**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**отчет**

**по курсовой работе**

**по дисциплине «Программирование»**

**Тема: Работа с бинарными файлами в языке С**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9304 |  | Каменская Е.К. |
| Преподаватель |  | Чайка К.В. |

Санкт-Петербург

2020

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент: Каменская Е.К. | | |
| Группа: 9304 | | |
| Тема работы: Работа с бинарными файлами в языке С | | |
| Исходные данные: Файл формата bmp. Модель цвета RGB с 24 битами на пиксель, без сжатия, без индексации цветов. | | |
| Содержание пояснительной записки: «Содержание», «Введение», «Исходное задание», «Описание кода программы», «Примеры работы программы», «Примеры обработки ошибок», «Заключение», «Список источников», «Приложение А», «Комментарии из пулл-реквестов» | | |
| Предполагаемый объем пояснительной записки:  Не менее 20 страниц. | | |
| Дата выдачи задания: 01.03.2020 | | |
| Дата сдачи реферата: 22.05.2020 | | |
| Дата защиты реферата: 22.05.2020 | | |
| Студент |  | Каменская Е.К. |
| Преподаватель |  | Чайка К.В. |

Аннотация

Курсовая работа представляет собой программу для считывания, обработки и сохранения bmp файлов в соответствии с выбранным пользователем действием. Код программы написан на языке программирования С с использованием функций стандартных библиотек и управляющих конструкций и предназначен для запуска на операционных системах семейства Linux. Для выполнения подзадач, а также чтения и сохранения файлов были реализованы собственные функции. Для проверки работоспособности и использования памяти было проведено тестирование. Результаты тестирования и исходный код программы представлены в соответствующих разделах и Приложении А соответственно.

This coursework is a program for reading, processing and saving bmp files in accordance with the action selected by the user. The program code is written in the C programming language using the functions of standard libraries and control structures and is designed to run on Linux operating systems. To perform subtasks, as well as read and save files, native functions were implemented. To test the working capacity and memory usage, testing was conducted. The test results and the source code of the program are presented in the corresponding sections and Appendix A, respectively.

Оглавление

[Аннотация 3](#__RefHeading___Toc210_4135906360)

[Введение 5](#__RefHeading___Toc1451_4162300111)

[Исходное задание 6](#__RefHeading___Toc1453_4162300111)

[Описание кода программы 7](#__RefHeading___Toc1455_4162300111)

[Структуры данных 7](#__RefHeading___Toc1457_4162300111)

[Общие функции 7](#__RefHeading___Toc1459_4162300111)

[Функции подзадач 9](#__RefHeading___Toc1461_4162300111)

[Примеры работы программы 11](#__RefHeading___Toc1463_4162300111)

[Примеры обработки ошибок 17](#__RefHeading___Toc1465_4162300111)

[Заключение 18](#__RefHeading___Toc1467_4162300111)

[Источники 19](#__RefHeading___Toc1469_4162300111)

Введение

Цель работы - разработка программы на языке С++ согласно заданию, используя для этого знания, полученные в течение двух семестров по предмету Программирование. Для достижения поставленной цели требуется решить следующие задачи: изучение устройства bmp файла, разработка интерфейса и функций программы, сборка и тестирование.

Исходное задание

**Вариант 2**

Программа должна иметь CLI или GUI. Более подробно тут:

http://se.moevm.info/doku.php/courses:programming:rules*extra*kurs

**Общие сведения**

• 24 бита на цвет

• без сжатия

• файл всегда соответствует формату BMP (но стоит помнить, что версий у формата несколько)

• обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями.

• обратите внимание на порядок записи пикселей

• все поля стандартных BMP заголовков в выходном файле должны иметь те же значения что и во входном (разумеется кроме тех, которые должны быть изменены).

**Программа должна реализовывать весь следующий функционал по обработке bmp-файла**

Отражение заданной области. Этот функционал определяется:

• Выбором оси относительно которой отражать (горизонтальная или вертикальная)

• Координатами левого верхнего угла области

• Координатами правого нижнего угла области

Копирование заданной области. Функционал определяется:

• Координатами левого верхнего угла области-источника

• Координатами правого нижнего угла области-источника

• Координатами левого верхнего угла области-назначения

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

• Цвет, который требуется заменить

• Цвет на который требуется заменить

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

Описание кода программы

Структуры данных

**BitmapFileHeader** и **BitmapInfoHeader** содержат информацию о bmp файле. **Rgb** - структура, из которой строится массив пикселей файла.

Общие функции

*int main(int argc, char \*\*argv)* - отвечает за взаимодействие с пользователем через командную строку при помощи getopt*long и за последовательное выполнения команд и сохранение файла(ов). В связи с тем, что название исходного файла по умолчанию вводится в конце списка команд пользователя, помимо цикла считывания команд был реализован также цикл для их выполнения, находящийся в функции \_processing*. Во время парсинга строки, введенной пользователем в массив *actions* заносятся флаги, соответствующие введенным действиям. В случае, если исходный файл корректен, создается структура *params*, в которую заносятся все данные, которые могут понадобиться в функциях, и запускается функция *processing*. После этого происходит очистка памяти, выделенной динамически, и программа завершается.

*int processing(Params\* params, Rgb\*\* arr, BitmapInfoHeader\* bmih, BitmapFileHeader\* bmfh)* - итерируется по массиву *actions* и, в зависимости от флага, выполняет ту или иную функцию.

*int fillArr(char\* filename, Rgb\*\*\* arr*p, BitmapFileHeader\* bmfh*p, BitmapInfoHeader\* bmih*p)\_ - считывает заголовок и сам файл, динамически выделяя память под массив пикселей. Перед считыванием массива проверяет, поддерживается ли формат файла.

*void memoClean(Rgb\*\* arr, int H, char\* filename, char\* alternative, char\* saveto)* - получает на вход все указатели, под которые могла выделяться динамическая память при работе программы. Перед очисткой памяти проверяет, является ли указатель нулевым.

*int readOperands(int argc, int optind, char\*\* argv, int\* operands, int operand*count)\_ - получает на вход количество аргументов, переданных пользователем, индекс, на котором остановился getopt*long, указатели на массив аргументов и массив для их сохранения, а также число нужных аргументов и копирует их из argv в operands. Возвращает новый индекс для getopt*long.

*int typeCheck(BitmapFileHeader\* bmfh, BitmapInfoHeader\* bmih)* - проверяет формат файла и в случае неподдерживаемого формата возвращает 0. Если формат поддерживается программой, возвращается 1.

*int operandCheck(int flag, BitmapInfoHeader\* bmih, int\* start, int\* end, int\* to, int\* old*color, int\* new*color)* - получает флаг, в соответствии с которым нужно проверить переменные, указатели на которые также переданы. Чтобы избежать ситуации, когда часть данных не введена, но программа все равно выполнилась, все переменные изначально инициализированы как -1, то есть непригодные для использования. Поэтому при проверке переменных для конкретной функции затрагиваются только необходимые для ее работы, что и обеспечивает наличие флага. При прохождении проверки всеми переменными функция вернет 1, в обратном случае - 0.

*int saveFile(const char\* filename, Rgb\*\* arr, BitmapFileHeader\* bmfh*p, BitmapInfoHeader\* bmih*p, int H, int W, int coordY, int coordX)* - сохраняет новый bmp файл. Для удобства с использованием при разделении файла была добавлена возможность изменять параметры в заголовке и передача координат нижнего левого угла фрагмента изображения. При успешном сохранении функция вернет 1, иначе - 0.

*void printInfo(BitmapFileHeader bmfh, BitmapInfoHeader bmih)* - распечатывает информацию о файле из BitmapFileHeader и BitmapInfoHeader.

*void printHelp()* - распечатывает справку по использованию утилиты.

Функции подзадач

*int copyPatch(Rgb\*\* arr, BitmapInfoHeader\* bmih*p, int\* left, int\* right, int\* to)\_ - **копирование фрагмента** изображения. После подсчета ширины и высоты фрагмента и его обрезки (в случае, если он не помещается в пределах изображения) в цикле перемещаются на новое место элементы массива пикселей arr. При успешном завершении копирования функция вернет 1, иначе - 0.

*int divideFile(Rgb\*\* arr, BitmapFileHeader\* bmfh*p, BitmapInfoHeader\* bmih*p, int countH, int countW, char\* prefix)* - **разделение изображения**. Получает указатель на массив пикселей, информацию об исходном файле, количество частей по оси Y и X, а также название директории для сохранения полученных фрагментов. Во вложенном цикле задает имя для каждого фрагмента, в соответствии с его расположением в исходном изображении и передает новые данные в функцию для записи нового файла saveFile. При удачном завершении возвращает 1, иначе - 0.

*int reflectFragmentX(Rgb\*\* arr, int\* left, int\* right)* - **отражение фрагмента относительно ОХ**. Копирует в буфер верхнюю половину фрагмента, перемещает перевернутую нижнюю половину на место верхней и вставляет перевернутую верхнюю половину из буфера на место нижней, после чего очищает буфер.

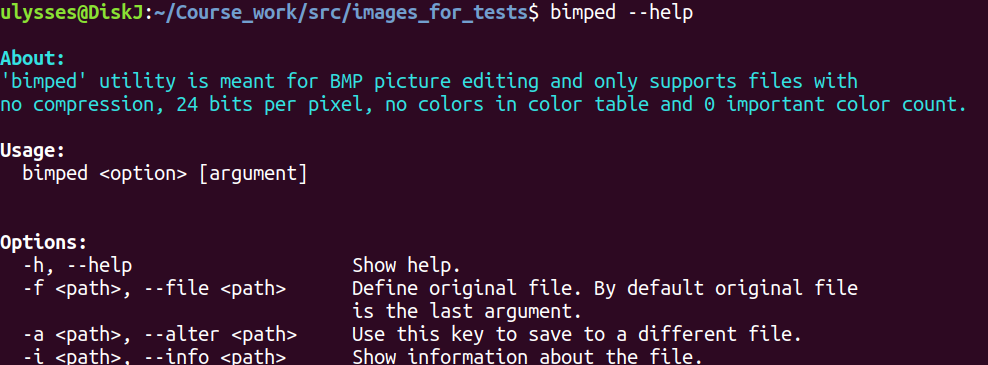
*int reflectFragmentY(Rgb\*\* arr, int\* left, int\* right)* - **отражение фрагмента относительно OY**. Меняет местами симметричные относительно середины фрагменты пиксели в каждой “строке”.

*void replaceColor(Rgb\*\* arr, BitmapInfoHeader\* bmih, int\* old*color, int\* new*color)* - **замена цвета**. Так как в формате bmp пиксели примерно одного цвета задаются разными RGB параметрами, было принято решение заменять также цвета близкие к исходному. Функция итерируется по всему массиву пикселей и, если цвет близок к заданному пользователем как старый, заменяет его на новый.

Примеры работы программы

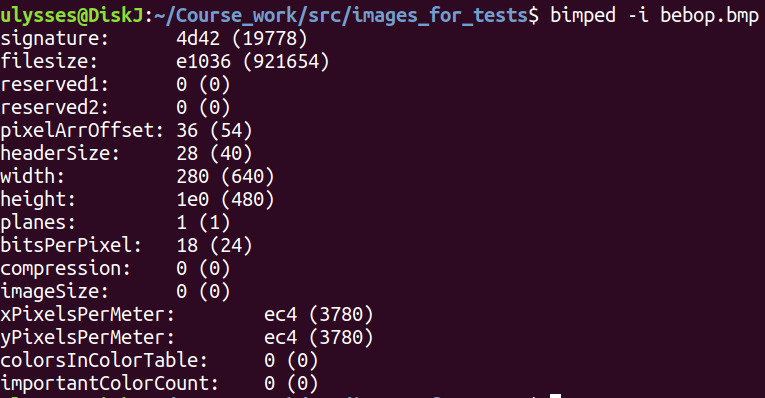
**help**

*bimped --help*



**info**

*bimped -i bebop.bmp*



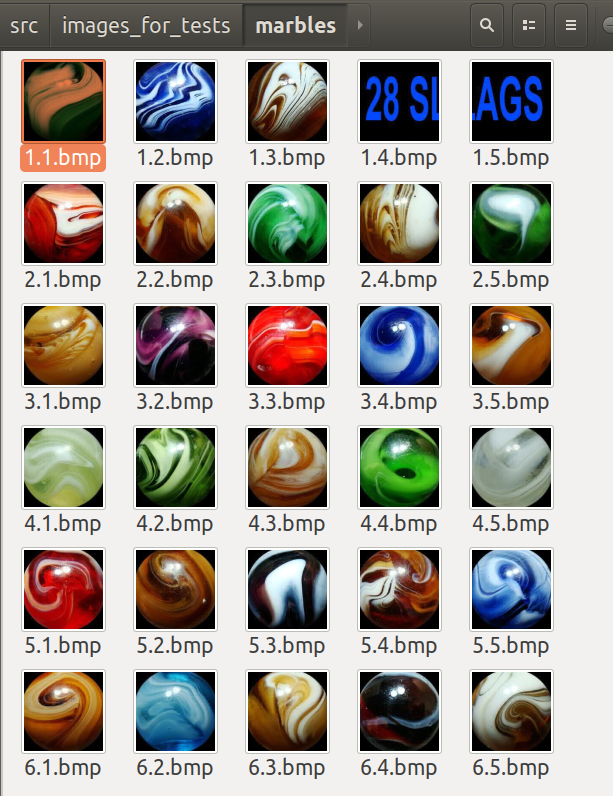
**divide**

*bimped --file=92.bmp -a ./marbles/ -d 5 6*

Исходный файл 92.bmp



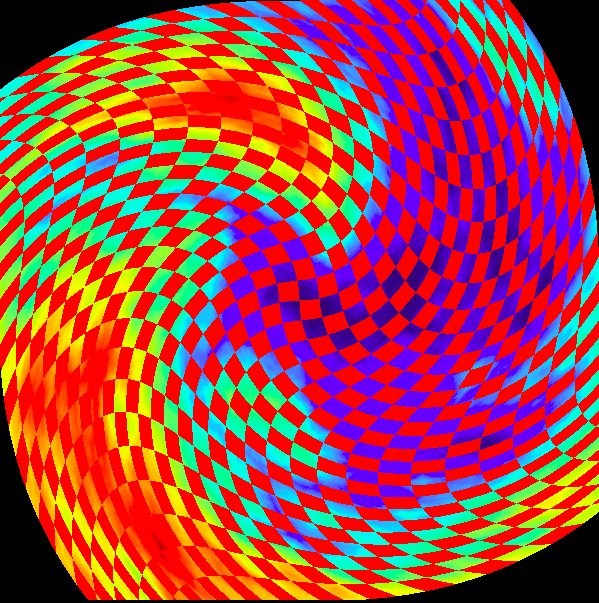
Результат в папке ./marbles



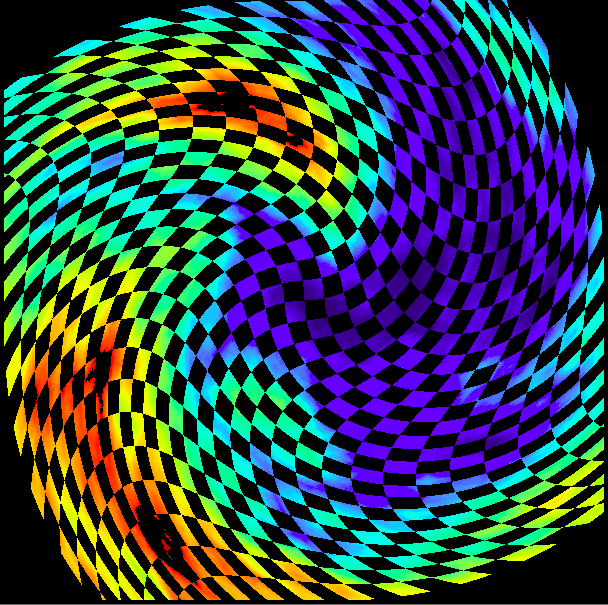
**replace-color**

*bimped --replace-color --old 255 0 0 --new 0 0 0 -a black.bmp swirl.bmp*

Исходный файл swirl.bmp



Результат black.bmp



**reflect-y**

*bimped -y -s 100 500 -e 500 0 -a reflection.bmp lena.bmp*

Исходный файл lena.bmp



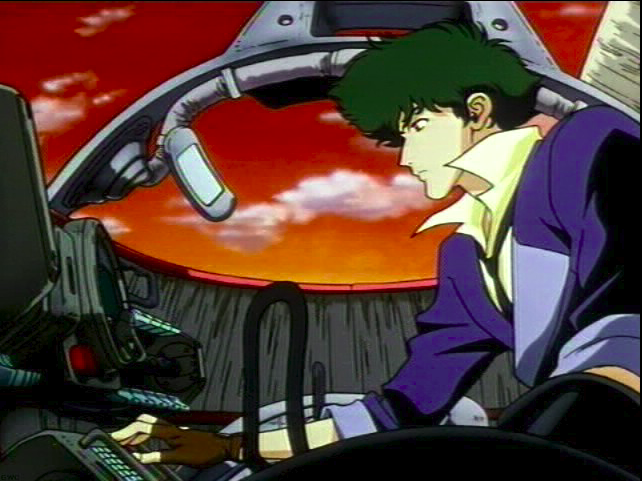
Результат reflection.bmp



**reflect-x**

*bimped -x -s 0 400 -e 300 0 -a /home/ulysses/Course\_work/x-reflect.bmp bebop.bmp*

Исходный файл bebop.bmp



Результат в папке Course\_work x-reflect.bmp



**copy-patch**

*bimped --copy-patch -s 600 1199 -e 999 1000 -t 200 400 --alter=28.bmp 92.bmp*

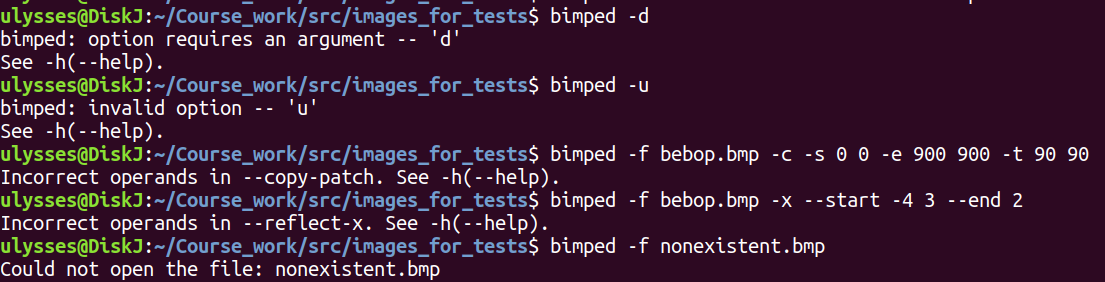
Исходный файл 92.bmp

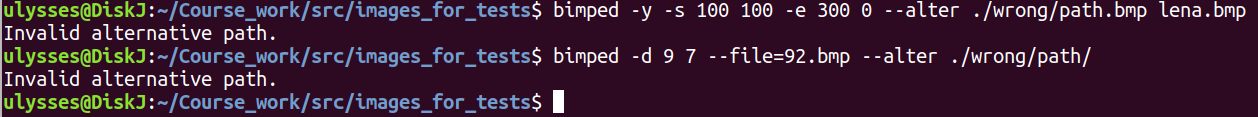


Результат 28.bmp



Примеры обработки ошибок





Заключение

Для выполнения задания были использованы знания о работе с функциями нескольких библиотек С и С++, динамической памятью, командной строкой, файлами и структурами, полученные за 2 семестра. Была написана и разделена на подзадачи собственная утилита, а также проведено тестирование. Таким образом, в ходе написания курсовой работы быди закреплены фундаментальные навыки создания программ с интерфесом в командной строке на языке С++.

Источники

Документация Linux [Электронный ресурс]  
URL: https://linux.die.net/man

Документация библиотек языков Си и C++ [Электронный ресурс]  
URL: http://cplusplus.com

Приложение A

./src/Makefile

1 CC=g++

2 FLAG=

3

4 all: bimped clean

5

6 bimped: editor.o printInfoHelp.o saveFile.o checker.o argumentHandler.o tasks.o processing.o

7 $(CC) $(FLAG) editor.o printInfoHelp.o saveFile.o checker.o argumentHandler.o tasks.o processing.o -o bimped

8

9 editor.o: editor.cpp structures.hpp printInfoHelp.hpp tasks.hpp saveFile.hpp checker.hpp argumentHandler.hpp processing.hpp

10 $(CC) $(FLAG) -c editor.cpp

11

12 printInfoHelp.o: printInfoHelp.cpp printInfoHelp.hpp

13 $(CC) $(FLAG) -c printInfoHelp.cpp

14

15 saveFile.o: saveFile.cpp saveFile.hpp

16 $(CC) $(FLAG) -c saveFile.cpp

17

18 checker.o: checker.cpp checker.hpp

19 $(CC) $(FLAG) -c checker.cpp

20

21 argumentHandler.o: argumentHandler.cpp argumentHandler.hpp checker.hpp

22 $(CC) $(FLAG) -c argumentHandler.cpp

23

24 tasks.o: tasks.cpp tasks.hpp saveFile.hpp

25 $(CC) $(FLAG) -c tasks.cpp

26

27 processing.o: processing.cpp processing.hpp tasks.hpp checker.hpp saveFile.hpp

28 $(CC) $(FLAG) -c processing.cpp

29

30 clean:

31 rm -f \*.o

./src/editor.cpp

1 #include <iostream>

2 #include <cstdlib>

3 #include <cstring>

4 #include <getopt.h>

5 #include "structures.hpp"

6 #include "printInfoHelp.hpp"

7 #include "tasks.hpp"

8 #include "saveFile.hpp"

9 #include "checker.hpp"

10 #include "argumentHandler.hpp"

11 #include "processing.hpp"

12

13 //using namespace std;

14

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

16

17 int c;

18 int index;

19 char\* next;

20 int i = 0;

21

22 char\* filename = NULL;

23 char\* alternative = NULL;

24

25 FILE\* f = NULL;

26 Rgb \*\*arr = NULL;

27 BitmapFileHeader bmfh;

28 BitmapInfoHeader bmih;

29

30 int counts[2] = {-1,-1};

31 int start[2] = {-1,-1};

32 int end[2] = {-1,-1};

33 int to[2] = {-1,-1};

34 int old\_color[3] = {-1,-1,-1};

35 int new\_color[3] = {-1,-1,-1};

36

37 char actions[100];

38 int action\_counter = 0;

39

40 while (action\_counter < 100)

41 {

42 static struct option long\_options[] = {

43 {"info", required\_argument, 0, 'i'},

44 {"file", required\_argument, 0, 'f'},

45 {"copy-patch", no\_argument, 0, 'c'},

46 {"reflect-x", no\_argument, 0, 'x'},

47 {"reflect-y", no\_argument, 0, 'y'},

48 {"replace-color", no\_argument, 0, 'r'},

49 {"divide", required\_argument, 0, 'd'},

50 {"alter", required\_argument, 0, 'a'},

51 {"start", required\_argument, 0, 's'},

52 {"end", required\_argument, 0, 'e'},

53 {"old", required\_argument, 0, 'o'},

54 {"new", required\_argument, 0, 'n'},

55 {"to", required\_argument, 0, 't'},

56 {"help", no\_argument, 0, 'h'},

57 {0, 0, 0, 0}

58 };

59

60 int option\_index = 0;

61

62 c = getopt\_long (argc, argv, "i:hf:a:cxyrd:s:e:o:n:t:", long\_options, &option\_index);

63

64 if (c == -1)

65 break;

66 switch (c){

67

68 case 'i': // info

69 filename = (char\*)malloc((strlen(optarg)+1)\*sizeof(char));

70 strcpy(filename, optarg);

71 f = fopen(filename, "rb");

72 if(f==NULL){

73 printf("Could not open the file.\n");

74 free(filename);

75 return 1;

76 }

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

78 fread(&bmih,1,sizeof(BitmapInfoHeader),f);

79 printInfo(bmfh, bmih);

80 if(!typeCheck(&bmfh, &bmih)){

81 printf("Unfortunately, this format is not supported. See -h(--help).\n");

82 }

83 fclose(f);

84 free(filename);

85 break;

86

87 case 'h': // help

88 printHelp();

89 return 0;

90

91 case 'f': // file

92 filename = (char\*)realloc(filename, (strlen(optarg)+1)\*sizeof(char));

93 strcpy(filename, optarg);

94 if(!fillArr(filename, &arr, &bmfh, &bmih) || arr==NULL){

95 free(filename);

96 return 1;

97 }

98 break;

99

100 case 'a': // alternative

101 alternative = (char\*)malloc((strlen(optarg)+1)\*sizeof(char));

102 strcpy(alternative, optarg);

103 break;

104

105 case 'c': // --copy-patch no args

106 actions[action\_counter++] = 'c';

107 break;

108

109 case 'x': // --reflect-x no args

110 actions[action\_counter++] = 'x';

111 break;

112

113 case 'y': // --reflect-y no args

114 actions[action\_counter++] = 'y';

115 break;

116

117 case 'r': // --replace no args

118 actions[action\_counter++] = 'r';

119 break;

120

121 case 'd': // --divide 2 args

122 optind = readOperands(argc, optind, argv, counts, 2);

123 actions[action\_counter++] = 'd';

124 break;

125

126 case 's': // start

127 optind = readOperands(argc, optind, argv, start, 2);

128 break;

129

130 case 'e': // end

131 optind = readOperands(argc, optind, argv, end, 2);

132 break;

133

134 case 't': // to

135 optind = readOperands(argc, optind, argv, to, 2);

136 break;

137

138 case 'o': // 3 args

139 optind = readOperands(argc, optind, argv, old\_color, 3);

140 break;

141

142 case 'n': // 3 args

143 optind = readOperands(argc, optind, argv, new\_color, 3);

144 break;

145

146 case ':': /\* error - missing operand \*/

147 fprintf(stderr, "Option -%c requires an operand. See -h(--help).\n", optopt);

148 break;

149

150 case '?': /\* error - unknown option \*/

151 fprintf(stderr, "See -h(--help).\n");

152 return 1;

153

154 default:

155 return 0;

156 }

157

158 }

159

160 if(filename == NULL){

161 if (optind < argc){

162 while (optind < argc){

163 filename = (char\*)realloc(filename, (strlen(argv[optind])+1)\*sizeof(char));

164 strcpy(filename, argv[optind]);

165 optind++;

166 }

167 }

168 else{

169 printf("Filename not entered. See -h(--help).\n");

170 //memoClean(arr, bmih.height, filename, alternative);

171 if(alternative)

172 free(alternative);

173 return 1;

174 }

175 if(!fillArr(filename, &arr, &bmfh, &bmih) || arr==NULL){

176 free(filename);

177 if(alternative)

178 free(alternative);

179 return 1;

180 }

181 }

182 Params params;

183 params.filename = filename;

184 params.alternative = alternative;

185 params.counts = counts;

186 params.start = start;

187 params.end = end;

188 params.to = to;

189 params.old\_color = old\_color;

190 params.new\_color = new\_color;

191 params.actions = actions;

192 params.action\_counter = action\_counter;

193

194 processing(&params, arr, &bmih, &bmfh);

195 memoClean(arr, bmih.height, filename, alternative);

196

197 return 0;

198 }

./src/processing.hpp

1 #pragma once

2 #include "structures.hpp"

3 #include "saveFile.hpp"

4 #include "tasks.hpp"

5 #include "checker.hpp"

6

7 int processing(Params\* params, Rgb\*\* arr, BitmapInfoHeader\* bmih, BitmapFileHeader\* bmfh);

./src/processing.cpp

1 #include "processing.hpp"

2

3 int processing(Params\* params, Rgb\*\* arr, BitmapInfoHeader\* bmih, BitmapFileHeader\* bmfh){

4 char\* saveto = NULL;

5

6 if(params->alternative == NULL){

7 saveto = (char\*)calloc(strlen(params->filename)+1, sizeof(char));

8 strcpy(saveto, params->filename);

9 }else{

10 saveto = (char\*)calloc(strlen(params->alternative)+1, sizeof(char));

11 strcpy(saveto, params->alternative);

12 }

13

14 char c;

15 for(int i = 0; i<params->action\_counter; i++){

16 c = params->actions[i];

17 switch (c){

18 case 'c':

19 if(operandCheck(c, bmih, params->start, params->end, params->to, params->old\_color, params->new\_color)

20 && copyPatch(arr, bmih, params->start, params->end, params->to)){

21 saveFile(saveto, arr, bmfh, bmih, 0, 0);

22 }

23 else{

24 printf("Incorrect operands in --copy-patch. See -h(--help).\n");

25 return 0;

26 }

27 break;

28

29 case 'x':

30 if(operandCheck(c, bmih, params->start, params->end, params->to, params->old\_color, params->new\_color)

31 && reflectFragmentX(arr, params->start, params->end)){

32 saveFile(saveto, arr, bmfh, bmih, 0, 0);

33 }

34 else{

35 printf("Incorrect operands in --reflect-x. See -h(--help).\n");

36 return 0;

37 }

38 break;

39

40 case 'y':

41 if(operandCheck(c, bmih, params->start, params->end, params->to, params->old\_color, params->new\_color)

42 && reflectFragmentY(arr, params->start, params->end)){

43 saveFile(saveto, arr, bmfh, bmih, 0, 0);

44 }

45 else{

46 printf("Incorrect operands in --reflect-y. See -h(--help).\n");

47 return 0;

48 }

49 break;

50

51 case 'r':

52 if(operandCheck(c, bmih, params->start, params->end, params->to, params->old\_color, params->new\_color)){

53 replaceColor(arr, bmih, params->old\_color, params->new\_color);

54 saveFile(saveto, arr, bmfh, bmih, 0, 0);

55 }

56 else{

57 printf("Incorrect operands in --replace-color. See -h(--help).\n");

58 return 0;

59 }

60 break;

61

62 case 'd':

63 if(params->alternative != NULL && params->alternative[strlen(params->alternative)-1] != '/'){

64 printf("Invalid alternative path for --divide. Must be a directory.\n");

65 //memoClean(arr, bmih->height, params->filename, params->alternative, saveto);

66 return 0;

67 }

68 char\* saveto\_copy = NULL;

69

70 if(params->alternative == NULL){

71 int len = strlen(params->filename)-1;

72 while(len>=0 && params->filename[len]!='/'){

73 len = len - 1;

74 }

75 if(len==-1){ // save to current directory

76 saveto\_copy = (char\*)calloc(3, sizeof(char));

77 strcpy(saveto\_copy, "./");

78 }else{ // save to the same directory as original file

79 saveto\_copy = (char\*)calloc(len+2, sizeof(char));

80 memmove(saveto\_copy, params->filename, (len+1)\*sizeof(char));

81 }

82 }else{ // save to alternative directory

83 saveto\_copy = (char\*)calloc(strlen(params->alternative)+1, sizeof(char));

84 strcpy(saveto\_copy, params->alternative);

85 }

86 if(!divideFile(arr, bmfh, bmih, params->counts[1], params->counts[0], saveto\_copy)){

87 printf("Incorrect operands in --divide. See -h(--help).\n");

88 return 0;

89 }

90 else

91 printf("Result will be saved in: %s\n", saveto\_copy);

92 if(saveto\_copy)

93 free(saveto\_copy);

94 break;

95 }

96 }

97 if(saveto){

98 free(saveto);

99 }

100 return 1;

101 }

./src/structures.hpp

1 #pragma once

2 #include <iostream>

3 #include <cstdlib>

4 #include <cstring>

5

6 #pragma pack (push, 1)

7 typedef struct

8 {

9 unsigned short signature;

10 unsigned int filesize;

11 unsigned short reserved1;

12 unsigned short reserved2;

13 unsigned int pixelArrOffset;

14 } BitmapFileHeader;

15

16 typedef struct

17 {

18 unsigned int headerSize;

19 unsigned int width;

20 unsigned int height;

21 unsigned short planes;

22 unsigned short bitsPerPixel;

23 unsigned int compression;

24 unsigned int imageSize;

25 unsigned int xPixelsPerMeter;

26 unsigned int yPixelsPerMeter;

27 unsigned int colorsInColorTable;

28 unsigned int importantColorCount;

29 } BitmapInfoHeader;

30

31 typedef struct

32 {

33 unsigned char b;

34 unsigned char g;

35 unsigned char r;

36 } Rgb;

37

38 typedef struct

39 {

40 char\* filename;

41 char\* alternative;

42 int\* counts;

43 int\* start;

44 int\* end;

45 int\* to;

46 int\* old\_color;

47 int\* new\_color;

48 char\* actions;

49 int action\_counter;

50 } Params;

51

52 #pragma pack(pop)

./src/argumentHandler.hpp

1 #pragma once

2

3 #include "structures.hpp"

4 #include "checker.hpp"

5

6 int fillArr(char\* filename, Rgb\*\*\* arr\_p, BitmapFileHeader\* bmfh\_p, BitmapInfoHeader\* bmih\_p);

7

8 int readOperands(int argc, int optind, char\*\* argv, int\* operands, int operand\_count);

9

10 void memoClean(Rgb\*\* arr, int H, char\* filename, char\* alternative);

./src/argumentHandler.cpp

1 #include "argumentHandler.hpp"

2

3 int fillArr(char\* filename, Rgb\*\*\* arr\_p, BitmapFileHeader\* bmfh\_p, BitmapInfoHeader\* bmih\_p){

4 FILE\* f = fopen(filename, "rb");

5 if(f==NULL){

6 fprintf(stderr, "Co!uld not open the file: %s\n", filename);

7 return 0;

8 }

9 fread(bmfh\_p,1,sizeof(BitmapFileHeader),f);

10 fread(bmih\_p,1,sizeof(BitmapInfoHeader),f);

11 if(!typeCheck(bmfh\_p, bmih\_p)){

12 printf("Unfortunately, this format is not supported. See -h(--help).\n");

13 return 0;

14 }

15

16 unsigned int H = bmih\_p->height;

17 unsigned int W = bmih\_p->width;

18

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

20 if(\*arr\_p==NULL){

21 fprintf(stderr, "Too little memory.\n");

22 return 0;

23 }

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

25 (\*arr\_p)[i] = (Rgb\*)malloc(W \* sizeof(Rgb) + (W\*3)%4);

26 if((\*arr\_p)[i] == NULL){

27 for(int j=0; j<i; j++)

28 free((\*arr\_p)[j]);

29 free(\*arr\_p);

30 fprintf(stderr, "Too little memory.\n");

31 return 0;

32 }

33 fread((\*arr\_p)[i],1,W \* sizeof(Rgb) + (W\*3)%4,f);

34 }

35 fclose(f);

36 return 1;

37 }

38

39 void memoClean(Rgb\*\* arr, int H, char\* filename, char\* alternative){

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

41 if(arr[i]!=NULL)

42 free(arr[i]);

43 }

44 if(arr!=NULL)

45 free(arr);

46 if(alternative!=NULL)

47 free(alternative);

48 if(filename!=NULL)

49 free(filename);

50

51 }

52

53 int readOperands(int argc, int optind, char\*\* argv, int\* operands, int operand\_count){

54 // returns optind

55 int index = optind - 1;

56 int i=0;

57 //char\* next = argv[index];

58 while(i < operand\_count && index < argc){// && next[0] != '-'){

59 //next = argv[index];

60 operands[i++] = atoi(argv[index]);

61 index++;

62 }

63 if(index < argc){

64 return index-1;

65 }

66 else{

67 return index;

68 }

69 }

./src/checker.hpp

1 #pragma once

2

3 #include "structures.hpp"

4

5 int typeCheck(BitmapFileHeader\* bmfh, BitmapInfoHeader\* bmih);

6

7 int operandCheck(int flag, BitmapInfoHeader\* bmih, int\* start, int\* end, int\* to, int\* old\_color, int\* new\_color);

./src/checker.cpp

1 #include "checker.hpp"

2

3 int typeCheck(BitmapFileHeader\* bmfh, BitmapInfoHeader\* bmih){

4 if((bmfh->signature == 19778) && (bmih->bitsPerPixel == 24) && (bmih->compression == 0) &&

5 (bmih->colorsInColorTable == 0) && (bmih->importantColorCount == 0)){

6 return 1;

7 }else{

8 return 0;

9 }

10 }

11

12 int operandCheck(int flag, BitmapInfoHeader\* bmih, int\* start, int\* end, int\* to, int\* old\_color, int\* new\_color){

13 switch (flag){

14 case 'c':

15 if(start[0] < 0 || start[0] >= bmih->width || end[0] < 0 || end[0] >= bmih->width || to[0] < 0 || to[0] >= bmih->width)

16 return 0;

17 if(start[1] < 0 || start[1] >= bmih->height || end[1] < 0 || end[1] >= bmih->height || to[1] < 0 || to[1] >= bmih->height)

18 return 0;

19 break;

20

21 case 'x':

22 if(start[0] < 0 || start[0] >= bmih->width || end[0] < 0 || end[0] >= bmih->width)

23 return 0;

24 if(start[1] < 0 || start[1] >= bmih->height || end[1] < 0 || end[1] >= bmih->height)

25 return 0;

26 break;

27

28 case 'y':

29 if(start[0] < 0 || start[0] >= bmih->width || end[0] < 0 || end[0] >= bmih->width)

30 return 0;

31 if(start[1] < 0 || start[1] >= bmih->height || end[1] < 0 || end[1] >= bmih->height)

32 return 0;

33 break;

34

35 case 'r':

36 for(int i=0; i<3; i++){

37 if(old\_color[i] < 0 || old\_color[i] > 255 || new\_color[i] < 0 || new\_color[i] > 255)

38 return 0;

39 }

40 break;

41 }

42 return 1;

43 }

./src/printInfoHelp.hpp

1 #pragma once

2

3 #include "structures.hpp"

4

5 void printInfo(BitmapFileHeader bmfh, BitmapInfoHeader bmih);

6

7 void printHelp();

./src/printInfoHelp.cpp

1 #include "printInfoHelp.hpp"

2

3 void printInfo(BitmapFileHeader bmfh, BitmapInfoHeader bmih){

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

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

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

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

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

9

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

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

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

13 printf("planes: \t%x (%hu)\n", bmih.planes, bmih.planes);

14 printf("bitsPerPixel:\t%x (%hu)\n", bmih.bitsPerPixel, bmih.bitsPerPixel);

15 printf("compression:\t%x (%u)\n", bmih.compression, bmih.compression);

16 printf("imageSize:\t%x (%u)\n", bmih.imageSize, bmih.imageSize);

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

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

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

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

21 }

22

23 void printHelp(){

24 printf("\033[1;96m\nAbout:\033[0m\n");

25 printf("\033[96m'bimped' utility is meant for BMP picture editing and only supports files with\n"

26 "no compression, 24 bits per pixel, no colors in color table and 0 important color count.\033[0m\n");

27 printf("\033[1m\nUsage:\033[0m\n");

28 printf(" bimped <option> [argument]\n\n");

29 printf("\033[1m\nOptions:\033[0m\n");

30 printf(" -h, --help \tShow help.\n");

31 printf(" -f <path>, --file <path> \tDefine original file. By default original file\n"

32 "\t\t\t\tis the last argument.\n");

33 printf(" -a <path>, --alter <path> \tUse this key to save to a different file.\n");

34 printf(" -i <path>, --info <path> \tShow information about the file.\n");

35 printf(" -c, --copy-patch \tCopies and pastes a rectangular fragment to a destination\n"

36 "\t\t\t\tin the same picture.\n"

37 "\t\t\t\tOnly used with \033[1m-s(--start)\033[0m and \033[1m-e(--end)\033[0m keys\n"

38 "\t\t\t\tthe upper left (-s) and the bottom right (-e) corners\n"

39 "\t\t\t\tof the fragment and \033[1m-t(--to)\033[0m key to define\n"

40 "\t\t\t\tthe upper left corner of the destination.\n");

41 printf(" -x, --reflect-x \tReflects a fragment by X axis.\n"

42 "\t\t\t\tOnly used with \033[1m-s(--start)\033[0m and \033[1m-e(--end)\033[0m keys to define\n"

43 "\t\t\t\tthe upper left (-s) and the bottom right (-e) corners\n"

44 "\t\t\t\tof the fragment.\n");

45 printf(" -y, --reflect-y \tReflects a fragment by Y axis.\n"

46 "\t\t\t\tOnly used with \033[1m-s(--start)\033[0m and \033[1m-e(--end)\033[0m keys to define\n"

47 "\t\t\t\tthe upper left (-s) and the bottom right (-e) corners\n"

48 "\t\t\t\tof the fragment.\n");

49 printf(" -r, --replace-color \tReplaces old color and colors close to it with new color.\n"

50 "\t\t\t\tOnly used with \033[1m-o(--old)\033[0m and \033[1m-n(--new)\033[0m keys to define\n"

51 "\t\t\t\told and new colors using RGB palette.\n");

52 printf(" -d, --divide <i> <j> \tDivivdes a picture into i pieces by X axis and j pieces\n"

53 "\t\t\t\tby Y axis. Stores all resulting fragments in the same\n"

54 "\t\t\t\tdirectory if \033[1m-a(--alter)\033[0m key is not defined.\n");

55 printf(" -s, --start <x> <y> \tCoordinates of the upper left corner.\n"

56 "\t\t\t\tDefault: 0, 0.\n");

57 printf(" -e, --end <x> <y> \tCoordinates of the bottom right corner.\n"

58 "\t\t\t\tDefault: 0, 0.\n");

59 printf(" -t, --to <x> <y> \tCoordinates of the upper left corner.\n"

60 "\t\t\t\tDefault: 0, 0.\n");

61 printf(" -o, --old <R> <G> <B> \tColor to be replaced. Range 0-255.\n");

62 printf(" -n, --new <R> <G> <B> \tNew color. Range 0-255.\n\n\n");

63 }

./src/saveFile.hpp

1 #pragma once

2

3 #include "structures.hpp"

4

5 int saveFile(const char\* filename, Rgb\*\* arr, BitmapFileHeader\* bmfh\_p, BitmapInfoHeader\* bmih\_p, int coordY, int coordX);

./src/saveFile.cpp

1 #include "saveFile.hpp"

2

3 int saveFile(const char\* filename, Rgb\*\* arr, BitmapFileHeader\* bmfh\_p, BitmapInfoHeader\* bmih\_p, int coordY, int coordX){

4 FILE \*ff = fopen(filename, "wb");

5 if(ff==NULL){

6 printf("Trouble with creating output file.\n");

7 return 0;

8 }

9 unsigned int w = bmih\_p->width \* sizeof(Rgb);

10 unsigned int w\_buff = (w\*3)%4;

11 char arr\_buff[3] = {0,0,0};

12

13 fwrite(bmfh\_p, 1, sizeof(BitmapFileHeader),ff);

14 fwrite(bmih\_p, 1, sizeof(BitmapInfoHeader),ff);

15

16 for(int i=coordY; i < coordY + bmih\_p->height; i++){

17 fwrite(arr[i]+coordX,1,w,ff);

18 fwrite(arr\_buff,1,w\_buff,ff);

19 }

20

21 fclose(ff);

22 return 1;

23 }

./src/tasks.hpp

1 #pragma once

2

3 #include "structures.hpp"

4

5 int divideFile(Rgb\*\* arr, BitmapFileHeader\* bmfh\_p, BitmapInfoHeader\* bmih\_p, int countH, int countW, char\* prefix);

6

7 void replaceColor(Rgb\*\* arr, BitmapInfoHeader\* bmih, int\* old\_color, int\* new\_color);

8

9 int reflectFragmentX(Rgb\*\* arr, int\* left, int\* right);

10

11 int reflectFragmentY(Rgb\*\* arr, int\* left, int\* right);

12

13 int copyPatch(Rgb\*\* arr, BitmapInfoHeader\* bmih\_p, int\* left, int\* right, int\* to);

./src/tasks.cpp

1 #include "tasks.hpp"

2 #include "saveFile.hpp"

3

4 int divideFile(Rgb\*\* arr, BitmapFileHeader\* bmfh\_p, BitmapInfoHeader\* bmih\_p, int countH, int countW, char\* prefix){

5 if(countH < 1 || countH > bmih\_p->height || countW < 1 || countW > bmih\_p->width){

6 return 0;

7 }

8 int chunkHeight = bmih\_p->height / countH;

9 int chunkWidth = bmih\_p->width / countW;

10

11 int old\_height = bmih\_p->height;

12 int old\_width = bmih\_p->width;

13

14 bmih\_p->height = chunkHeight;

15 bmih\_p->width = chunkWidth;

16 bmfh\_p->filesize = chunkHeight \* chunkWidth \* sizeof(Rgb);

17 bmih\_p->imageSize = chunkHeight \* chunkWidth \* sizeof(Rgb);

18

19 int number = 0;

20 char s[100];

21 char newFileName[strlen(prefix)+100];

22 for(int i=0; i < countH; i++){

23 for(int j=0; j < countW; j++){

24 strcpy(newFileName, prefix);

25 sprintf(s, "%d", countH-i);

26 strcat(newFileName, s);

27 strcat(newFileName, ".");

28 sprintf(s, "%d", j+1);

29

30 strcat(newFileName, s);

31 strcat(newFileName, ".bmp");

32

33 if(!saveFile(newFileName, arr, bmfh\_p, bmih\_p, i\*chunkHeight, j\*chunkWidth))

34 return 0;

35 }

36 }

37

38 // back to old parameters

39 bmih\_p->height = old\_height;

40 bmih\_p->width = old\_width;

41 bmfh\_p->filesize = old\_height \* old\_width \* sizeof(Rgb);

42 bmih\_p->imageSize = old\_height \* old\_width \* sizeof(Rgb);

43

44 return 1;

45 }

46

47

48 void replaceColor(Rgb\*\* arr, BitmapInfoHeader\* bmih, int\* old\_color, int\* new\_color){

49 for(int i = 0; i < bmih->height; i++){

50 for(int j = 0; j < bmih->width; j++){

51 if((abs(arr[i][j].b - old\_color[2]) < 50)&&(abs(arr[i][j].g - old\_color[1]) < 50)&&(abs(arr[i][j].r - old\_color[0]) < 50)){

52 arr[i][j] = {(unsigned char)new\_color[2], (unsigned char)new\_color[1], (unsigned char)new\_color[0]};

53 }

54 }

55 }

56 }

57

58

59 int reflectFragmentX(Rgb\*\* arr, int\* left, int\* right){

60 int fr\_width = (right[0] - left[0])\* sizeof(Rgb);

61 int fr\_height = (left[1] - right[1])/2;

62 if(fr\_width<=0 || fr\_height<=0){

63 //printf("Invalid coordinates in --reflect-x.\n");

64 return 0;

65 }

66 Rgb \*\*half = (Rgb\*\*)malloc(fr\_height \* sizeof(Rgb\*));

67 int count = 0;

68 // copy upper half to memory

69 for(int i = left[1]; i > left[1]-fr\_height; i--){

70 count++;

71 half[fr\_height-count] = (Rgb\*)malloc(fr\_width \* sizeof(Rgb));

72 memmove(half[fr\_height-count], arr[i]+left[0], fr\_width);

73 }

74 // copy lower half and insert it into upper half upside down

75 count = 0;

76 for(int i = right[1]; i < right[1] + fr\_height; i++){

77 memmove(arr[left[1] - count] + left[0], arr[i] + left[0], fr\_width);

78 count++;

79 }

80 // insert upper half from memo into lower half upside down

81 count = 0;

82 for(int i = right[1]; i < right[1] + fr\_height; i++){

83 count++;

84 memmove(arr[i] + left[0], half[fr\_height - count], fr\_width);

85 }

86 for(int i = 0; i < fr\_height; i++) {

87 free(half[i]);

88 }

89 if(half){

90 free(half);

91 }

92 return 1;

93 }

94

95

96 int reflectFragmentY(Rgb\*\* arr, int\* left, int\* right){

97 int fr\_width = (right[0] - left[0])/2;

98 int fr\_height = (left[1] - right[1]);

99 if(fr\_width<=0 || fr\_height<=0){

100 //printf("Invalid coordinates in --reflect-y.\n");

101 return 0;

102 }

103 Rgb temp;

104 for(int i = right[1]; i <= left[1]; i++){

105 for(int j = 0; j <= fr\_width; j++){

106 temp = arr[i][left[0] + j];

107 arr[i][left[0] + j] = arr[i][right[0] - j];

108 arr[i][right[0] - j] = temp;

109 }

110 }

111 return 1;

112 }

113

114

115 int copyPatch(Rgb\*\* arr, BitmapInfoHeader\* bmih\_p, int\* left, int\* right, int\* to){

116 int patch\_width = right[0] - left[0];

117 int patch\_height = left[1] - right[1];

118 if(patch\_width <= 0 || patch\_height <= 0){

119 //printf("Invalid coordinates in --copy-patch.\n");

120 return 0;

121 }

122 if(to[1] - patch\_height < 0){

123 patch\_height = to[1];

124 }

125 if(to[0] + patch\_width > bmih\_p->width){

126 patch\_width = bmih\_p->width - to[0];

127 }

128 patch\_width \*= sizeof(Rgb);

129 int j = 0;

130 for(int i = left[1]; i >= left[1] - patch\_height; i--){

131 memmove(arr[to[1] - j] + to[0], arr[i]+left[0], patch\_width);

132 j++;

133 }

134 return 1;

135 }

Комментарии из пулл-реквестов

Исходный код:

void printInfo(BitmapFileHeader bmfh, BitmapInfoHeader bmih){  
  
#include "printInfo.hpp"

Комментарии:

pro100kot:`printInfo` и `printHelp` нет смысла выносить в различные файлы. Логически эти функции сильно связаны

Исходный код:

void replaceColor(Rgb\*\* arr, BitmapInfoHeader\* bmih, int\* old\_color, int\* new\_color){  
  
  
#include "replaceColor.hpp"

Комментарии:

pro100kot:Функции обработки изображения есть смысл также сгруппировать в одном файле

Исходный код:

int saveFile(const char\* filename, Rgb\*\* arr, BitmapFileHeader\* bmfh\_p,  
  
#include "structures.hpp"  
  
#pragma once

Комментарии:

pro100kot:Точно ли требуется передавать функции ` int H, int W` ?   
  
Функция имеет и без того слишком большое количество аргументов

Исходный код:

Rgb \*\*half = (Rgb\*\*)malloc(fr\_height \* sizeof(Rgb\*));  
 }  
 return 0;  
 //printf("Invalid coordinates in --reflect-x.\n");  
 if(fr\_width<=0 || fr\_height<=0){  
 int fr\_height = (left[1] - right[1])/2;  
 int fr\_width = (right[0] - left[0])\* sizeof(Rgb);  
int reflectFragmentX(Rgb\*\* arr, int\* left, int\* right){  
  
#include "reflect.hpp"

Комментарии:

pro100kot:Вы не освобождаете память под этот массив.

Исходный код:

int main(int argc, char \*\*argv){  
  
  
  
//using namespace std;  
  
  
  
#include "argumentHandler.hpp"  
  
#include "checker.hpp"  
  
#include "divideFile.hpp"  
  
#include "saveFile.hpp"  
  
#include "replaceColor.hpp"  
  
#include "reflect.hpp"  
  
#include "copyPatch.hpp"  
  
#include "printInfo.hpp"  
  
#include "structures.hpp"  
  
#include "printHelp.hpp"  
  
#include <getopt.h>  
  
#include <cstring>  
  
#include <cstdlib>  
  
#include <iostream>

Комментарии:

pro100kot:Разделите main на несколько функций. Сейчас она слишком огромная.

Изменения: