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

ОТЧЕТ
КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка изображений

Студент гр. 0382 _____ Павлов С.Р
Шевская Н.В

Преподаватель _____ Чайка К.в

Санкт-Петербург
2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Стундент Павлов С.Р.

Группа 0382

Тема работы: Обработка изображений в формате BMP

Исходные данные:

Вариант 6

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 отрезка. Отрезок определяется:
  - координатами начала
  - координатами конца
  - цветом
  - толщиной
- (2) Инвертировать цвета в заданной окружности. Окружность определяется
  - **либо** координатами левого верхнего и правого нижнего угла квадрата, в который она вписана, **либо** координатами ее центра и радиусом
- (3) Обрезка изображения. Требуется обрезать изображение по заданной области. Область определяется:
  - Координатами левого верхнего угла
  - Координатами правого нижнего угла

Дата выдачи задания: 25.04.2021

Дата сдачи реферата 21.05.2021

Дата защиты реферата 21.05.2021

Студент	_____	Павлов С.Р Шевская Н.В
Преподаватель	_____	Чайка К.в

СОДЕРЖАНИЕ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

1.1 Цель работы — создать приложение на языке Си с CLI для обработки изображений формата BMP.

2.2 Для выполения необходимо решить следующие задачи:

- Создание пользовательского интерфейса, путем обработке команд и флагов командной строки_____
- Создание структур для работы с файлом BMP
- Чтение и запись фотографии
- Обработка ошибок

2. ХОД ВЫПОЛЕНИЯ РАБОТЫ

2.1 Подключение библиотек и объявление структур

Для работы программы используются такие библиотеки: *stdio.h* — для организации ввода и вывода, *stdlib.h* — для *malloc()* и других функций, *string.h* — для удобной обработки аргументов пользователя, *stdint.h* — для граммотного менджмента выделяемой памяти на элементы структуры, и *getopt.h* — для создание linux-подобного CLI интерфейса.

Затем производится объявление структур *BitmapFileHeader*, *BitmapInfoHeader*, и *Rgb.* Так же для корректного расположение структур в памяти и для записи, используется директива препроцессора *pragma*.

2.2 Организация CLI

В первую очередь объявляется структура *Configs,* в которую будут сохранены параметры,флаги, и значения для работы программы. Затем реализуется функция *void print_help(),* поясняющая возможный функционал и примеры использования программы и работу с командным интерфейсом.

Затем с помощь функции *getopt_long()* и цикла, производиться обработка входного командного потока, и значения аргументов и указаные флаги пользователем записываются в конфиг для дальнешего исполнения.

Так же, если есть записывается имя входного и выходного файла.

2.2 Первичная обработка и чтение файла

Следующий шаг после обработки команд программой — работа с фотографией. С помощью функции *int is_bmp_file(char* f)* - производится три проверки на существование,синтаксис и тип файла. (соответствие BMP формату). Затем файл побайтово читается с помощь функц. *fread()* , и сохраняется его bitmapfileheader и bitmapinfoheader.

Затем с динамическим выделением памяти считывается двумерный массив пикселей имеющий структуру RGB;

2.3 Решение подзадачи 1 «--info»

После успешного чтения и записи фотографии в память, если пользователем был указан флаг -i или —info , вызываются функции printFileHeader() и printInfoHeader(), которые печатают всю полную информацию о фотографии.

2.4 Решение подзадачи 2 «--section»

Для удобства создается структкура с конфигурацией выбраных пользователем параметров для конкретно этой подзадачи, далее в нее записываются все соответстующие параметры. Затем функция *make_inverse() ,* выполняет поставленную задачу, путем изменния массива пикселей. А конкретно, в функции расмотрены 4 частных случая поведения прямой «отрезка», и с помощью двух алгоритмов Брезенхема (для прямой и окружности), рисуется отрезок с задаными ему координатами, толщиной и цветом.

2.5 Решение подзадачи 3 «--inverse»

Так же считываются и обрабатываются параметры введеные пользователем. Затем на основе алгоритма для окружности и равенства окружности из прошлого пункта, двойным циклом инвертируются цвета в заданой кругом области.

2.6 Решение подзадачи 4 «--cut»

После считывания данных. Затем программа обрезает изображение.

2.7 Решение подзадачи 5 — сохранение файла.

По умолчанию файл сохраняется с таким же названием, с которым и поступил. Но пользователь может указать флаг -o или —out и тогда файл запишется с указаным именем. Сначала записывается bmfh и bmih, а затем массив пикселей.

Разработанный программный код смотрите в приложении А.

3. ТЕСТИРОВАНИЕ

1. --help / -h

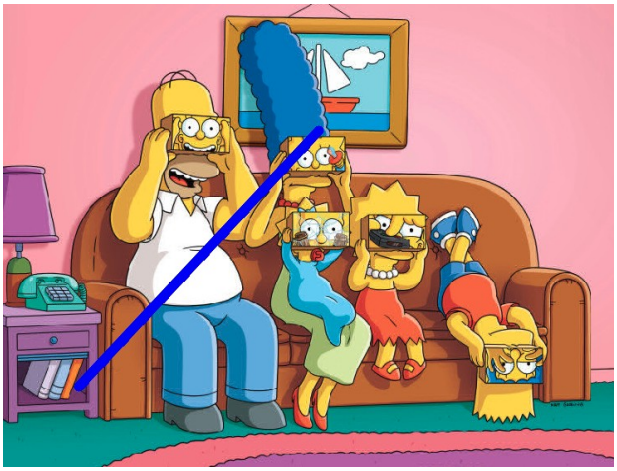
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

→  cw gcc main.c -lm && ./a.out -h
Используйте: myphoto [опции] файл...
        -h --help               - справочная информация (Вы читаете это)
        -o --out (название.bmp)         - Устанавливает и сохраняет фотографию под выбраный названием
        -i --info               - Печать полной информации о bmp файле

        -s --section (x1,y1|x2,y2|толщина|цвет)         - Рисует отрезок по параметрам
                ---> Пример: myphoto --section 2,2,6,6,3,red photo.bmp
                        Где цвет имеет тип: [white,black,brown,red,orange,yellow,green,blue,purple,grey]

        -I --inverse (radius x1,y1,R)   - Инвертировать цвета в заданной окружности
                ---> Пример: myphoto --inverse 50,50,25 photo.bmp

        -c --cut (x1;y1|x2;y2)          - Обрезка изображения
                ---> Пример: myphoto --cut 2,2,3,3 photo.bmp
→  cw
```

2. --info

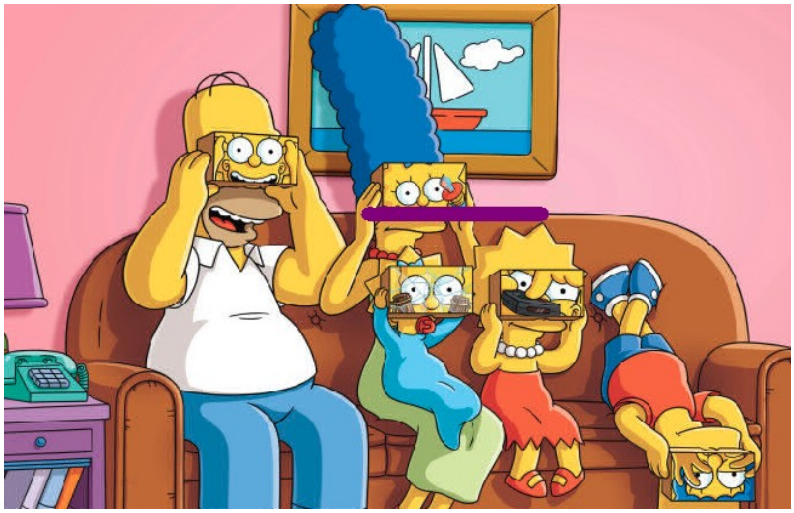
```
→  cw gcc main.c -lm && ./a.out --info simpsonsvr.bmp
Signature:      4d42 [19778]
Filesize:       141a62 [1317474]
Reserved1:      0 [0]
Reserved2:      0 [0]
pixelArrOffSet:         36 [54]
headerSize:     28 (40)
width:          30c (780)
height:         233 (563)
planes:         1 (1)
bitsPerPixel:   18 (24)
compression:    0 (0)
imageSize:      0 (0)
xPixelsPerMeter:        2e23 (11811)
yPixelsPerMeter:        2e23 (11811)
colorsInColorTable:     0 (0)
importantColorCount:    0 (0)

→  cw
```

3. Построение отрезка и вывод файла

```
→  cw gcc main.c -lm && ./a.out --section 100,100,400,400,3,blue simpsonsvr.bmp --out nice.bmp && eog nice.bmp
```

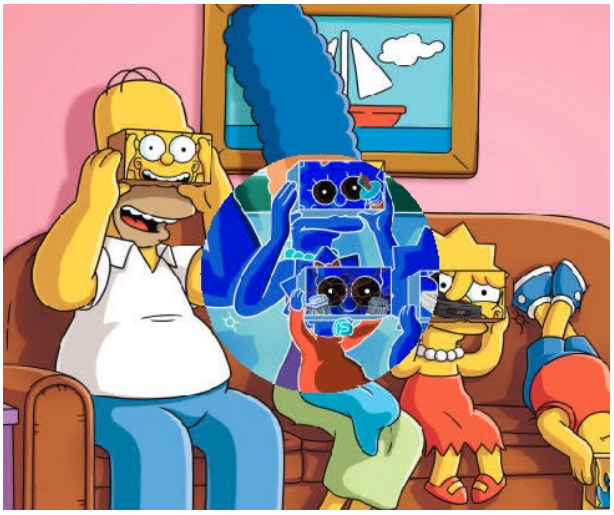


```
cw gcc main.c -lm && ./a.out --section 350,350,350,500,3,purple simpsonsvr.bmp --out nice.bmp && eog nice.bmp
```



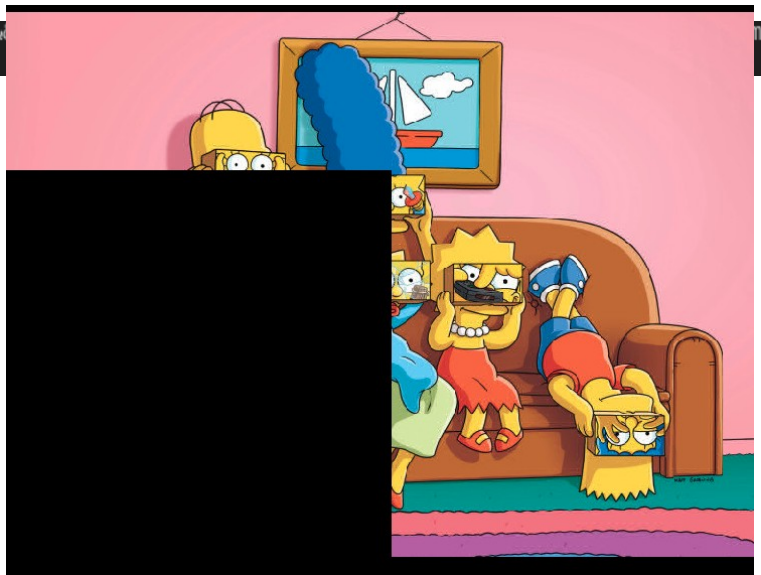
4. Инверсия цвета

```
➜  cw gcc main.c -lm && ./a.out --inverse 380,300,100 simpsonsvr.bmp --out nice.bmp && eog nice.bmp
```



5. Обрезка фото

```
cw gcc main.c -lm &&                                                    mp && eog nice.bmp
```



6. Микс функций

```
cw gcc main.c -lm && ./a.out --section 100,100,100,300,4,red --inverse 300,300,150 --cut 300,300,500,550 simpsonsvr.bmp --out nice.bmp && eog nice.bmp
```

ЗАКЛЮЧЕНИЕ

В ходе выполения данного задания была создана программа для выполениня простейших действий над BMP файлом. Был реализован ввод параметров необходимых для работы через функцию *getopt_long()*.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

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

#define MIN(a,b) (((a)<(b))?(a):(b))
#define MAX(a,b) (((a)>(b))?(a):(b))
#define roundf(x) floor(x + 0.5f)

#pragma pack(push, 1)

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

#pragma pack(pop)

void printFileHeader(BitmapFileHeader header){
```

```
printf("Signature: \t%x [%u]\n", header.signature,
header.signature);
printf("Filesize: \t%x [%u]\n", header.filesize, header.filesize);
printf("Reserved1: \t%x [%u]\n", header.reserved1,
header.reserved1);
printf("Reserved2: \t%x [%u]\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);
}

// CLI;

void print_help(){

printf("Используйте: myphoto [опции] файл...\n");
printf("\t-h --help\t\t- справочная информация (Вы читаете это)\
n");
printf("\t-o --out (название.bmp)\t\t- Устанавливает и сохраняет
фотографию под выбраный названием\n");
printf("\t-i --info \t\t- Печать полной информации о bmp файле\n\
n");
printf("\t-s --section (x1,y1|x2,y2|толщина|цвет)\t\t- Рисует
отрезок по параметрам\n");
printf("\t\t ---> Пример: myphoto --section 2,2,6,6,3,red
photo.bmp\n");
printf("\t\t \tГде цвет имеет тип:
[white,black,brown,red,orange,yellow,green,blue,purple,grey]\n\
n");
printf("\t-I --inverse (radius x1,y1,R)\t- Инвертировать цвета в
заданной окружности\n");
```

```
printf("\t\t ---> Пример: myphoto --inverse 50,50,25 photo.bmp\n\
n");
printf("\t-c --cut (x1;y1|x2;y2)\t\t- Обрезка изображения\n");
printf("\t\t ---> Пример: myphoto --cut 2,2,3,3 photo.bmp\n");
}
struct Configs{
char* input_file;
char* out_file;
_Bool info;
char* section;
char* inverse;
char* cut;
};

// CHECK INPUT FILE;
int is_bmp_file(char* f){
//flags;

char name_flag = 0;
char exist_flag = 0;
char meta_flag = 0;

// name review:
if ((f[strlen(f)-4] == '.')&&(f[strlen(f)-3] ==
'b')&&(f[strlen(f)-2] == 'm')&&(f[strlen(f)-1] == 'p')){
name_flag += 1;
}
else {
printf("Ошибка: Неверный формат файла [только .bmp]\n");
return 0;
}

// exist_flag:

FILE* tmp = fopen(f,"rb");
if (tmp != NULL){
exist_flag += 1;
}
else {
printf("Ошибка: указанного файла не существует!\n");
return 0;
}
// meta_flag:

unsigned short sign;

fread(&sign,1,2,tmp);

if (sign == 19778) {
meta_flag += 1;
}
else {
printf("Ошибка: фото не является не поддерживает .bmp формат\n");
```

```
return 0;
}

// answer
if ((name_flag == 1)&&(exist_flag == 1)&&(meta_flag == 1)) {
return 1;
}
else {
return 0;
}

}

// section funcs

struct section_params{
unsigned int x1;
unsigned int y1;
unsigned int x2;
unsigned int y2;
unsigned int thick;
char color[20];
};

struct section_params preprocess_section(char* str){
struct section_params conf;
char* tmp;
sscanf(str,"%d,%d,%d,%d,%s",
&conf.x1,&conf.y1,&conf.x2,&conf.y2,tmp);
sscanf(tmp,"%d,%s",&conf.thick,conf.color);
free(tmp);
return conf;
}

Rgb change_color(char* str){

Rgb ans;

if (strcmp(str,"blue") == 0){
ans.b = 255;
ans.g = 0;
ans.r = 0;
}

else if (strcmp(str,"black") == 0){
ans.b = 0;
ans.g = 0;
ans.r = 0;
}
else if (strcmp(str,"white") == 0){
ans.b = 255;
ans.g = 255;
ans.r = 255;
```

```
}
else if (strcmp(str,"purple") == 0){
ans.b = 128;
ans.g = 0;
ans.r = 128;
}
else if (strcmp(str,"purple") == 0){
ans.b = 128;
ans.g = 0;
ans.r = 128;
}
else if (strcmp(str,"red") == 0){
ans.b = 0;
ans.g = 0;
ans.r = 255;
}
else if (strcmp(str,"yellow") == 0){
ans.b = 0;
ans.g = 255;
ans.r = 255;
}
else if (strcmp(str,"green") == 0){
ans.b = 0;
ans.g = 255;
ans.r = 0;
}
else if (strcmp(str,"orange") == 0){
ans.b = 0;
ans.g = 165;
ans.r = 255;
}
else if (strcmp(str,"brown") == 0){
ans.b = 19;
ans.g = 69;
ans.r = 139;
}
return ans;
}

void make_circle(int H, int W,int x0,int y0,Rgb **arr, int radius,
Rgb color){
int x = 0;
int y = radius;
int delta = 1 - 2 * radius;
int error = 0;
while(y >= 0) {
arr[y0 + y][x0 + x] = color;
arr[y0 - y][x0 + x] = color;
arr[y0 + y][x0 - x] = color;
arr[y0 - y][x0 - x] = color;
error = 2 * (delta + y) - 1;
if(delta < 0 && error <= 0) {
++x;
```

```
delta += 2 * x + 1;
continue;
}
error = 2 * (delta - x) - 1;
if(delta > 0 && error > 0) {
--y;
delta += 1 - 2 * y;
continue;
}
++x;
delta += 2 * (x - y);
--y;
}
for(int i = 0; i < H; i++){
for(int j = 0; j < W; j++)
if((j-x0)*(j-x0)+(i-x0)*(i-x0)>=radius*radius &&
(j-x0)*(j-x0)+(i-y0)*(i-y0)<=(radius+3)*(radius+3)) arr[i][j] =
color;
}
for(int i = 0; i < H; i++)
for(int j = 0; j < W; j++)
if((j-x0)*(j-x0)+(i-y0)*(i-y0)<=radius*radius) arr[i][j] = color;
}

void swap(char *a, char *b){
char t = *a;
*a = *b;
*b = t;
}

void make_section(unsigned int H,unsigned int W,Rgb **arr, struct
section_params conf){

Rgb color = change_color(conf.color);
// размер;
int dx = abs(conf.x2 - conf.x1);
int dy = abs(conf.y2 - conf.y1);
int len = MAX(dx,dy);

if (dy == 0){
for(int i=conf.x1;i<conf.x2;i++){
//arr[conf.y1][i] = color;

make_circle(H,W,conf.y1,i, arr,conf.thick,color);
}
}
else if (dx == 0){
for(int i=conf.y1;i<conf.y2;i++){
//arr[i][conf.x1] = color;
make_circle(H,W,i,conf.x1, arr,conf.thick,color);
}
}
else if (dx >= dy){
```

```
int x = conf.x1;
int y = conf.y1;
float koef = (float) dy/dx;

while(x<=conf.x2+1){
//arr[(int) roundf(y)][x] = color;
make_circle(H,W,roundf(y),x, arr,conf.thick,color);
//printf("X: %d Y: %d X*koef: %f\n", x, y, x*koef);
x++;
y = x*koef;
}
}
else if(dy > dx){
int x = conf.x1;
int y = conf.y1;
float koef = (float) dy/dx;
while(y<=conf.y2+1){
//arr[y][(int) roundf(x)] = color;
make_circle(H,W,y,roundf(x), arr,conf.thick,color);
//printf("X: %d Y: %d \n", x, y);
y++;
x = y/koef;
}
}

}

// cut funcs

struct cut_params{
int x1;
int y1;
int x2;
int y2;
};

void make_cut(unsigned int H, unsigned int W, Rgb **arr, struct
cut_params conf){
for (int i=conf.x1;i<conf.x2;i++){
for (int j=conf.y1;j<conf.y2;j++){
arr[i][j].r = 0;
arr[i][j].g = 0;
arr[i][j].b = 0;
}
}
}

// inverse funcs;

struct inverse_params{
int x1;
int y1;
int radius;
```

```
};

void make_inverse(int H, int W, Rgb **arr, struct inverse_params
conf){
int radius = conf.radius;
for(int i = 0; i < H; i++)
for(int j = 0; j < W; j++)
if((j-conf.x1)*(j-conf.x1)+(i-conf.y1)*(i-conf.y1)<=radius*radius)
{
arr[i][j].r = 255 - arr[i][j].r;
arr[i][j].g = 255 - arr[i][j].g;
arr[i][j].b = 255 - arr[i][j].b;
}
}

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

// CLI;

struct Configs param_config = {0,0,0,0,0,0};

static struct option long_options[] = {
{"help", 0,0,'h'},
{"info", 0,0,'i'},
{"out", 1,0,'o'},
{"section", 1,0,'s'},
{"inverse", 1,0,'I'},
{"cut", 1,0,'c'},
{0,0,0,0}
};

int opt;
int option_index = 0;
opterr = 0;
opt = getopt_long(argc, argv,"hio:s:I:c:", long_options,
&option_index);

while (opt!=-1){
switch (opt)
{
case 'h':
print_help();
return 0;
case '?':
printf("Ошибка: Неверный флаг \"-%c\"\n", optopt);
print_help();
return 0;
case 'o':
param_config.out_file = optarg;
break;
case 'i':
param_config.info = 1;
break;
```

```
case 's':
param_config.section = optarg;
break;
case 'I':
param_config.inverse = optarg;
break;
case 'c':
param_config.cut = optarg;
break;
}

opt = getopt_long(argc, argv,"hio:s:I:c:", long_options,
&option_index);
}

// check for file;

argc -= optind;
argv += optind;

if (is_bmp_file(argv[argc-1]) == 1){
param_config.input_file = argv[argc-1];
}
else{
printf("Фатальная Ошибка: Укажите название изменяемого файла!\n");
return 0;
}

/*
===== Start of programm; =====
*/

// [0] PREmap;

FILE *f = fopen(param_config.input_file, "rb");
BitmapFileHeader bmfh;
BitmapInfoHeader bmih;

fread(&bmfh, 1, sizeof(BitmapFileHeader), f);
fread(&bmih, 1, sizeof(BitmapInfoHeader), f);

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

// Image data:
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);
}

//[1] INFO
```

```
if (param_config.info == 1) {
printFileHeader(bmfh);
printInfoHeader(bmih);

}
//[2] Section

if (param_config.section != 0){

struct section_params sect_conf =
preprocess_section(param_config.section);
make_section(H, W , arr, sect_conf);
}

//[3] Inverse

if (param_config.inverse != 0){

struct inverse_params inv_conf;
sscanf(param_config.inverse,"%d,%d,%d",
&inv_conf.x1,&inv_conf.y1,&inv_conf.radius);
make_inverse(H,W,arr,inv_conf);
}

//[4] Cut

if (param_config.cut != 0){
struct cut_params lp;
sscanf(param_config.cut,"%d,%d,%d,%d",
&lp.x1,&lp.y1,&lp.y2,&lp.x2);
make_cut(H, W , arr, lp);
}

//[5 - FINAL] Out file
FILE* ff;

if (param_config.out_file == 0){
ff = fopen(param_config.input_file, "wb");
}
else {
ff = fopen(param_config.out_file, "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);
printf("\n");
return 0;
```