной для удобства дальнейшей работы. Рассчитывается центр квадрата и получившийся радиус окружности. Затем вызывается функция *DrawCircle().*

3.13. Операции команды '-p'

Команда для закрашивания ранее нарисованной окружности. НЕ БУДЕТ РЕАЛИЗОВАНА БЕЗ ВЫЗОВА ОДНОЙ ИЗ ДВУХ РАНЕЕ ИДУЩИХ ФУНКЦИЙ. Принимает только цвет окружности по 3 компонентам.

Считывание происходит в массив *inletP[]*. Затем вызывается функция *PaintCircle().*

Функция PaintCircle():

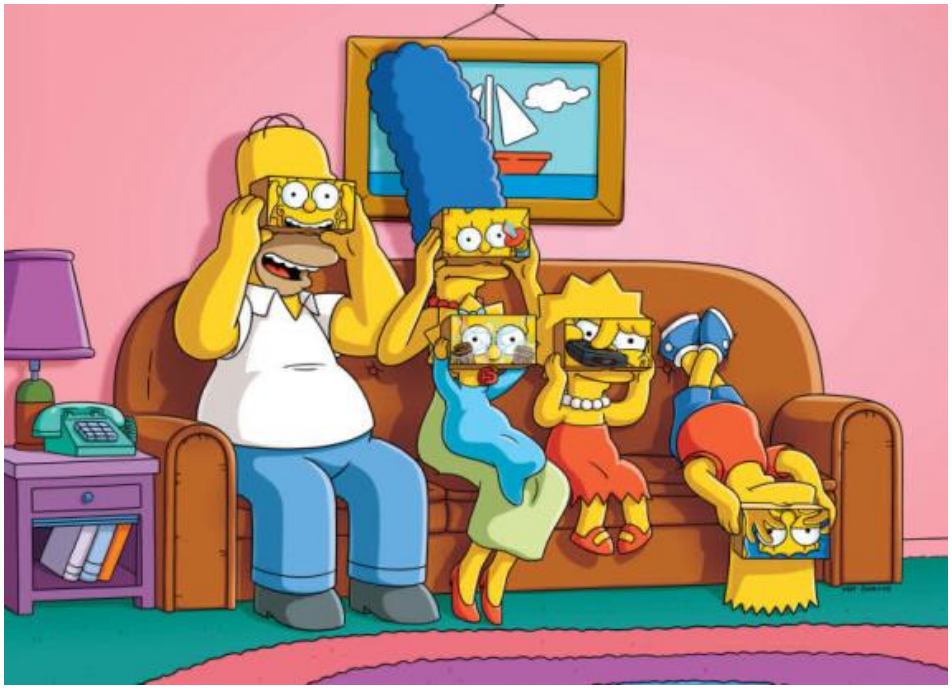
Внутри функции посредствам двух циклов *for() и* арифметического уравнения окружности (данные для значений уравнения сохраняются из прошлых функций) закрашивается участок изображения, находящийся внутри заданного круга путем замены каждого цвета пикселей по данному параметру.

3.14. Исключительные ситуации

На каждом шагу выполнения проверяются возможные ситуации, которые могут привести к ошибкам в работе программы. При обнаружении таковой — печатается сообщение о данной ошибке. Проверка соблюдения возможного диапазона при вводе цвета реализована в функции *chekRGB(),* остальные проверки находятся в местах их возможного возникновения. В каждом case оператора *switch{}* переменная *Count*, инициализированная ранее хранит соответствующее значение — количество данных, необходимых для ввода в той или иной команде. Это позволяет сообщить о недостатке входных данных.

4. ТЕСТИРОВАНИЕ

Изначальное изображение:



1. Ввод:

   ./a.out qwerqwerqwerqwerqwerqwer

   Результат:

   Please add a file.

   - Верно

2. Ввод:

   ./a.out -c t 1000 input.bmp

   Результат:

   Please enter the color again.

   - Верно

3. Ввод:

   ./a.out -c r 100 input.bmp

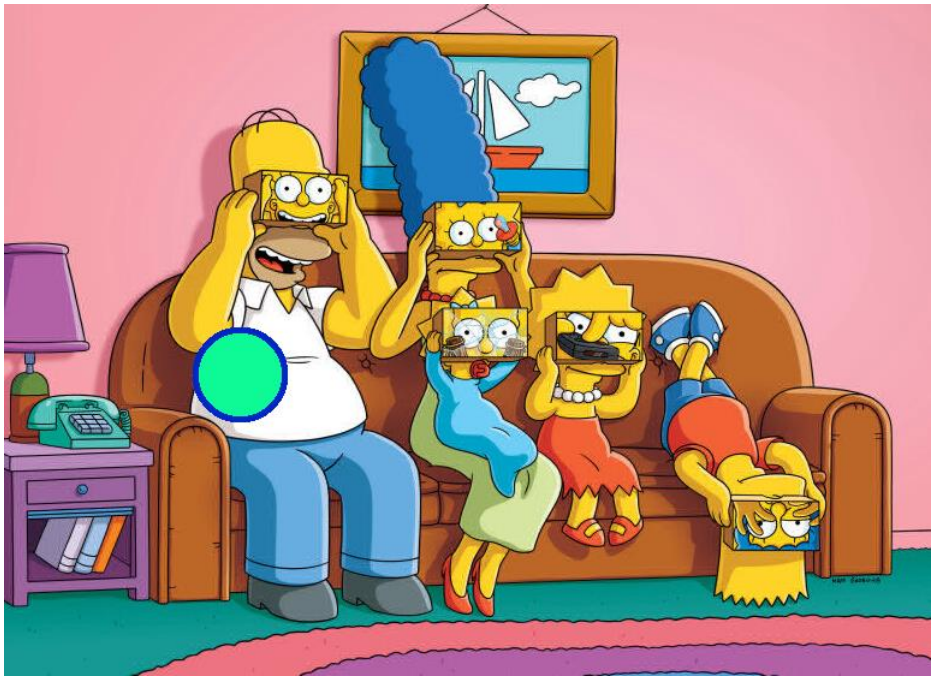
   Результат:

- Верно

4. Ввод:

   ./a.out -r 200 250 40 5 0 40 200 -p 13 250 150 input.bmp

   Результат:



- Верно

5. Ввод:

   ./a.out -p 155 155 100 input.bmp

   Результат:

   This option is available only if there is a drawn circle.

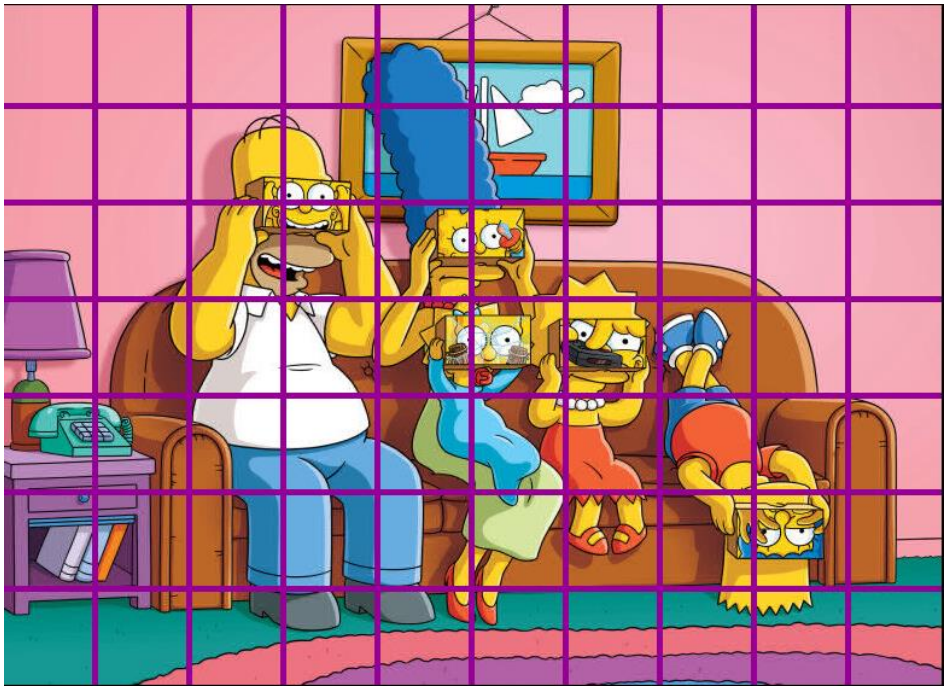
   Try again!

   - Верно

6. Ввод:

   ./a.out -l 10 7 4 150 0 150 input.bmp

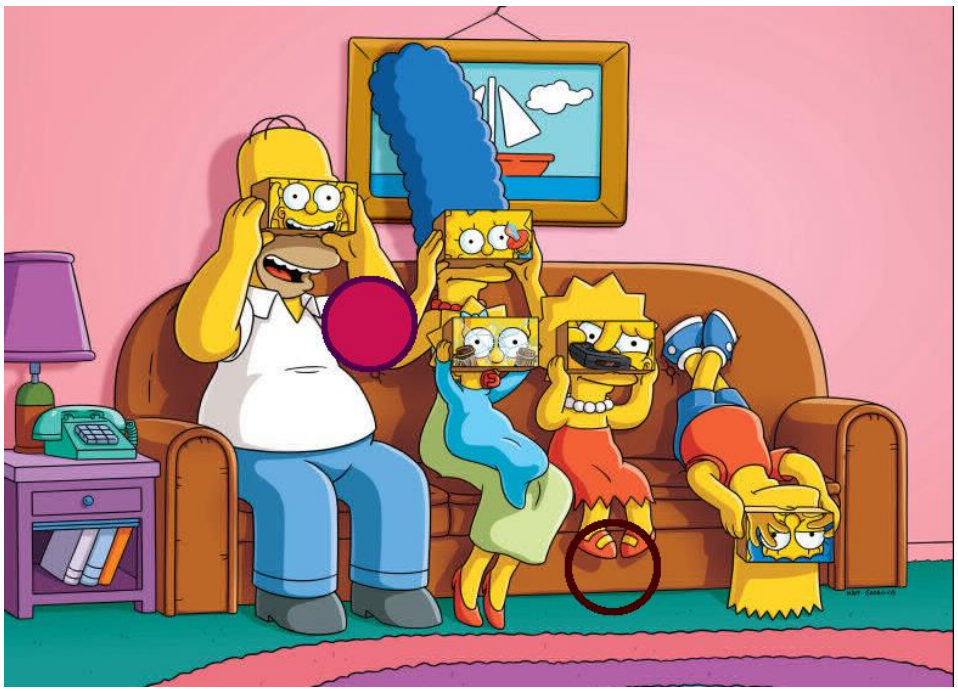
   Результат:



   - Верно

7. Ввод: ./a.out -r 300 300 40 5 80 0 80 -p 200 15 80 -r 500 100 40 6 70 0 0 input.bmp

   Результат:



- Верно

8. Ввод:

   ./a.out -r 300 300 40 5 80 0 80 -p 200 15 80 -r 500 100 40 6 70 0 0 input.jpg

   Результат:

   Unsupported image format.

   Try again!

   - Верно

5. ЗАКЛЮЧЕНИЕ

В процессе курсовой работы была создана корректно работающее приложение по обработке изображений BMP-формата на языке Си. Реализован CLI — интерфейс взаимодействия с пользователем. Функционал включает: рисование круга, замену RGB-компоненты и деление изображения на части посредством рисования линий. Выполнено выравнивание. Исключены ошибки при неверном вводе данных пользователем.

При реализации использовались библиотеки: <stdio.h>, <stdlib.h>, <unistd.h>, <getopt.h>, <string.h>, <stdint.h>.

6. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. vsokovikov.narod.ru: http://www.vsokovikov.narod.ru/New_MSDN_API/Bitmaps/str_bitmapinfoheader.htm

2. Cplusplus URL: https://www.cplusplus.com.

3. firststeps.ru: https://firststeps.ru/linux/r.php?11 https://firststeps.ru/linux/r.php?12

4. Керниган Б. и Ритчи Д. Язык программирования Си. М.: Вильямс, 1978 288 с.

ПРИЛОЖЕНИЕ А
ПРИМЕР РАБОТЫ ПРОГРАММЫ

Изначальный файл:

input.bmp

- ./a.out

```
.....................................................................
This CLI-enabled program handles images in BMP format,
uncompressed and 24-bit color.
.....................................................................
Information about possible commands:     -h,  --help

Image information:       -i,  --info

Entering the file name to display the result:    -o,  --output
( -o filename.bmp)
Default: out.bmp

Filter RGB-components:   -c,  --color
Input requirements:      replacement color, parameter.
( -c [r/g/b] [0<k<255] )

Divide the image into NxM parts:         -l,  --line
Input requirements:      number of horizontal and vertical axes,
line thickness, color (for each parameter).
( -l [Ox] [Oy] [thick] [red] [green] [blue] )

Draw a circle inscribed in a square:     -s,  --square_circle
Input requirements:      Coordinates of the upper left (along the x and y axes)
and lower right (along the x and y axes) corner of the squarer,
line thickness, color (for each parameter).
( -s [LVw] [LVh] [PNw] [PNh] [thick] [red] [green] [blue] )

Draw circle in center and radius:        -r,  --rad_circle
Input requirements:      Center coordinates (along the x and y axes)
radius, line thickness, color (for each parameter).
( -r [point_w] [point_h] [rad] [thick] [red] [green] [blue] )

Paint over the circle (if available):    -p,  --paint_circle
Input requirements:      color options.
( -p [thick] [red] [green] [blue] )
Please try again
```

- ./a.out -c b 0 input.bmp

out.bmp

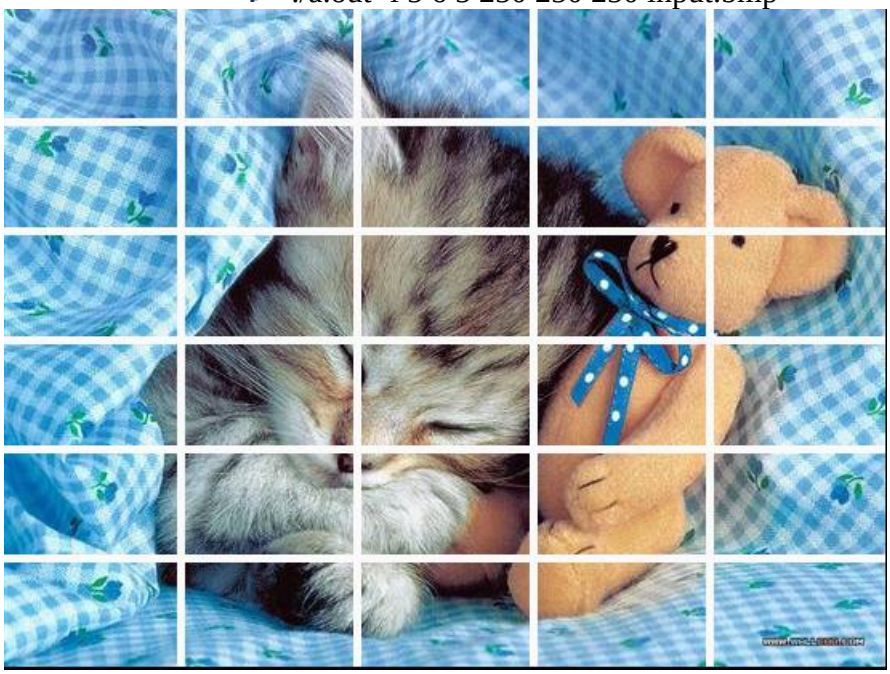
- ./a.out -r 480 100 30 3 100 0 100 -p 150 150 150 input.bmp

out.bmp

- ./a.out -s 200 200 300 300 3 80 15 155 -o kotik.bmp input.bmp

kotik.bmp

- ./a.out -l 5 6 5 250 250 250 input.bmp



out.bmp

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: cw.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <getopt.h>
#include <string.h>
#include <stdint.h>
#pragma pack (push, 1)

static const char *optString = "hip:o:s:r:c:l:";
char * outputFile  = "out.bmp";
const struct option long_options[] = {
          {"help",no_argument,NULL,'h'},
          {"info",no_argument,NULL,'i'},
          {"output",required_argument,NULL,'o'},
               {"color",no_argument,NULL,'c'},
          {"line",no_argument,NULL,'l'},
          {"square_circle",no_argument,NULL,'s'},
          {"rad_circle",no_argument,NULL,'r'},
          {"paint_circle",no_argument,NULL,'p'},
          {NULL,0,NULL,0}
};
typedef struct{
    uint16_t signature;
    uint32_t filesize;
    uint16_t reserved1;
    uint16_t reserved2;
    uint32_t pixelArrOffset;
} BitmapFileHeader;

typedef struct{
    uint32_t headerSize;
```

```
    uint32_t width;
    uint32_t height;
    uint16_t planes;
    uint16_t bitsPerPixel;
    uint32_t compression;
    uint32_t imageSize;
    uint32_t xPixelsPerMeter;
    uint32_t yPixelsPerMeter;
    uint32_t colorsInColorTable;
    uint32_t importantColorCount;
} BitmapInfoHeader;

typedef struct{
    uint8_t b;
    uint8_t g;
    uint8_t r;
}Rgb;

#pragma pack(pop)

Rgb** arr;
unsigned int H;
unsigned int W;
int point_w =-1;
int point_h =-1 ;
int rad =-1;
int thick;
int Ox;
int Oy;
int M;
int N;
int Count = 2;

int chekRGB(int R, int G, int B){
    int flag = 0;
```

```
    if ((R<0) || (R>255)){
        flag = 1;
    }
    if ((G<0) || (G>255)){
        flag = 1;
    }
    if ((B<0) || (B>255)){
        flag = 1;
    }
    return flag;
}

void printFileHeader(BitmapFileHeader header){
    printf("BitmapFileHeader:\n");
    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("BitmapInfoHeader:\n");
    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);
}

void PrintHelp(){

printf("..................................................
.................\n");
    printf("This CLI-enabled program handles images in BMP
format,\n");
    printf("uncompressed and 24-bit color.\n");

printf("..................................................
.................\n");
    printf("Information about possible commands:\t -h,  --help
\n");
    printf("\n");
    printf("Image information:\t -i,  --info \n");
    printf("\n");
    printf("Entering the file name to display the result:\t -o,
--output \n");
    printf("( -o filename.bmp) \n");
    printf("Default: out.bmp\n");
```

```
    printf("\n");
    printf("Filter RGB-components:\t -c,  --color \n");
    printf("Input requirements:\t replacement color,
parameter.\n");
    printf("( -c [r/g/b] [0<k<255] ) \n");
    printf("\n");
    printf("Divide the image into NxM parts:\t -l,  --line\n");
    printf("Input requirements:\t number of horizontal and
vertical axes,\n");
    printf("line thickness, color (for each parameter).\n");
    printf("( -l [Ox] [Oy] [thick] [red] [green] [blue] )\n");
    printf("\n");
    printf("Draw a circle inscribed in a square:\t -s,  --
square_circle \n");
    printf("Input requirements:\t Coordinates of the upper left
(along the x and y axes) \n");
    printf("and lower right (along the x and y axes) corner of
the squarer,\n");
    printf("line thickness, color (for each parameter).\n");
    printf("( -s [LVw] [LVh] [PNw] [PNh] [thick] [red] [green]
[blue] ) \n");
    printf("\n");
    printf("Draw circle in center and radius:\t -r,  --rad_circle
\n");
    printf("Input requirements:\t Center coordinates (along the x
and y axes) \n");
    printf("radius, line thickness, color (for each
parameter).\n");
    printf("( -r [point_w] [point_h] [rad] [thick] [red] [green]
[blue] ) \n");
    printf("\n");
    printf("Paint over the circle (if available):\t -p,  --
paint_circle \n");
    printf("Input requirements:\t color options.\n");
    printf("( -p [thick] [red] [green] [blue] ) \n");
```

```
}

void DrawCircle(int R, int G, int B){
    for(int i=0; i<H; i++){
      for(int j=0; j<W; j++){
           if((j - point_w) * (j - point_w) + (i - point_h) * (i
- point_h) - rad * rad > -2 * thick * rad && (j - point_w) * (j -
point_w) + (i - point_h) * (i - point_h) - rad * rad < 0 ) {
               arr[i][j].r=R;
               arr[i][j].g=G;
               arr[i][j].b=B;
           }
      }
    }
}

void PaintCircle(int R,  int G, int B){
    for(int i=0; i<H; i++){
     for(int j=0; j<W; j++){
           if((j - point_w) * (j - point_w) + (i - point_h) * (i
- point_h) - rad * rad <= -2 * thick * rad && (j - point_w) * (j
- point_w) + (i - point_h) * (i - point_h) - rad * rad < 0 ) {
               arr[i][j].r=R;
               arr[i][j].g =G;
               arr[i][j].b=B;

      }

     }
    }
}

void ChangeColor(char c, int k){
    for(int i=0; i<H; i++){
      for(int j=0; j<W; j++){
```

```
            if (c == 'r'){
                arr[i][j].r=k;
            }
            if (c == 'g'){
                arr[i][j].g=k;
            }
            if (c == 'b'){
               arr[i][j].b=k;
            }
        }
      }
}

void DrawLines(int R, int G, int B){
   for(int i=0; i<H; i++){
     for(int j=0; j<W; j++){
            if(j == N) {
                for (int l = j - (thick / 2); l < j + (thick / 2) +
1; l++){
                  arr[i][l].r=R;
                  arr[i][l].g=G;
                  arr[i][l].b=B;
                }
                N=N+W/Ox;
            }
     }
       N=W/Ox;
   }

   for(int j=0; j<W; j++){
     for(int i=0; i<H; i++){
     if(i== H - thick)
     {
     break;
     }
```

```
            if(i == M) {
                for (int l = i - (thick / 2) - 1; l < i + (thick /
2); l++){
                  arr[l][j].r=R;
                  arr[l][j].g=G;
                  arr[l][j].b=B;

                }
                M=M+H/Oy;
            }

     }
     M=H/Oy;
   }
}


int main(int argc, char ** argv){
        if(argc == 1){
            PrintHelp();
            printf("Please try again\n"); //Ничего нет;
            return 0;
        }
    char * inputFile;
    int opt = 0;//какой аргумент сейчас просматриваем

    int inputFileNameLength = strlen(argv[argc-1]);
    inputFile = malloc(inputFileNameLength * sizeof(char));
    strcpy(inputFile, argv[argc-1]);
    FILE *f = fopen(inputFile, "rb"); //Открывает двоичный файл
для чтения
        if (f == NULL){
            printf("Please add a file.\n"); //Нет файла для
обработки;
            return 0;
```

```
        }
    BitmapFileHeader bmfh;
    BitmapInfoHeader bmif;
    fread(&bmfh,1,sizeof(BitmapFileHeader),f);
    fread(&bmif,1,sizeof(BitmapInfoHeader),f);
        if(bmfh.signature!=0x4d42){ //Сигнатура bmp-изображения;
             printf("Unsupported image format.\n"); //Изображение
не bmp;
                  printf("Try again!\n");
                 return 0;
        }
        if(bmif.bitsPerPixel!=24){
                printf("The color depth of this image is not
supported.\n"); //Изображение глубиной НЕ 24 бита на цвет;
                 printf("Requires 24 bits per color.\n");
            printf("Try again!\n");
                  return 0;
           }
        if(bmif.colorsInColorTable!=0){
                printf("Please use an uncompressed image.\n");
//Таблица цветов;
            printf("The color table of this image is not
supported.\n");
            printf("Try again!\n");
                  return 0;
           }

        if(bmif.compression!=0){
                printf("Please use an uncompressed image.\n");
//Изображение сжато;
                 return 0;
           }

    H = bmif.height;
    W = bmif.width;
```

```
    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);
    }
    int option_index;
    opt = getopt_long(argc, argv, optString, long_options,
&option_index ); //Получаем новый аргумент
        if (opt == -1){
            printf("Please add the command.\n"); //Нет команды
            return 0;
        }
    while(opt != -1) {
     switch (opt){
            case 'h':{
          Count = Count+1;
               if(Count>argc){
                    printf("The arguments do not match the
command.\n");//Данных не хватает для обработки;
                         return 0;
               }

                PrintHelp();
                break;
           }
         case 'o':{
          Count = Count+2;
               if(Count>argc){
                    printf("The arguments do not match the
command.\n");//Данных не хватает для обработки;
                         return 0;
               }
          outputFile = optarg;
          break;
         }
```

```
          case 'i':{
           Count = Count+1;
                if(Count>argc){
                     printf("The arguments do not match the
command.\n");//Данных не хватает для обработки;
                          return 0;
                }
           printFileHeader(bmfh);
           printInfoHeader(bmif);
           break;
               }

             case 'c':{
                Count = Count+3;
                if(Count>argc){
                     printf("The arguments do not match the
command.\n");//Данных не хватает для обработки;
                          return 0;
                }
                 char colorName;
                 int Value;
                 int a=0;
                 if (option_index!=0){
                     a=1;
                 }
                 for (int k = -1+a; k<1+a; ++k) {
                     if (k == -1 + a) {
                         colorName = *argv[optind + k];
                     }
                     if (k != -1 + a) {
                         Value = atoi(argv[optind + k]);
                     }
                 }
                     if ((colorName != 'r') && (colorName != 'g')
&& (colorName != 'b')){
```

```
                        printf("Please enter the color
again.\n");//Цвет не принадлижит RGB;
                        return 0;
                    }
                    if ((Value<0) || (Value>255)){
                        printf("Color parameters can take values
from 0 to 255.\n");//Параметр цвета вне границ;
                        printf("Try again!\n");
                        return 0;
                    }
                ChangeColor(colorName, Value);
                break;
            }
            case 'l':{
               Count = Count+7;
               if(Count>argc){
                    printf("The arguments do not match the
command.\n");//Данных не хватает для обработки;
                         return 0;
               }
                int inletL[6];
                int a=0;
                if (option_index!=0){
                    a=1;
                }
                for (int k = -1+a; k<5+a; ++k){
                    inletL[k + 1 - a] = atoi(argv[optind + k]);
                }
                Ox = inletL[0];
                Oy = inletL[1];
                thick = inletL[2];
                int R=inletL[3];
                int G=inletL[4];
                int B=inletL[5];
                    if (chekRGB(R, G, B)==1){
```

```
                        printf("Color parameters can take values
from 0 to 255.\n");//Параметр цвета вне границ;
                        printf("Try again!\n");
                        return 0;
                    }
                M= H / Oy;
                N= W / Ox ;
                DrawLines(R, G, B);
                break;
            }case 'r':{
               Count = Count+8;
               if(Count>argc){
                    printf("The arguments do not match the
command.\n");//Цвет не принадлижит RGB;
                         return 0;
               }
                int inletR[7];
                int a=0;
                if (option_index!=0){
                    a=1;
                }
                for (int k = -1+a; k<6+a; ++k){
                    inletR[k + 1 - a] = atoi(argv[optind + k]);
                }
                point_w=inletR[0];
                point_h=inletR[1];
                rad=inletR[2];
                thick=inletR[3];
                int R=inletR[4];
                int G=inletR[5];
                int B=inletR[6];
                if ((point_w>=W) || (point_h>=H)){
                    printf("The specified coordinates are outside
the image.\n"); //Координаты вне изображения
                    printf("Try again!\n");
```

```
                    return 0;
                }
                if ((rad>=H) || ((rad>=W))){
                    printf("Radius exceeds image
dimensions.\n");//Слишком большой радиус
                    printf("Try again!\n");
                    return 0;
                }
                if (chekRGB(R, G, B)==1){
                    printf("Color parameters can take values from
0 to 255.\n");//Параметр цвета вне границ;
                    printf("Try again!\n");
                    return 0;
                }
                DrawCircle(R, G, B);
                break;
            }
            case 's':{
               Count = Count+9;
               if(Count>argc){
                    printf("The arguments do not match the
command.\n");//Данных не хватает для обработки;
                         return 0;
               }
                int inletS[8];
                int a=0;
                if (option_index!=0){
                    a=1;
                }
                for (int k = -1+a; k<8+a; ++k){
                    inletS[k + 1 - a] = atoi(argv[optind + k]);
                }
                int LVw = inletS[0];
                int LVh = inletS[1];
                int PNw = inletS[2];
```

```
                int PNh = inletS[3];
                if ((LVw>=W) || (LVh>=H)){ //Верняя левая
координатка
                    printf("The specified coordinates are outside
the image.\n"); //Координаты вне изображения
                    printf("Try again!\n");
                    return 0;
                }
                if ((PNw>=W) || (PNh>=H)){ //Нижняя левая
координатка
                    printf("The specified coordinates are outside
the image.\n"); //Координаты вне изображения
                    printf("Try again!\n");
                    return 0;
                }
                if ((PNw - LVw)!=(PNh - LVh)){
                    printf("The entered data does not match the
square.\n"); //Введён не квадрат
                    printf("Try again!\n");
                    return 0;
                }

                point_w= (PNw + LVw) / 2 ;
                point_h= (PNh + LVh) / 2;
                rad=(PNh-LVh)/2;
                if ((rad>=H) || ((rad>=W))) {
                    printf("Radius exceeds image
dimensions.\n");//Слишком большой радиус
                    printf("Try again!\n");
                    return 0;
                }
                thick=inletS[4];
                int R=inletS[5];
                int G=inletS[6];
                int B=inletS[7];
```

```
                if (chekRGB(R, G, B)==1){
                    printf("Color parameters can take values from
0 to 255.\n");//Параметр цвета вне границ;
                    printf("Try again!\n");
                    return 0;
                }
                DrawCircle(R, G, B);
                break;
            }
            case 'p': {
               Count = Count+4;
               if(Count>argc){
                    printf("The arguments do not match the
command.\n");//Данных не хватает для обработки;
                         return 0;
               }
                if((point_w == -1) || (point_h == -1) || (rad==-
1) ){
                    printf("This option is available only if
there is a drawn circle.\n");//Круга нет;
                    printf("Try again!\n");
                    return 0;
                }
                int inletP[3];
                int a = 0;
                if (option_index != 0) {
                    a = 1;
                }
                for (int k = -1 + a; k < 2 + a; ++k) {
                    inletP[k + 1 - a] = atoi(argv[optind + k]);
                }
                int R = inletP[0];
                int G = inletP[1];
                int B = inletP[2];
                if (chekRGB(R, G, B)==1){
```

```
                    printf("Color parameters can take values from
0 to 255.\n");//Параметр цвета вне границ;
                    printf("Try again!\n");
                    return 0;
                }
                PaintCircle(R, G, B);
                break;
            }
         default:
                {
               break;
                }

        }
        option_index=0;
        opt = getopt_long(argc, argv, optString,
long_options,&option_index );
    }

    FILE *ff = fopen(outputFile, "wb");
    bmif.height = H;
    bmif.width = W;
    fwrite(&bmfh, 1, sizeof(BitmapFileHeader),ff);
    fwrite(&bmif, 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);
    free(arr);
    printf("\n");
    return 0;
}
```