**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Обработка изображений на языке программирования Си

|  |  |  |
| --- | --- | --- |
| Студент гр. 3384 |  | Козьмин Н.В. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Козьмин Н.В. | | |
| Группа 3384 | | |
| Тема работы: Обработка изображений на языке программирования Си | | |
| Исходные данные:  Программа обязательно должна иметь CLI [1, 2]. Программа должна реализовывать весь следующий функционал по обработке bmp-файла.  Общие сведения   * 24 бита на цвет * без сжатия * файл может не соответствовать формату BMP, т.е. необходимо проверка на BMP формат (дополнительно стоит помнить, что версий у формата несколько). Если файл не соответствует формату BMP или его версии, то программа должна завершиться с соответствующей ошибкой. * 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 иметь следующую функции по обработке изображений:  (1) Рисование квадрата с диагоналями. Флаг для выполнения данной операции: `--squared\_lines`. Квадрат определяется:   * Координатами левого верхнего угла. Флаг `--left\_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y * Размером стороны. Флаг `--side\_size`. На вход принимает число больше 0 * Толщиной линий. Флаг `--thickness`. На вход принимает число больше 0 * Цветом линий. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет) * Может быть залит или нет (диагонали располагаются “поверх” заливки). Флаг `--fill`. Работает как бинарное значение: флага нет – false , флаг есть – true. * Цветом которым он залит, если пользователем выбран залитый. Флаг `--fill\_color` (работает аналогично флагу `--color`)   (2) Фильтр rgb-компонент. Флаг для выполнения данной операции: `--rgbfilter`. Этот инструмент должен позволять для всего изображения либо установить в диапазоне от 0 до 255 значение заданной компоненты. Функционал определяется   * Какую компоненту требуется изменить. Флаг `--component\_name`. Возможные значения `red`, `green` и `blue`. * В какой значение ее требуется изменить. Флаг `--component\_value`. Принимает значение в виде числа от 0 до 255   (3) Поворот изображения (части) на 90/180/270 градусов. Флаг для выполнения данной операции: `--rotate`. Функционал определяется   * Координатами левого верхнего угла области. Флаг `--left\_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y * Координатами правого нижнего угла области. Флаг `--right\_down`, значение задаётся в формате `right.down`, где right – координата по x, down – координата по y * Углом поворота. Флаг `--angle`, возможные значения: `90`, `180`, `270`   Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции. | | |
| Содержание пояснительной записки:  «Содержание», «Введение», «Обработка опций», «Файл с главной функцией», «Мейкфайл», «Заключение», «Список использованных источников» | | |
| Предполагаемый объем пояснительной записки:  Не менее 20 страниц. | | |
| Дата выдачи задания: 18.03.2024 | | |
| Дата сдачи реферата: 27.05.2024 | | |
| Дата защиты реферата: 31.05.2024 | | |
| Студент |  | Козьмин Н.В. |
| Преподаватель |  | Глазунов С.А. |

**Аннотация**

В проделанной работе описана программа, реализованная на языке Си, которая предназначена для работы с изображениями. Важным элементом программы является использование интерфейса командной строки (CLI), который позволяет грамотно считывать команды, нежели простая обработка стандартного потока ввода. Примечательно, что в написании кода использовался алгоритм Брезенхэма и сравнение реализованного функционала с тем, как он сделан в популярном Paint’e. Для сборки использовался Makefile. Тесты и их результаты см. в приложении Б. Разработанный программный код см. в приложении А. Скриншоты с успешными запусками прилагаются.

**Summary**

The work done describes a program implemented in the C language, which is designed to work with images. An important element of the program is the use of a command line interface (CLI), which allows you to read commands intelligently, rather than simply processing standard input. It is noteworthy that the Bresenham algorithm was used in writing the code and a comparison of the implemented functionality with how it was done in the popular Paint. Makefile was used for assembly. Tests and their results can be found in Appendix B. The developed program code can be found in Appendix A. Screenshots of successful launches are attached.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 7 |
| 1. | Обработка опций | 8 |
| 1.1. | Предварительная подготовка | 8 |
| 1.2. | Функция processopts. Считывание команд | 8 |
| 1.3 | Функция processopts. Обработка команд | 9 |
| 2. | Файл с главной функцией | 10 |
| 2.1. | Предварительная работа с bmp-файлами | 10 |
| 2.2. | Реализация функций из задания | 11 |
| 2.3 | Главная функция | 11 |
| 3. | Мейкфайл | 12 |
|  | Заключение | 13 |
|  | Список использованных источников | 14 |
|  | Приложение А. Исходный код программы | 15 |
|  | Приложение Б. Тестирование | 32 |

**введение**

Целью данной работы является разработка программы на языке Си, обрабатывающую изображения по определённым правилам. Для достижения поставленной цели были решены следующие задачи:

1. Записать и обработать команды [3, 4]
2. Реализовать по методическим материалам [5] ввод и вывод изображений вместе с выравниванием.
3. Сделать установку пикселя и рисование линий [6].
4. Создать функции напрямую нужные по условию задачи.
5. Отделить от основного файла файл с getopt. Сборка программы.
6. Реализовать выполнение команд в main’е.

**1. Обработка опций**

**1.1. Предварительная подготовка**

Включаем заголовочные файлы: getopt.h для непосредственной обработки команд, stdio.h для вывода сообщений и считывания информации из строк, stdlib.h для работы с динамической памятью и string.h для копирования и сравнения строк. Extern’ы для optarg и optind используем, чтобы показать, что они определены, но в другом месте (в getopt.h). Также создаём структуру Argf, которая будет хранить в себе данные, полученные из CLI.

Затем создаем функцию печати справки, в которую постепенно добавляем информацию. Также делаем функции проверки: is\_bmp(char \*s) – для проверки на то, является ли строка файлом bmp формата, is\_color(int i) – является ли число цветом из 8 бит.

**1.2. Функция processopts. Считывание команд**

Задаем функции processopts параметры, которые стандартно получает main и argf, указатель на Argf. Если из команд только сама команда запуска, то печатаем справку и выходим со значением завершения (1). Иначе задаём long\_options и short\_options для опций, флаги для них и начальные значения.

Затем в цикле проходим по командам, считывая при необходимости аргументы. Если команды закончились, выходим из цикла. Иначе проверяем через switch-case. Для большинства команд характерно наличие проверки на дублирование посредством флагов. Также используются флаги, которые характерны только для какой-то одной функции, чтобы упростить последующие проверки на ошибки. Из особенностей реализации обработки отдельных команд можно отметить следующие: input и output сохраняют значения в динамическую память и проверяют на bmp, используя функцию из подготовки. Проверка трёх цветов проходит также уже с помощью подготовленной функции. Команда со знаком вопроса означает неизвестную команду, соответственно возвращается ошибка.

**1.3. Функция processopts. Обработка команд**

Если есть справка и какая-нибудь другая команда – ошибка.

Если есть запрос инфо о файле и какая-нибудь другая команда, кроме input – ошибка.

Если есть первая функция и либо вторая, либо третья – ошибка.

Если есть вторая и третья – ошибка.

Затем проходим по аргументам, не являющимися опциями (или их аргументами), через цикл, в котором все действия не требуются в подробном объяснении. Просто заносим input и при необходимости также печатаем и возвращаем ошибки.

Если есть справка и нет пропущенных аргументов выводим аналогично случаю в начале справку. Если есть пропущенные аргументы – ошибка. Если нет ввода – ошибка, так как все остальные функции требуют input. Если ввод не является bmp-файлом (а это может быть, если ввод – свободный аргумент) – ошибка. Если выбрана функция пока информации о файле, заканчиваем обработку. Если не выбрана никакая функция – ошибка. Если нет вывода, задаем дефолтный. Если input и output идентичны – ошибка. Если нужно закрасить, но нет цвета закрашивания – ошибка. Если нет одного из компонентов для функции установки цвета – ошибка. Если нет угла поворота для соответствующей функции – ошибка.

**2. файл с главной функцией**

**2.1. Предварительная работа с bmp-файлами**

Импортируем все файлы, за исключением getopt.h, описанные ранее по понятным причинам. Объявляем структуры BitmapFileHeader и BitmapInfoHeader для хранения метаданных итруктуру Rgb для хранения информации о цветах отдельного пикселя. Оборачиваем их в pragma pack, чтобы задать выравнивание этих структур. Выбираем 1бит с аргументом push, так как в дальнейшим с этим будет проще работать.

Создаем функции печати информации из метаданных, это нам пригодится для печати информации о файле. Затем переходим к функции для чтения файла. Пытаемся открыть файл, если получили 0, значит файла нет, возвращаем ошибку. Иначе считываем метаданные. Берем размеры изображения. Выделяем память под пиксели и в строках до выделяем память для выравнивания картинок. Закрываем файл. Для считывания выполняем действия, наоборот.

Теперь создаем функцию установки пикселя. Выражение y = H-y-1 нужно для перевода начала отсчета координат из нижнего в верхний. Циклы же нужны, чтобы задать размеры закрашиваемых пикселей исходным. Также при каждой попытке закрасить пиксель проверяем его выход за границы изображения. Далее реализуем функцию fill, чтобы заполнять изображение не квадратами так как в set\_pixel, а прямоугольниками, также создаем функцию обмена точками, чтобы менять точки местами, если это необходимо, либо ниже другой точки и наконец создаем функцию рисования линии.

В draw\_line используем целочисленный алгоритм Брезенхэма и добавляем манипуляции с координатами, чтобы создать рисования во все стороны и на полуинтервале [начало, конец). Также можно заметить, что, если нужно, чтобы концы линий закруглялись, это можно сделать, модифицируя ранее описанные функции. Вдобавок можно отметить, что рисование линий относительно ТЗ усложнено, но оно добавляет расширяемость.

**2.2. Реализация функций из задания**

Функция рисования квадрата с диагоналями, используя функцию рисования линий очень проста. Указываем шесть отрезков. Готово.

В функции rgbfilter мы проходим в циклах по всему изображению. И в зависимости от выбранного цвета меняем нужный на выбранное значение.

Сложнее всего из описания этого блока – функция поворота. Сперва мы устанавливаем левую верхнюю координату, если это не так, и обрезаем область, если она выходит за пределы. Затем проходим разными способами по области, в зависимости от того, какой угол поворота нам нужен, так, чтобы брались те пиксели, которые при вставке будут идти первыми, кстати поэтому нам нужно дополнительное изображение, так как, если будет одно, пиксели перемешаются.

Функция поворота области сначала была реализована так, чтобы при любом раскладе пиксели начинали вставляться с левого верхнего угла. Это было сделано по той причине, что так проще. Уже далее была добавлена переменная shift для хранения вычисленного отступа (там, где это нужно) и после добавления во вставку пикселей shift’а, функция заработала, как надо.

**2.3. Главная функция**

Получаем стандартным образом аргументы (int argc, char \*\*argv) и объявляем структуру Argf argf, если processopts требует успешно завершить работу, выходим, иначе работаем с заполненным argf. Объявляем метаданные, считываем изображение в bmp\_file\_pixels. Если argf.function == 'b', значит получаем информацию о файле и выходим. Если ‘q’, значит закрашиваем при необходимости область и рисуем в ней квадрат с диагоналями. Если 'r', применяем фильтр rgb компонент. Если ‘t’, значит используем функцию поворота, для этого повторно считываем исходное изображение, так как это самые способ копирования того, что нам нужно. Имея все, что нужно, также вызываем angle. При любом из оставшихся вариантов записываем изображение и заканчиваем работу.

**3. Мейкфайл**

Мы делим исходный файл на два, примерно одинаковых по размеру файла: menu.c и processopts.c. По такому коду проще ориентироваться, а сборка происходит изолированно и примерно за одинаковое время, что доказано опытным путём. Так как структура Argf требуется в двух файлах, то мы заносим её объявление в заголовочный фал и импортируем её и там, и там, очевидно, также проделываем с объявлением processopts. Стоит отметить, что все повторные включения заголовочных файлов, которые уже теперь явно разделены по файлам, мы оборачиваем в проверку на дублирование.

**заключение**

В проделанной работе была разработана программа на языке Си, обрабатывающая изображения по всем требуемым правилам. Все поставленные задачи были выполнены, а также был проведён глубокий анализ написанного кода. Эти действия подвели итог программированию на языке Си во втором семестре и помогли закрепить полученные знания.

**список использованных источников**

1. Требования к CLI для курсовой работы // МОЭВМ Вики. URL: https://se.moevm.info/doku.php/courses:programming:pr\_cw\_spring\_requirements (дата обращения: 25.05.2024).
2. Требования к курсовым (весенний семестр) // МОЭВМ Вики. URL: https://se.moevm.info/doku.php/courses:programming:rules\_extra\_kurs (дата обращения: 25.05.2024).
3. Parsing program options using getopt // ОС GNU. URL: https://www.gnu.org/software/libc/manual/html\_node/Getopt.html (дата обращения: 25.05.2024).
4. Getopt // Opennet (Аналог команды man, используемой в Linux). URL: https://www.opennet.ru/man.shtml?topic=getopt&category=3&russian=0 (дата обращения: 25.05.2024)
5. Базовые сведения к выполнению курсовой работы по дисциплине «Программирование». Второй семестр: учеб.-метод. пособие / М. М. Заславский, А. А. Лисс, А. В. Гаврилов, и др. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2024. 36 с.
6. Алгоритм Брезенхэма // Википедия. URL: https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC\_%D0%91%D1%80%D0%B5%D0%B7%D0%B5%D0%BD%D1%85%D1%8D%D0%BC%D0%B0

**приложение А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

Название файла: processopts.h

#ifndef \_INC\_PROCESSOPTS

#define \_INC\_PROCESSOPTS

#endif

typedef struct Argf{

char \*input;

char \*output;

char function;

int left\_up\_x;

int left\_up\_y;

int side\_size;

int thickness;

int color\_r;

int color\_g;

int color\_b;

char is\_fill;

int fill\_color\_r;

int fill\_color\_g;

int fill\_color\_b;

char component\_name;

int component\_value;

int right\_down\_x;

int right\_down\_y;

int angle;

} Argf;

char processopts(int argc, char \*\*argv, Argf \*argf);

Название файла: processopts.c

#include <getopt.h>

#ifndef \_INC\_STDIO

#include <stdio.h>

#endif

#ifndef \_INC\_STDLIB

#include <stdlib.h>

#endif

#ifndef \_INC\_STRING

#include <string.h>

#endif

#ifndef \_INC\_PROCESSOPTS

#include "processopts.h"

#endif

extern char \*optarg;

extern int optind;

void printhelp(){

puts(

"Course work for option 4.12, created by Nikita Kozmin\n"

"--help -h Display this information\n"

"--input -i Input file name (can be omitted; default value is last argument)\n"

"--output -i Output file name (can be omitted; default value is \"out.bmp\")\n"

"--info Image information\n"

"--squared\_lines Drawing a square with diagonals\n"

" --left\_up Coordinates of the upper left corner (format: \"x.y\")\n"

" --side\_size Side size (number greater than zero)\n"

" --thicknes Thickness of lines (number greater than zero)\n"

" --color Line color (format: \"rrr.ggg.bbb\")\n"

" --fill Fill flag (no arguments)\n"

" --fill\_color Fill color (format: \"rrr.ggg.bbb\")\n"

"--rgbfilter Filter rgb-component\n"

" --component\_name Changeable component (possible values: \"red\", \"green\" and \"blue\")\n"

" --component\_value New color value (0 to 255)\n"

"--rotate Rotate the image (part) by 90/180/270 degrees\n"

" --left\_up Coordinates of the upper left corner (format: \"x.y\")\n"

" --right\_down Coordinates of the lower right corner (format: \"x.y\")\n"

" --angle Angle of rotation (possible values: \"90\", \"180\", \"270\")"

);

}

char is\_bmp(char \*s){

int len = strlen(s);

if (len<5){

return 0;

}

if (s[len-4] == '.'){

return 1;

}

return 0;

}

char is\_color(int c){

if (0 <= c && c <= 255){

return 1;

}

return 0;

}

char processopts(int argc, char \*\*argv, Argf \*argf){

if (argc == 1){

printhelp();

return 1;

}

int c;

static struct option long\_options[] = {

//name has\_arg flag val

{"help", 0, 0, 'h'},

{"input", 1, 0, 'i'},

{"output", 1, 0, 'o'},

{"info", 0, 0, 'b'},

{"squared\_lines", 0, 0, 'q'},

{"left\_up", 1, 0, 'u'},

{"side\_size", 1, 0, 's'},

{"thickness", 1, 0, 'k'},

{"color", 1, 0, 'c'},

{"fill", 0, 0, 'f'},

{"fill\_color", 1, 0, 'l'},

{"rgbfilter", 0, 0, 'r'},

{"component\_name", 1, 0, 'n'},

{"component\_value", 1, 0, 'v'},

{"rotate", 0, 0, 't'},

{"right\_down", 1, 0, 'd'},

{"angle", 1, 0, 'a'},

{0, 0, 0, 0 }

};

char short\_options[] = "hi:o:";

char has\_help = 0;

char has\_squared\_lines = 0;

char has\_only\_squared\_lines\_opts = 0;

char has\_input = 0;

char has\_output = 0;

char has\_left\_up = 0;

char has\_side = 0;

char has\_thickness = 0;

char has\_color = 0;

char has\_fill = 0;

char has\_fill\_color = 0;

char has\_rgbfilter = 0;

char has\_only\_rgbfilter\_opts = 0;

char has\_component\_name = 0;

char has\_component\_value = 0;

char has\_rotate = 0;

char has\_only\_rotate\_opts = 0;

char has\_right\_down = 0;

char has\_angle = 0;

char has\_info = 0;

argf->input = NULL;

argf->output = NULL;

argf->function = '\0';

argf->left\_up\_x = 0;

argf->left\_up\_y = 0;

argf->side\_size = 0;

argf->thickness = 1;

argf->color\_r = 0;

argf->color\_g = 0;

argf->color\_b = 0;

argf->is\_fill = 0;

argf->fill\_color\_r = 0;

argf->fill\_color\_g = 0;

argf->fill\_color\_b = 0;

argf->component\_name = '\0';

argf->component\_value = 0;

argf->right\_down\_x = 0;

argf->right\_down\_y = 0;

argf->angle = 0;

while (1)

{

int option\_index = 0;

c = getopt\_long(argc, argv, short\_options,

long\_options, &option\_index);

if (c == -1)

break;

switch (c){

case 'h':

{

// Проверка на дублирование

if (has\_help){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

has\_help = 1;

break;

}

case 'q':

{

// Проверка на дублирование

if (has\_squared\_lines){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

argf->function = 'q';

has\_squared\_lines = 1;

has\_only\_squared\_lines\_opts = 1;

break;

}

case 'i':

{

// Проверка на дублирование

if (has\_input){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

argf->input = malloc(sizeof(char)\*strlen(optarg));

strcpy(argf->input, optarg);

if (!is\_bmp(argf->input)){

fprintf(stderr, "Invalid argument (the file isn't bmp). Error 45\n");

exit(45);

}

has\_input = 1;

break;

}

case 'o':

{

// Проверка на дублирование

if (has\_output){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

argf->output = malloc(sizeof(char)\*strlen(optarg));

strcpy(argf->output, optarg);

if (!is\_bmp(argf->output)){

fprintf(stderr, "Invalid argument (the file isn't bmp). Error 45\n");

exit(45);

}

has\_output = 1;

break;

}

case 'u':

{

// Проверка на дублирование

if (has\_left\_up){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

if (sscanf(optarg, "%d.%d", &argf->left\_up\_x, &argf->left\_up\_y) != 2){

fprintf(stderr, "Invalid argument (no coordinates). Error 45\n");

exit(45);

}

has\_left\_up = 1;

break;

}

case 's':

{

// Проверка на дублирование

if (has\_side){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

if (sscanf(optarg, "%d", &argf->side\_size) != 1 || argf->side\_size <= 0){

fprintf(stderr, "Invalid argument (no side). Error 45\n");

exit(45);

}

has\_side = 1;

has\_only\_squared\_lines\_opts = 1;

break;

}

case 'k':

{

// Проверка на дублирование

if (has\_thickness){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

if (sscanf(optarg, "%d", &argf->thickness) != 1 || argf->thickness <= 0){

fprintf(stderr, "Invalid argument (no thickness). Error 45\n");

exit(45);

}

has\_thickness = 1;

has\_only\_squared\_lines\_opts = 1;

break;

}

case 'c':

{

// Проверка на дублирование

if (has\_color){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

if (sscanf(optarg, "%d.%d.%d", &argf->color\_r, &argf->color\_g, &argf->color\_b) != 3){

fprintf(stderr, "Invalid argument (no color). Error 45\n");

exit(45);

}

if (!is\_color(argf->color\_r) || !is\_color(argf->color\_g) || !is\_color(argf->color\_b)){

fprintf(stderr, "Invalid color. Error 41\n");

exit(41);

}

has\_color = 1;

has\_only\_squared\_lines\_opts = 1;

break;

}

case 'f':

{

// Проверка на дублирование

if (has\_fill){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

argf->is\_fill = 1;

has\_fill = 1;

has\_only\_squared\_lines\_opts = 1;

break;

}

case 'l':

{

// Проверка на дублирование

if (has\_fill\_color){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

if (sscanf(optarg, "%d.%d.%d", &argf->fill\_color\_r, &argf->fill\_color\_g, &argf->fill\_color\_b) != 3){

fprintf(stderr, "Invalid argument (no color). Error 45\n");

exit(45);

}

if (!is\_color(argf->fill\_color\_r) || !is\_color(argf->fill\_color\_g) || !is\_color(argf->fill\_color\_b)){

fprintf(stderr, "Invalid color. Error 41\n");

exit(41);

}

has\_fill\_color = 1;

has\_only\_squared\_lines\_opts = 1;

break;

}

case 'r':

{

// Проверка на дублирование

if (has\_rgbfilter){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

argf->function = 'r';

has\_rgbfilter = 1;

has\_only\_rgbfilter\_opts = 1;

break;

}

case 'n':

{

// Проверка на дублирование

if (has\_component\_name){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

if (strcmp("red", optarg) == 0){

argf->component\_name = 'r';

}

else if (strcmp("green", optarg) == 0){

argf->component\_name = 'g';

}

else if (strcmp("blue", optarg) == 0){

argf->component\_name = 'b';

}

else{

fprintf(stderr, "Invalid argument (no name color). Error 45\n");

exit(45);

}

has\_component\_name = 1;

has\_only\_rgbfilter\_opts = 1;

break;

}

case 'v':

{

// Проверка на дублирование

if (has\_component\_value){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

if (sscanf(optarg, "%d", &argf->component\_value) != 1){

fprintf(stderr, "Invalid argument (no color). Error 45\n");

exit(45);

}

if (!is\_color(argf->component\_value)){

fprintf(stderr, "Invalid color. Error 41\n");

exit(41);

}

has\_component\_value = 1;

has\_only\_rgbfilter\_opts = 1;

break;

}

case 't':

{

// Проверка на дублирование

if (has\_rotate){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

argf->function = 't';

has\_rotate = 1;

has\_only\_rotate\_opts = 1;

break;

}

case 'd':

{

// Проверка на дублирование

if (has\_right\_down){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

if (sscanf(optarg, "%d.%d", &argf->right\_down\_x, &argf->right\_down\_y) != 2){

fprintf(stderr, "Invalid argument (no coordinates). Error 45\n");

exit(45);

}

has\_right\_down = 1;

has\_only\_rotate\_opts = 1;

break;

}

case 'a':

{

// Проверка на дублирование

if (has\_angle){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

if (sscanf(optarg, "%d", &argf->angle) != 1){

fprintf(stderr, "Invalid argument (no angle). Error 45\n");

exit(45);

}

if ((argf->angle != 90) && (argf->angle != 180) && (argf->angle != 270)){

fprintf(stderr, "Invalid argument (no angle). Error 45\n");

exit(45);

}

has\_angle = 1;

has\_only\_rotate\_opts = 1;

break;

}

case 'b':

{

// Проверка на дублирование

if (has\_info){

fprintf(stderr, "Invalid flag (duplicate). Error 46\n");

exit(46);

}

argf->function = 'b';

has\_info = 1;

break;

}

case '?':

{

fprintf(stderr, "Invalid or missing flag or argument. Error 46 or 45\n");

exit(46);

}

}

}

if (has\_help && (has\_only\_squared\_lines\_opts || has\_only\_rgbfilter\_opts || has\_only\_rotate\_opts || has\_input || has\_output || has\_info || has\_left\_up)){

fprintf(stderr, "Invalid flag. Error 46\n");

exit(46);

}

if (has\_info && (has\_only\_squared\_lines\_opts || has\_only\_rgbfilter\_opts || has\_only\_rotate\_opts || has\_output || has\_left\_up)){

fprintf(stderr, "Invalid flag. Error 46\n");

exit(46);

}

if ((has\_only\_squared\_lines\_opts || has\_left\_up) && has\_only\_rgbfilter\_opts){

fprintf(stderr, "Invalid flag. Error 46\n");

exit(46);

}

if (has\_only\_squared\_lines\_opts && has\_only\_rotate\_opts){

fprintf(stderr, "Invalid flag. Error 46\n");

exit(46);

}

if (has\_only\_rgbfilter\_opts && (has\_only\_rotate\_opts || has\_left\_up)){

fprintf(stderr, "Invalid flag. Error 46\n");

exit(46);

}

// Проход по аргументам, не являющимися опциями (или их аргументами)

int has\_ignored\_argv = 0;

for (int i = optind; i < argc; i++){

if (!has\_ignored\_argv && (has\_help || has\_input)){

// Были введены аргументы для флага, который не принимает аргументов или введены лишние аргументы

fprintf(stderr, "An extra argument encountered. Error 44\n");

has\_ignored\_argv = 1;

}

if (has\_ignored\_argv){

fprintf(stderr, "Ignored: %s\n", argv[i]);

}

if (!has\_input && !has\_ignored\_argv){

argf->input = malloc(sizeof(char)\*strlen(argv[i]));

strcpy(argf->input, argv[i]);

has\_input = 1;

}

}

if (has\_help && !has\_ignored\_argv){

printhelp();

return 1;

}

if (has\_ignored\_argv){

exit(44);

}

if (!has\_input){

fprintf(stderr, "No input. Error 46\n");

exit(46);

}

if (!is\_bmp(argf->input)){

fprintf(stderr, "Invalid argument (the file isn't bmp). Error 45\n");

exit(45);

}

if (argf->function == 'b'){

return 0;

}

if (!argf->function){

fprintf(stderr, "Missing flag. Error 46\n");

exit(46);

}

if (!has\_output){

argf->output = malloc(sizeof(char)\*8);

strcpy(argf->output, "out.bmp");

}

if (strcmp(argf->input, argf->output) == 0){

fprintf(stderr, "Identical files. Error 47\n");

exit(47);

}

if (argf->is\_fill && !has\_fill\_color){

fprintf(stderr, "Missing flag. Error 46\n");

exit(46);

}

if ((argf->function == 'r') && (!has\_component\_name || !has\_component\_value)){

fprintf(stderr, "Missing flag. Error 46\n");

exit(46);

}

if ((argf->function == 't') && !has\_angle){

fprintf(stderr, "Missing flag. Error 46\n");

exit(46);

}

return 0;

}

Название файла: menu.c

#ifndef \_INC\_STDIO

#include <stdio.h>

#endif

#ifndef \_INC\_STDLIB

#include <stdlib.h>

#endif

#ifndef \_INC\_STRING

#include <string.h>

#endif

#ifndef \_INC\_PROCESSOPTS

#include "processopts.h"

#endif

#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;

typedef struct Rgb{

unsigned char b;

unsigned char g;

unsigned char r;

} Rgb;

#pragma pack(pop)

void print\_file\_header(BitmapFileHeader header){

puts("File header:");

printf(" signature:\t\t%x (%hu)\n", header.signature, header.signature);

printf(" filesize:\t\t%x (%u)\n", header.filesize, header.filesize);

printf(" reserved1:\t\t%x (%hu)\n", header.reserved1, header.reserved1);

printf(" reserved2:\t\t%x (%hu)\n", header.reserved2, header.reserved2);

printf(" pixelArrOffset:\t%x (%u)\n", header.pixelArrOffset, header.pixelArrOffset);

}

void print\_info\_header(BitmapInfoHeader header){

puts("Info header:");

printf(" headerSize:\t\t%x (%u)\n", header.headerSize, header.headerSize);

printf(" width: \t\t%x (%u)\n", header.width, header.width);

printf(" height: \t\t%x (%u)\n", header.height, header.height);

printf(" planes: \t\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\t%x (%u)\n", header.imageSize, header.imageSize);

printf(" xPixelsPerMeter:\t\t%x (%u)\n", header.xPixelsPerMeter, header.xPixelsPerMeter);

printf(" yPixelsPerMeter:\t\t%x (%u)\n", header.yPixelsPerMeter, header.yPixelsPerMeter);

printf(" colorsInColorTable:\t\t%x (%u)\n", header.colorsInColorTable, header.colorsInColorTable);

printf(" importantColorCount:\t%x (%u)\n", header.importantColorCount, header.importantColorCount);

}

Rgb \*\*read\_bmp(char file\_name[], BitmapFileHeader\* bmfh, BitmapInfoHeader\* bmif){

FILE \*f = fopen(file\_name, "rb");

if (!f){

fprintf(stderr, "File is missing. Error 45\n");

fclose(f);

exit(45);

}

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

fread(bmif, 1, sizeof(BitmapInfoHeader), f);

unsigned int H = bmif->height;

unsigned int W = bmif->width;

Rgb \*\*arr = malloc(H \* sizeof(Rgb\*));

int padd = 4 - (W \* sizeof(Rgb)) % 4;

for(unsigned int i = 0; i < H; i++){

arr[i] = malloc(W \* sizeof(Rgb) + padd%4);

fread(arr[i], 1, W \* sizeof(Rgb) + padd%4, f);

}

fclose(f);

return arr;

}

void write\_bmp(char \*file\_name, Rgb \*\*arr, unsigned int H, unsigned int W, BitmapFileHeader bmfh, BitmapInfoHeader bmif){

FILE \*ff = fopen(file\_name, "wb");

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

fwrite(&bmif, 1, sizeof(BitmapInfoHeader), ff);

int padd = 4 - (W \* sizeof(Rgb)) % 4;

for(unsigned int i = 0; i < H; i++){

fwrite(arr[i], 1, W \* sizeof(Rgb) + padd%4, ff);

}

fclose(ff);

}

void set\_pixel(Rgb \*\*\*bmp\_file\_pixels, unsigned int H, unsigned int W,

unsigned char r, unsigned char g, unsigned char b, int x, int y, int thickness){

y = H-y-1;

for (int i = -thickness/2; i<(thickness+1)/2; i++){

for (int j = -thickness/2; j<(thickness+1)/2; j++){

if (y+i < 0 || x+j < 0 || (unsigned int)(y+i) >= H || (unsigned int)(x+j) >= W){

continue;

}

(\*bmp\_file\_pixels)[y+i][x+j].r = r;

(\*bmp\_file\_pixels)[y+i][x+j].g = g;

(\*bmp\_file\_pixels)[y+i][x+j].b = b;

}

}

}

void fill(Rgb \*\*\*bmp\_file\_pixels, BitmapInfoHeader bmif, unsigned char r, unsigned char g, unsigned char b,

int left\_x, int up\_y, int right\_x, int down\_y){

for (int y = up\_y; y<down\_y; y++){

for (int x = left\_x; x<right\_x; x++){

set\_pixel(bmp\_file\_pixels, bmif.height, bmif.width, r, g, b, x, y, 1);

}

}

}

void swap\_points(int \*x0, int \*y0, int \*x1, int \*y1){

int c = \*x0;

\*x0 = \*x1;

\*x1 = c;

c = \*y0;

\*y0 = \*y1;

\*y1 = c;

}

void draw\_line(Rgb \*\*\*bmp\_file\_pixels, BitmapInfoHeader bmif, unsigned char r, unsigned char g, unsigned char b,

int x0, int y0, int x1, int y1, int thickness){

//y = (x-x0)\*(y1-y0)/(x1-x0)+y0

int deltax = abs(x1 - x0);

int deltay = abs(y1 - y0);

float error = 0;

if (deltax>=deltay){

if (x0 > x1){

swap\_points(&x0, &y0, &x1, &y1);

x0 += 1;

x1 += 1;

if (y1 - y0 > 0){

y0 += 1;

y1 += 1;

}

else if (y1 - y0 < 0){

y0 -= 1;

y1 -= 1;

}

}

int deltaerr = (deltay + 1);

int y = y0;

int diry = y1 - y0;

if (diry > 0)

diry = 1;

if (diry < 0)

diry = -1;

for (int x = x0; x < x1; x++){

set\_pixel(bmp\_file\_pixels, bmif.height, bmif.width, r, g, b, x, y, thickness);

error = error + deltaerr;

if (error >= (deltax + 1)){

y = y + diry;

error = error - (deltax + 1);

}

}

}

else{

if (y0 > y1){

swap\_points(&x0, &y0, &x1, &y1);

y0 += 1;

y1 += 1;

if (x1 - x0 > 0){

x0 += 1;

x1 += 1;

}

else if (x1 - x0 < 0){

x0 -= 1;

x1 -= 1;

}

}

int deltaerr = (deltax + 1);

int x = x0;

int dirx = x1 - x0;

if (dirx > 0)

dirx = 1;

if (dirx < 0)

dirx = -1;

for (int y = y0; y < y1; y++){

set\_pixel(bmp\_file\_pixels, bmif.height, bmif.width, r, g, b, x, y, thickness);

error = error + deltaerr;

if (error >= (deltay + 1)){

x = x + dirx;

error = error - (deltay + 1);

}

}

}

}

void squared\_lines(Rgb \*\*\*bmp\_file\_pixels, BitmapInfoHeader bmif, unsigned char r, unsigned char g, unsigned char b,

int x0, int y0, int side, int thickness){

draw\_line(bmp\_file\_pixels, bmif, r, g, b, x0, y0, x0+side, y0+side, thickness);

draw\_line(bmp\_file\_pixels, bmif, r, g, b, x0, y0, x0+side, y0, thickness);

draw\_line(bmp\_file\_pixels, bmif, r, g, b, x0, y0, x0, y0+side, thickness);

draw\_line(bmp\_file\_pixels, bmif, r, g, b, x0, y0+side-1, x0+side, y0-1, thickness);

draw\_line(bmp\_file\_pixels, bmif, r, g, b, x0+side-1, y0, x0+side-1, y0+side, thickness);

draw\_line(bmp\_file\_pixels, bmif, r, g, b, x0, y0+side-1, x0+side, y0+side-1, thickness);

}

void rgbfilter(Rgb \*\*\*bmp\_file\_pixels, BitmapInfoHeader bmif, char component\_name, int component\_value){

for (unsigned int y = 0; y<bmif.height; y++){

for (unsigned int x = 0; x<bmif.width; x++){

switch (component\_name)

{

case 'r':

{

(\*bmp\_file\_pixels)[y][x].r = component\_value;

break;

}

case 'g':

{

(\*bmp\_file\_pixels)[y][x].g = component\_value;

break;

}

case 'b':

{

(\*bmp\_file\_pixels)[y][x].b = component\_value;

break;

}

}

}

}

}

void angle(Rgb \*\*\*bmp\_file\_pixels, Rgb \*\*\*original\_bf\_pixels, BitmapInfoHeader bmif, int left\_x, int up\_y,

int right\_x, int down\_y, int angle){

int tmp;

if (right\_x < left\_x){

tmp = right\_x;

right\_x = left\_x;

left\_x = tmp;

}

if (down\_y < up\_y){

tmp = down\_y;

down\_y = up\_y;

up\_y = tmp;

}

//Проверка на выход за границы при взятии пикселей с оригинала

if (up\_y < 0){

up\_y = 0;

}

if ((unsigned int)(down\_y) >= bmif.height){

down\_y = bmif.height-1;

}

if (left\_x < 0){

left\_x = 0;

}

if ((unsigned int)(right\_x) >= bmif.width){

right\_x = bmif.width-1;

}

int r;

int g;

int b;

int shift;

switch (angle)

{

case 90:

{

shift = (down\_y - up\_y - (right\_x - left\_x)) / 2;

for (int x = right\_x; x > left\_x; x--){

for (int y = up\_y; y < down\_y; y++){

r = (\*original\_bf\_pixels)[bmif.height-y-1][x-1].r;

g = (\*original\_bf\_pixels)[bmif.height-y-1][x-1].g;

b = (\*original\_bf\_pixels)[bmif.height-y-1][x-1].b;

set\_pixel(bmp\_file\_pixels, bmif.height, bmif.width, r, g, b, left\_x+y-shift-up\_y, up\_y+right\_x-x+shift, 1);

}

}

break;

}

case 180:

{

for (int y = down\_y; y > up\_y; y--){

for (int x = right\_x; x > left\_x; x--){

r = (\*original\_bf\_pixels)[bmif.height-y][x-1].r;

g = (\*original\_bf\_pixels)[bmif.height-y][x-1].g;

b = (\*original\_bf\_pixels)[bmif.height-y][x-1].b;

set\_pixel(bmp\_file\_pixels, bmif.height, bmif.width, r, g, b, left\_x+right\_x-x, up\_y+down\_y-y, 1);

}

}

break;

}

case 270:

{

shift = (down\_y - up\_y - (right\_x - left\_x)) / 2;

for (int x = left\_x; x < right\_x; x++){

for (int y = down\_y; y > up\_y; y--){

r = (\*original\_bf\_pixels)[bmif.height-y][x].r;

g = (\*original\_bf\_pixels)[bmif.height-y][x].g;

b = (\*original\_bf\_pixels)[bmif.height-y][x].b;

set\_pixel(bmp\_file\_pixels, bmif.height, bmif.width, r, g, b, left\_x+down\_y-y-shift, up\_y+x+shift-left\_x, 1);

}

}

break;

}

}

}

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

Argf argf;

if (processopts(argc, argv, &argf)){

return 0;

}

BitmapFileHeader bmfh;

BitmapInfoHeader bmif;

Rgb \*\*bmp\_file\_pixels = read\_bmp(argf.input, &bmfh, &bmif);

if (argf.function == 'b'){

puts("Course work for option 4.12, created by Nikita Kozmin");

print\_file\_header(bmfh);

print\_info\_header(bmif);

return 0;

}

else if (argf.function == 'q'){

if (argf.is\_fill){

fill(&bmp\_file\_pixels, bmif, argf.fill\_color\_r, argf.fill\_color\_g, argf.fill\_color\_b,

argf.left\_up\_x, argf.left\_up\_y, argf.left\_up\_x+argf.side\_size, argf.left\_up\_y+argf.side\_size);

}

squared\_lines(&bmp\_file\_pixels, bmif, argf.color\_r, argf.color\_g, argf.color\_b,

argf.left\_up\_x, argf.left\_up\_y, argf.side\_size, argf.thickness);

}

else if (argf.function == 'r'){

rgbfilter(&bmp\_file\_pixels, bmif, argf.component\_name, argf.component\_value);

}

else if (argf.function == 't'){

Rgb \*\*original\_bf\_pixels = read\_bmp(argf.input, &bmfh, &bmif);

angle(&bmp\_file\_pixels, &original\_bf\_pixels, bmif, argf.left\_up\_x,

argf.left\_up\_y, argf.right\_down\_x, argf.right\_down\_y, argf.angle);

}

write\_bmp(argf.output, bmp\_file\_pixels, bmif.height, bmif.width, bmfh, bmif);

puts("Course work for option 4.12, created by Nikita Kozmin");

return 0;

}

**ПРИЛОЖЕНИЕ Б**

**ТЕСТИРОВАНИЕ**

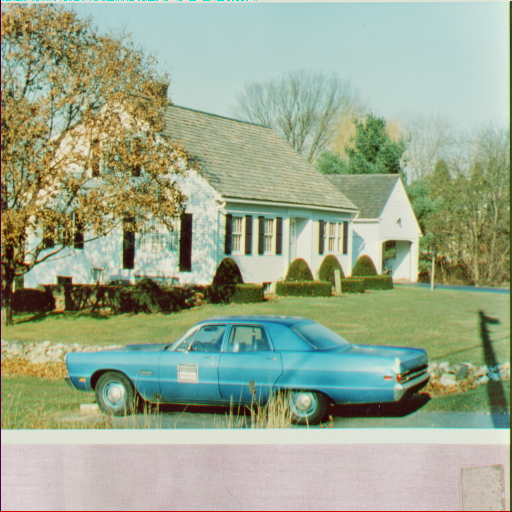


Рисунок 1 – Подаваемое изображение

Таблица 1 – Результаты тестирования функциональности

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
|  | cw --squared\_lines ..\Inp.bmp --left\_up 100.100 --side\_size 100 --thickness 5 -o ..\Squared\_lines.bmp | Course work for option 4.12, created by Nikita Kozmin |
|  | cw --rgbfilter -o ..\Rgbfilter.bmp -i ..\Inp.bmp --component\_name red --component\_value 255 | Course work for option 4.12, created by Nikita Kozmin |
|  | cw --rotate ..\Inp.bmp --left\_up 30.315 --right\_down 430.415 --angle 90 --output ..\Rotate.bmp | Course work for option 4.12, created by Nikita Kozmin |

Таблица 2 – Результаты тестирования ошибок

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | cw -help | cw: unknown option -- e  Invalid or missing flag or argument. Error 46 or 45 | Пропущено второе тире |
|  | cw --rotate --left\_up 30.315 --right\_down 430.415 --angle 90 --output ..\Rotate.bmp | No input. Error 46 |  |
|  | cw --rgbfilter -o ..\Rgbfilter.bmp -i ..\Inp.bmp --component\_name red --component\_value 300 ..\Inp.bmp | Invalid color. Error 41 |  |