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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Работа с файлами формата PNG.

Студент гр. 0381 _____ Кирильцев Д.А.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург
2020

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Кирильцев Д.А.

Группа 0381

Тема работы: Работа с файлами формата PNG.

Исходные данные:

Программа **должна** иметь CLI или GUI.

Программа должна реализовывать следующий функционал по обработке PNG-файла:

1. Разделяет изображение на N*M частей. Реализация: провести линии заданной толщины, тем самым разделив изображение.
2. Рисование прямоугольника.
3. Сделать рамку в виде узора.
4. Поворот изображения (части) на 90/180/270 градусов.

Содержание пояснительной записки:

Аннотация, содержание, введение, описание архитектуры программы, вывод результата, примеры работы.

Предполагаемый объем пояснительной записки:

Не менее 13 страниц.

Дата выдачи задания: 05.04.2021

Дата сдачи реферата: 28.05.2021

Дата защиты реферата: 28.05.2021

Студент _____ Кирильцев Д.А.

Преподаватель _____ Берленко Т.А.

1.АННОТАЦИЯ

В курсовой работе были реализованы функции по работе с изображениями в формате PNG, такие как: рисование прямоугольника, разбиение области на n * m частей, отображение рамки, поворот области или всего изображения на угол 90/180/270. Всё это предоставляется пользователю в виде консольного приложения. Для написания CLI был использован Getopt. Исходный код написан на языке C.

В результате было разработано консольное приложение, запрашивающее у пользователя файл формата PNG, после чего предоставляющее ему возможность изменять файл, и сохранять результат.

2.SUMMARY

In this course work, functions with images in PNG format are implemented, namely: drawing a rectangle, drawing a grid with n * m fractions, drawing a frame, turning an area with an angle of rotation with 90/180/270. All this is presented to the user in the form of an application with a command-line interface. The Getopt framework was used to write the CLI.

The source code has written in the C language. As a result, an application was developed that prompts the user for a PNG file and allows him to save changes.

СОДЕРЖАНИЕ

1.	Аннотация	3
2.	Summary	3
3.	Введение	5
4.	Краткое описание кода	
4.1.	Структуры, функции их обработки	6
4.2.	Основные функции	7
5.	Инструкция по использованию	8
6.	Проверка работы программы	10
7.	Заключение	13
	Приложение А.	14

3. ВВЕДЕНИЕ

Цель работы:

Научиться работать файлами в формата PNG, реализовать функционал, описанный в условии.

Задачи работы:

1. Считывание изображение в структуру Img. Программа работает только с изображениями формата PNG, в которых используется RGBA.

2. Реализация интерфейса в командной строке в виде CLI для обработки команд пользователя.

3. Реализация подзадач (решение каждой подзадачи вынесено в отдельную функцию):

· Рисование прямоугольника с последующей его заливкой, если выбрана соответствующая опция: по заданным координатам левого верхнего угла и правого нижнего угла, также задается цвет линии, её толщина и цвет залитой области.

· Разбиение изображения на n*m кусочков, с введением информации о делении оси x и оси y, также толщины линий деления и её цвета.

· Рисование рамки в виде узора с выбором цвета рамки.

· Поворот изображения или его области на угол 90/180/270.

4.1. **СТРУКТУРЫ, ФУНКЦИИ И ИХ ОБРАБОТКИ**

1. *struct Img;* - Структура, которая хранит информацию о PNG файле.

2. *struct Configs;* - Структура, которая хранит информацию о ключах, введенных пользователем.

3. *void PNG_info(struct Img* img);* - Функция, которая выводит на экран информацию о выбранном PNG файле.

4. *int rfile(struct Img* img, char* fname);* - Функция заполнения структуры *Img*.

5. *int wfile(struct Img* img, char* fname);* - Функция заполнения записи информации из структуры *Img* в файл.

6. *struct option longOpts[] = {};* -Структура, хранящая в себе длинные названия команд и их короткие ключи для CLI.

4.2. ФУНКЦИИ ДЛЯ РЕАЛИЗАЦИИ ЗАДАНИЙ ПОЛЬЗОВАТЕЛЯ

1. int dgrid(struct Img *img, int x_partition, int y_partition,int thick, int red, int green, int blue, int alpha); - функция пробегается по массиву пикселей и рисует линии выбранным цветом и заданной толщины, в соответствии с указанием пользователя по разделению изображения.

2. int drectangle(struct Img *img, int x1, int y1, int x2, int y2, int thick, int filled, int red, int green, int blue, int alpha, int d_red, int d_green, int d_blue, int d_alpha); - функция пробегается по массиву пикселей и рисует прямоугольник цвета заливки (если выбрано) в определенной области, после этого рисуется окантовка заданного цвета и толщины.

3. int dborder(struct Img *img, int option, int thick, int red, int green, int blue, int alpha); - функция по заданной пользователем ширине нарисует один из трех узоров на выбор с помощью арифметики остатков.

4. int rotate(struct Img* img, int x1, int y1, int x2, int y2, int angle); - функция с помощью перестановок пикселей оригинального массива поворачивает изображение, переставленные пиксели копируются в новый массив, тем самым создавая новое изображение, а результат сохраняется в файл.

5. **Инструкция по использованию**

Для начала работы пользователь должен установить библиотеку libpng. Чтобы это сделать пользователь должен ввести у себя в терминале команду:
sudo apt-get install libpng-dev

После успешной установки пользователь компилирует проект у себя на компьютере в директории с исходным кодом, путем введения команды:
make

После успешной компиляции пользователь вводит команду:
./app —keys —input file_name.PNG (--result file_name.PNG)

Вместо *—keys* пользователь вводит необходимые ему ключи модификации.

Описание ключей модификации:

-f, --fill - Заливка.

-c, --color <red_channel_value>,<green_channel_value>,<blue_channel_value>,<alpha_channel_value> : 0...255 для каждого цветового канала — цвет линии.

-t, --thickness <value> : 0...25 толщина линии или ширина рамки. (по умолчанию стоит 1).

-a, --angle <value> : 90, 180, 270 — угол поворота.

-o, --option <value> : 1..3 — вариант рамки. (по умолчанию стоит 1).

-s, --start <x_value>,<y_value> - координаты верхнего левого угла.

-e, --end <x_value>,<y_value> - координаты нижнего правого угла.

-p, --partition <x_partition>,<y_partition> : 1...20 для каждого — деление по осям.

-m, --manipulation <value> : 1...4 — выбор обработки изображения.
(1 — деление изображения на части;
2 — рисование прямоугольника;
3 — рисование рамки;
4 — поворот области.)

-b, --bgcolor
<red_channel_value>,<green_channel_value>,<blue_channel_value>,<alpha_channel_value> : 0...255 для каждого цветового канала — цвет заливки.

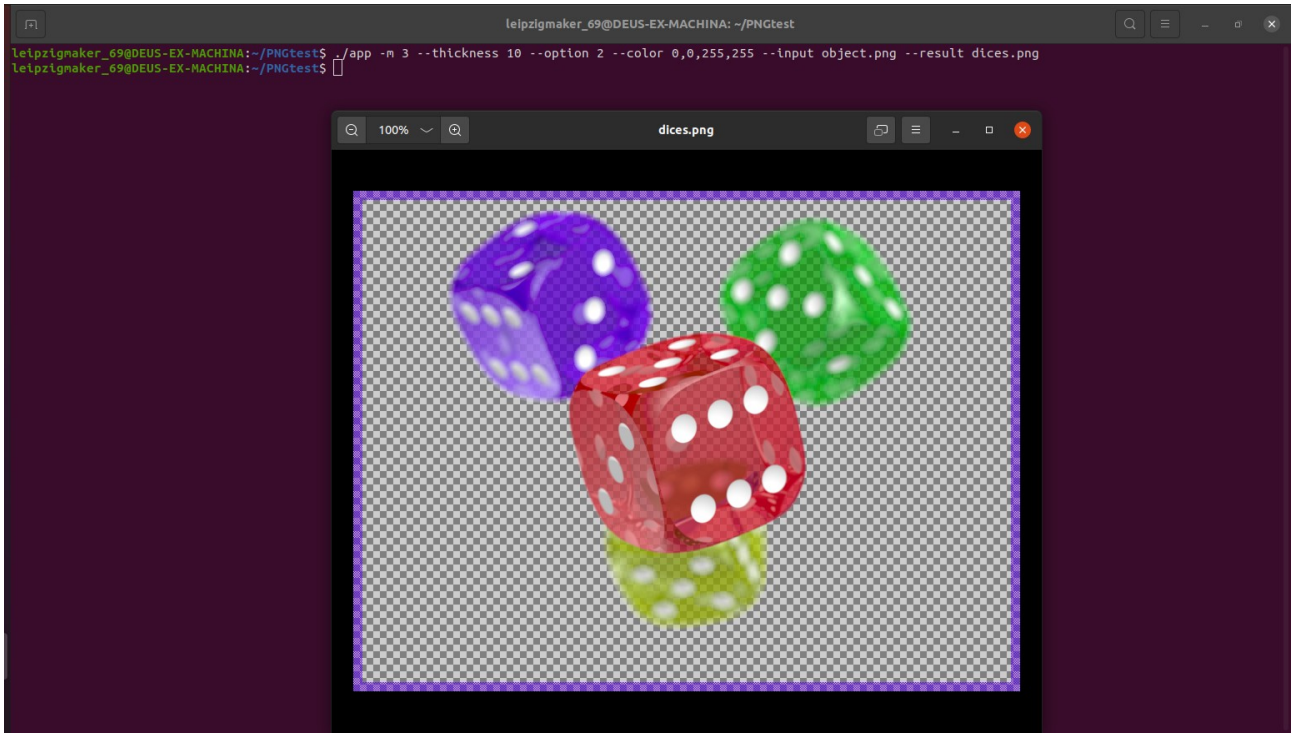
-r, --result <file_name> Имя выходного файла.

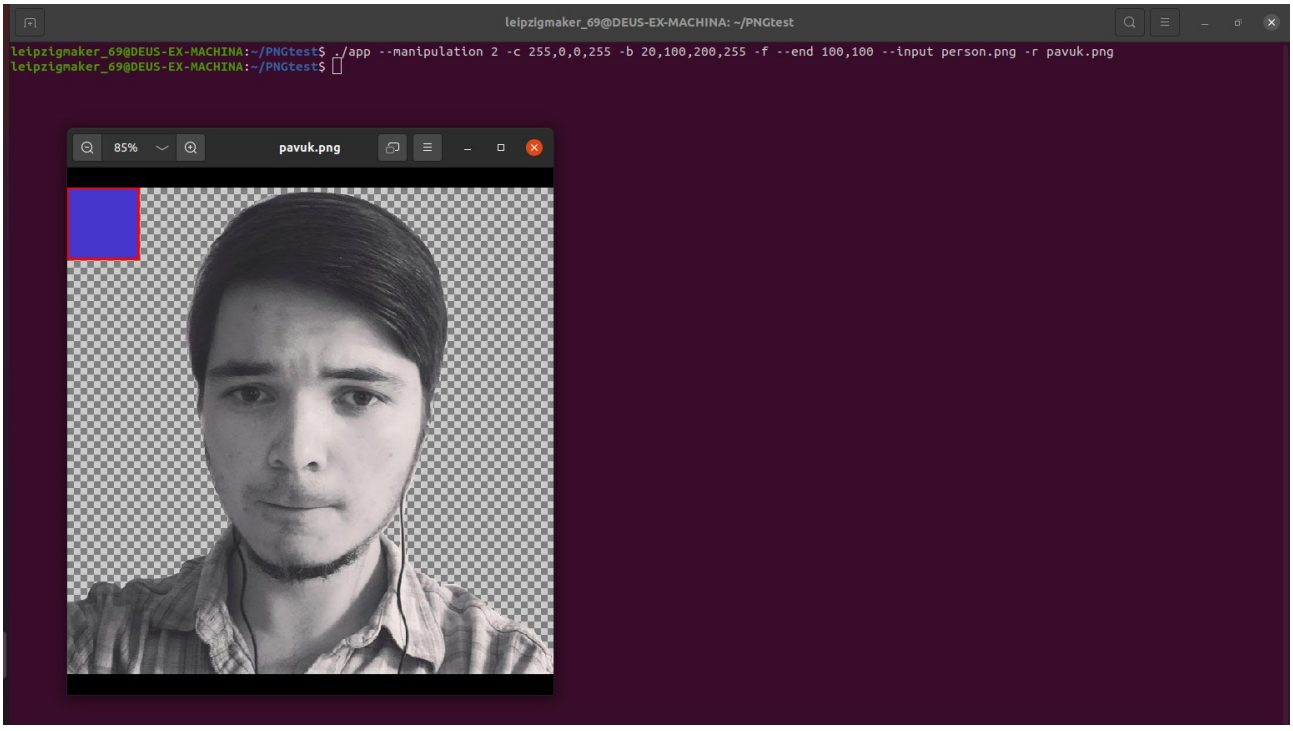
-i, --input <input_name> Имя входного файла.

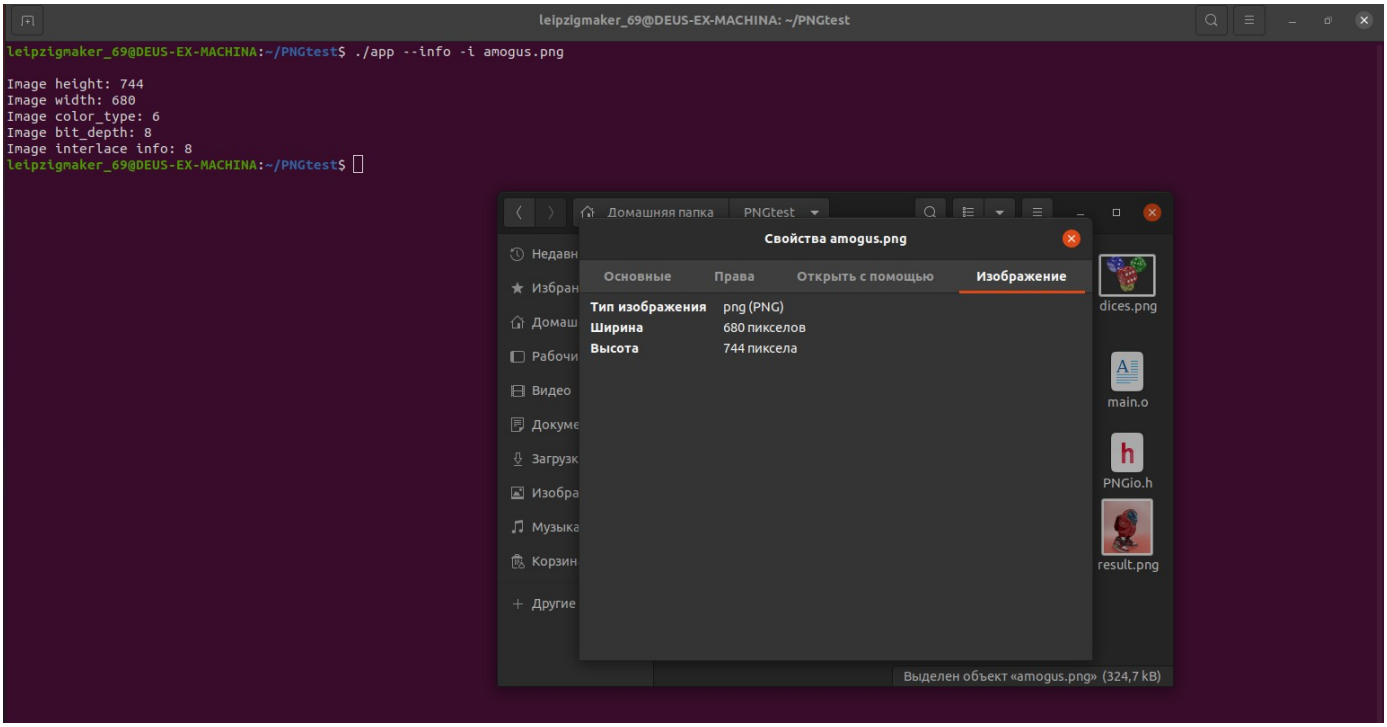
-n, --info — вызов информации об изображении.

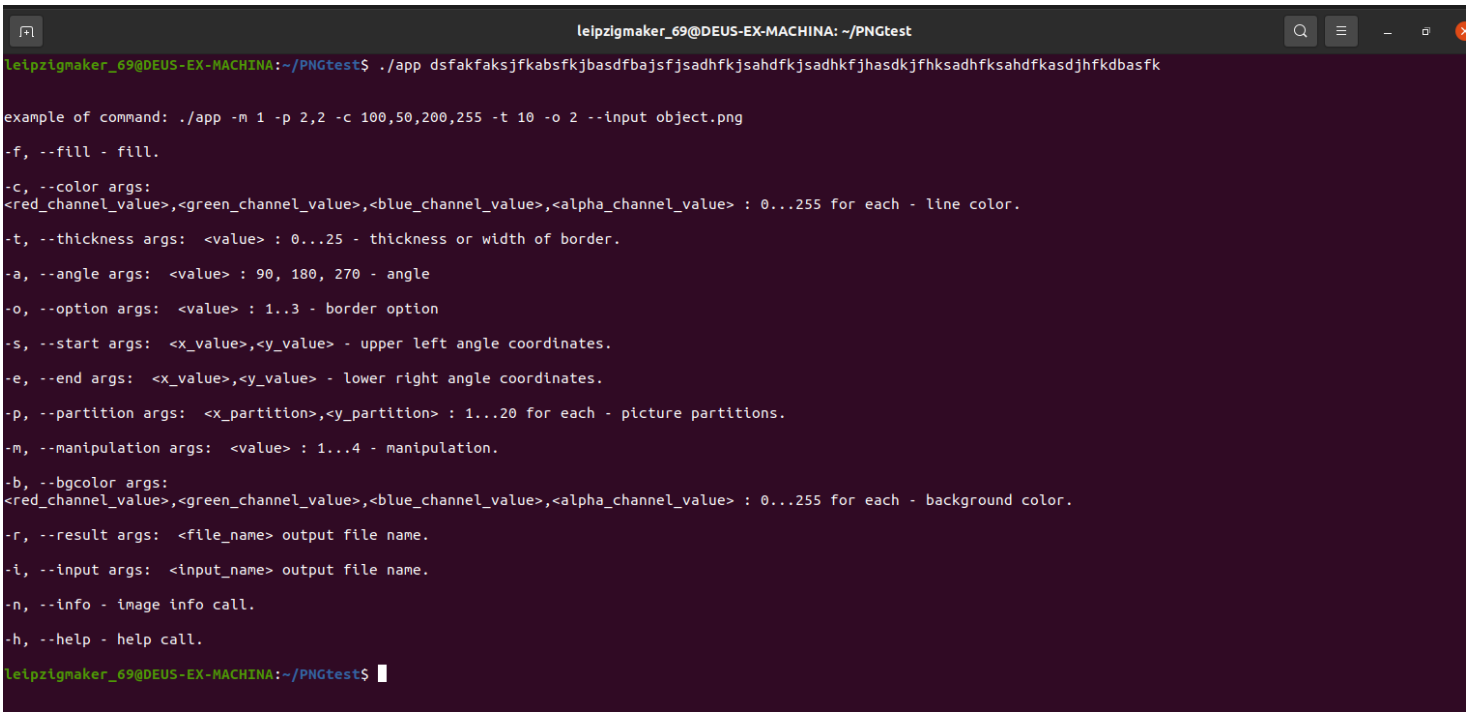
-h, --help — вызов помощи.

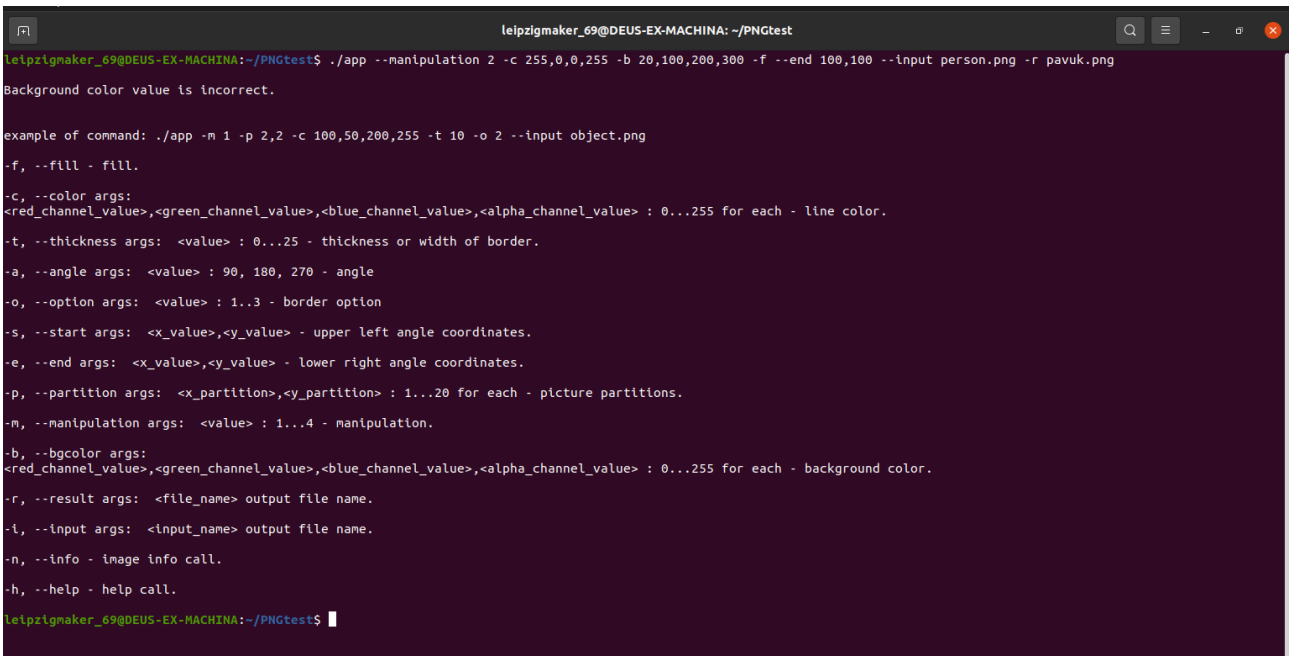
6. ПРОВЕРКА РАБОТЫ ПРОГРАММЫ

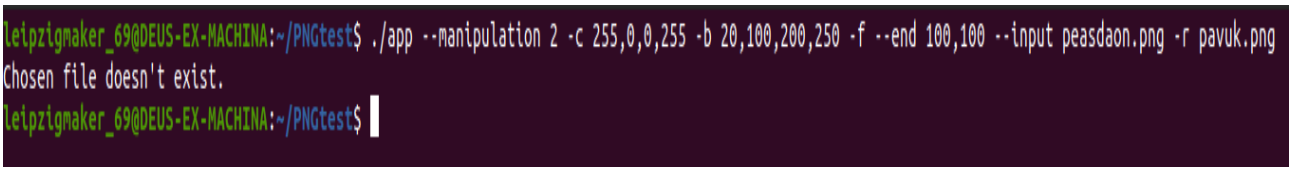





leipzigmaker_69@DEUS-EX-MACHINA: ~/PNGtest
leipzigmaker_69@DEUS-EX-MACHINA:~/PNGtest$ ./app --info -i amogus.png
Image height: 744
Image width: 680
Image color_type: 6
Image bit_depth: 8
Image interlace info: 8
leipzigmaker_69@DEUS-EX-MACHINA:~/PNGtest$
Домашняя папка
PNGtest
Свойства amogus.png
Основные
Права
Открыть с помощью
Изображение
Тип изображения
png (PNG)
Ширина
680 пикселов
Высота
744 пиксела
dices.png
main.o
PNGio.h
result.png
Выделен объект «amogus.png» (324,7 kB)


leipzigmaker_69@DEUS-EX-MACHINA: ~/PNGtest
leipzigmaker_69@DEUS-EX-MACHINA:~/PNGtest$ ./app dsfakfaksjfkabsfkjbasdfbajsfjsadhfkjsahdfkjsadhkfjhasdkjfhksadhfksahdfkasdjhfkdbasfk
example of command: ./app -m 1 -p 2,2 -c 100,50,200,255 -t 10 -o 2 --input object.png
-f, --fill - fill.
-c, --color args:
<red_channel_value>,<green_channel_value>,<blue_channel_value>,<alpha_channel_value> : 0...255 for each - line color.
-t, --thickness args: <value> : 0...25 - thickness or width of border.
-a, --angle args: <value> : 90, 180, 270 - angle
-o, --option args: <value> : 1..3 - border option
-s, --start args: <x_value>,<y_value> - upper left angle coordinates.
-e, --end args: <x_value>,<y_value> - lower right angle coordinates.
-p, --partition args: <x_partition>,<y_partition> : 1...20 for each - picture partitions.
-m, --manipulation args: <value> : 1...4 - manipulation.
-b, --bgcolor args:
<red_channel_value>,<green_channel_value>,<blue_channel_value>,<alpha_channel_value> : 0...255 for each - background color.
-r, --result args: <file_name> output file name.
-i, --input args: <input_name> output file name.
-n, --info - image info call.
-h, --help - help call.
leipzigmaker_69@DEUS-EX-MACHINA:~/PNGtest$


leipzigmaker_69@DEUS-EX-MACHINA: ~/PNGtest
leipzigmaker_69@DEUS-EX-MACHINA:~/PNGtest$ ./app --manipulation 2 -c 255,0,0,255 -b 20,100,200,300 -f --end 100,100 --input person.png -r pavuk.png
Background color value is incorrect.
example of command: ./app -m 1 -p 2,2 -c 100,50,200,255 -t 10 -o 2 --input object.png
-f, --fill - fill.
-c, --color args:
<red_channel_value>,<green_channel_value>,<blue_channel_value>,<alpha_channel_value> : 0...255 for each - line color.
-t, --thickness args: <value> : 0...25 - thickness or width of border.
-a, --angle args: <value> : 90, 180, 270 - angle
-o, --option args: <value> : 1..3 - border option
-s, --start args: <x_value>,<y_value> - upper left angle coordinates.
-e, --end args: <x_value>,<y_value> - lower right angle coordinates.
-p, --partition args: <x_partition>,<y_partition> : 1...20 for each - picture partitions.
-m, --manipulation args: <value> : 1...4 - manipulation.
-b, --bgcolor args:
<red_channel_value>,<green_channel_value>,<blue_channel_value>,<alpha_channel_value> : 0...255 for each - background color.
-r, --result args: <file_name> output file name.
-i, --input args: <input_name> output file name.
-n, --info - image info call.
-h, --help - help call.
leipzigmaker_69@DEUS-EX-MACHINA:~/PNGtest$


leipzigmaker_69@DEUS-EX-MACHINA:~/PNGtest$ ./app --manipulation 2 -c 255,0,0,255 -b 20,100,200,250 -f --end 100,100 --input peasdaon.png -r pavuk.png
Chosen file doesn't exist.
leipzigmaker_69@DEUS-EX-MACHINA:~/PNGtest$

ЗАКЛЮЧЕНИЕ

В результате работы было реализовано считывание PNG-файла в структуры и динамическую память, созданы различные функции для обработки пикселей, интерфейс в виде CLI для взаимодействия с пользователем и получена программа, работающая на Linux.

ПРИЛОЖЕНИЕ А
НАЗВАНИЕ ПРИЛОЖЕНИЯ

MAKEFILE

```
ALL: APP

APP: MAIN.O PNGIO.O RENDER.O
	GCC MAIN.O PNGIO.O RENDER.O -O APP -LPNG

MAIN.O: MAIN.C PNGIO.H RENDER.H STRUCTS.H
	GCC -C MAIN.C -LPNG

RENDER.O: RENDER.C STRUCTS.H
	GCC -C RENDER.C -LPNG

PNGIO.O: PNGIO.C STRUCTS.H
	GCC -C PNGIO.C -LPNG

CLEAN:
	RM -RF *.O APP
```

STRUCTS.H

```
STRUCT CONFIGS{
	INT FILLED;
	INT RED;
	INT GREEN;
	INT BLUE;
	INT ALPHA;
	INT THICK;
```

```
        INT ANGLE;
        INT OPTION;
        INT START_X;
        INT START_Y;
        INT END_X;
        INT END_Y;
        INT X_PARTITION;
        INT Y_PARTITION;
        INT MANIPULATION;
        INT D_RED;
        INT D_GREEN;
        INT D_BLUE;
        INT D_ALPHA;
        CHAR* INPUT;
        CHAR* RESULT;
};


STRUCT IMG{
        INT WIDTH, HEIGHT;
        PNG_BYTE COLOR_TYPE;
        PNG_BYTE BIT_DEPTH;

        PNG_STRUCTP PNG_PTR;
        PNG_INFOP INFO_PTR;
        INT NOP;
        PNG_BYTEP *ROW_PTS;
};
```

MAIN.C

```
#INCLUDE <STDIO.H>
#INCLUDE <STDLIB.H>
#INCLUDE <PNG.H>
#INCLUDE <UNISTD.H>
#INCLUDE <STRING.H>
#INCLUDE <GETOPT.H>
#INCLUDE "STRUCTS.H"
#INCLUDE "PNGIO.H"
#INCLUDE "RENDER.H"

VOID PNG_INFO(STRUCT IMG* IMG){
        PRINTF("\NIMAGE HEIGHT: %D\N", IMG->HEIGHT);
        PRINTF("IMAGE WIDTH: %D\N", IMG->WIDTH);
        PRINTF("IMAGE COLOR_TYPE: %HHU\N", IMG->COLOR_TYPE);
        PRINTF("IMAGE BIT_DEPTH: %HHU\N", IMG->BIT_DEPTH);
        PRINTF("IMAGE INTERLACE INFO: %D\N", IMG->BIT_DEPTH);
}

CHAR* GET_FILE_NAME(CHAR *LINE){

   INT SIZE = 1;
   CHAR* FNAME = (CHAR *)MALLOC(SIZE);
   INT I = 0;

   WHILE (I - 1 != STRLEN(LINE) + 1){
      IF (I + 1 > SIZE){
         SIZE++;
         FNAME = (CHAR *)REALLOC(FNAME, SIZE);
      }
      FNAME[I++] = LINE[I];
```

```
    }
    FNAME[I - 1] = '\0';

    RETURN FNAME;
}

VOID PRINTHELP(){
        PRINTF("\N\NEXAMPLE OF COMMAND: ./APP -M 1 -P 2,2 -C
100,50,200,255 -T 10 -O 2 --INPUT OBJECT.PNG\N\N");
        PRINTF("-F, --FILL - FILL.\N\N");
        PRINTF("-C, --COLOR ARGS: \
N<RED_CHANNEL_VALUE>,<GREEN_CHANNEL_VALUE>,<BLUE_CHA
NNEL_VALUE>,<ALPHA_CHANNEL_VALUE> : 0...255 FOR EACH - LINE
COLOR.\N\N");
        PRINTF("-T, --THICKNESS ARGS:  <VALUE> : 0...25 - THICKNESS
OR WIDTH OF BORDER.\N\N");
        PRINTF("-A, --ANGLE ARGS:  <VALUE> : 90, 180, 270 - ANGLE\N\
N");
        PRINTF("-O, --OPTION ARGS:  <VALUE> : 1..3 - BORDER OPTION\
N\N");
        PRINTF("-S, --START ARGS:  <X_VALUE>,<Y_VALUE> - UPPER
LEFT ANGLE COORDINATES.\N\N");
        PRINTF("-E, --END ARGS:  <X_VALUE>,<Y_VALUE> - LOWER
RIGHT ANGLE COORDINATES.\N\N");
        PRINTF("-P, --PARTITION ARGS:
<X_PARTITION>,<Y_PARTITION> : 1...20 FOR EACH - PICTURE
PARTITIONS.\N\N");
        PRINTF("-M, --MANIPULATION ARGS:  <VALUE> : 1...4 -
MANIPULATION.\N\N");
```

```
        PRINTF("-B, --BGCOLOR ARGS:  \
N<RED_CHANNEL_VALUE>,<GREEN_CHANNEL_VALUE>,<BLUE_CHA
NNEL_VALUE>,<ALPHA_CHANNEL_VALUE> : 0...255 FOR EACH -
BACKGROUND COLOR.\N\N");
        PRINTF("-R, --RESULT ARGS:  <FILE_NAME> OUTPUT FILE
NAME.\N\N");
        PRINTF("-I, --INPUT ARGS:  <INPUT_NAME> OUTPUT FILE
NAME.\N\N");
        PRINTF("-N, --INFO - IMAGE INFO CALL.\N\N");
        PRINTF("-H, --HELP - HELP CALL.\N\N");
}

VOID CLEANUP(STRUCT IMG* IMG, STRUCT CONFIGS* CONFIG){
        IF(CONFIG->INPUT !=  NULL)
                FREE(CONFIG->INPUT);
        IF(CONFIG->RESULT != NULL)
                FREE(CONFIG->RESULT);
        IF(IMG->ROW_PTS != NULL ){
                FOR (INT Y = 0; Y < IMG->HEIGHT; Y++)
                FREE(IMG->ROW_PTS[Y]);
        FREE(IMG->ROW_PTS);
   }
   PRINTHELP();
}

INT MAIN(INT ARGC, CHAR* ARGV[]){
        STRUCT IMG IMG;
        INT FLAG = 0;
        STRUCT CONFIGS CONFIG = {0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, NULL, NULL};
```

```
CHAR *OPTS = "M:C:T:A:O:S:E:P:B:I:R:FNH";
STRUCT OPTION LONGOPTS[] = {
        {"FILL", NO_ARGUMENT, NULL, 'F'},
        {"HELP", NO_ARGUMENT, NULL, 'H'},
        {"COLOR", REQUIRED_ARGUMENT, NULL, 'C'},
        {"THICKNESS", REQUIRED_ARGUMENT, NULL, 'T'},
        {"ANGLE", REQUIRED_ARGUMENT, NULL, 'A'},
        {"OPTION", REQUIRED_ARGUMENT, NULL, 'O'},
        {"START", REQUIRED_ARGUMENT, NULL, 'S'},
        {"END", REQUIRED_ARGUMENT, NULL, 'E'},
        {"PARTITION", REQUIRED_ARGUMENT, NULL, 'P'},
        {"MANIPULATION", REQUIRED_ARGUMENT, NULL, 'M'},
        {"BGCOLOR", REQUIRED_ARGUMENT, NULL, 'B'},
        {"RESULT", REQUIRED_ARGUMENT, NULL, 'R'},
        {"INPUT", REQUIRED_ARGUMENT, NULL, 'I'},
        {"INFO", NO_ARGUMENT, NULL, 'N'},
        { NULL, 0, NULL, 0}
};
INT OPT;
INT LONGINDEX;
OPT = GETOPT_LONG(ARGC, ARGV, OPTS, LONGOPTS,
&LONGINDEX);
        WHILE(OPT!=-1){
            SWITCH(OPT){
                CASE 'F':
                    CONFIG.FILLED = 1;
                    BREAK;
                CASE 'N':
                    FLAG = 1;
                    BREAK;
```

```
                CASE 'C':
                    IF(OPTARG)
                        SSCANF(OPTARG, "%D,%D,%D,%D",
&CONFIG.RED, &CONFIG.GREEN, &CONFIG.BLUE, &CONFIG.ALPHA);
                    IF((CONFIG.RED > 255 || CONFIG.RED < 0) ||
(CONFIG.GREEN > 255 || CONFIG.GREEN < 0) || (CONFIG.BLUE > 255 ||
CONFIG.BLUE < 0) ||(CONFIG.ALPHA > 255 || CONFIG.ALPHA < 0)){
                        PRINTF("\NCOLOR VALUE IS
INCORRECT.\N");
                        CLEANUP(&IMG, &CONFIG);
                        RETURN 0;
                    }
                    BREAK;
                CASE 'B':
                    IF(OPTARG)
                        SSCANF(OPTARG, "%D,%D,%D,%D",
&CONFIG.D_RED, &CONFIG.D_GREEN, &CONFIG.D_BLUE,
&CONFIG.D_ALPHA);
                    IF((CONFIG.D_RED > 255 || CONFIG.D_RED <
0) || (CONFIG.D_GREEN > 255 || CONFIG.D_GREEN < 0) ||
(CONFIG.D_BLUE > 255 || CONFIG.D_BLUE < 0) ||(CONFIG.D_ALPHA >
255 || CONFIG.D_ALPHA < 0)){
                        PRINTF("\NBACKGROUND COLOR
VALUE IS INCORRECT.\N");
                        CLEANUP(&IMG, &CONFIG);
                        RETURN 0;
                    }
                    BREAK;
                CASE 'T':
                    IF(OPTARG)
```

```
                CONFIG.THICK = ATOI(OPTARG);
            IF(CONFIG.THICK < 1 || CONFIG.THICK >25){
                PRINTF("\NTHICKNESS VALUE IS
INCORRECT.\N");
                CLEANUP(&IMG, &CONFIG);
                RETURN 0;
            }
            BREAK;
        CASE 'A':
            IF(OPTARG)
                CONFIG.ANGLE = ATOI(OPTARG);
            IF((CONFIG.ANGLE != 90) &&
(CONFIG.ANGLE != 180) && (CONFIG.ANGLE != 270)){
                PRINTF("\NANGLE VALUE IS
INCORRECT.\N");
                CLEANUP(&IMG, &CONFIG);
                RETURN 0;
            }
            BREAK;
        CASE 'O':
            IF(OPTARG)
                CONFIG.OPTION = ATOI(OPTARG);
            IF((CONFIG.OPTION != 1) &&
(CONFIG.OPTION != 2) && (CONFIG.OPTION != 3)){
                PRINTF("\NBORDER OPTION VALUE IS
INCORRECT.\N");
                CLEANUP(&IMG, &CONFIG);
                RETURN 0;
            }
            BREAK;
```

```
                CASE 'S':
                    IF(OPTARG)
                        SSCANF(OPTARG, "%I,%I",
&CONFIG.START_X, &CONFIG.START_Y);
                    BREAK;
                CASE 'E':
                    IF(OPTARG)
                        SSCANF(OPTARG, "%I,%I",
&CONFIG.END_X, &CONFIG.END_Y);
                    BREAK;
                CASE 'P':
                    IF(OPTARG)
                        SSCANF(OPTARG, "%D,%D",
&CONFIG.X_PARTITION, &CONFIG.Y_PARTITION);
                    IF(CONFIG.X_PARTITION < 0 ||
CONFIG.Y_PARTITION < 0){
                        PRINTF("\NPARTITION VALUE IS
INCORRECT.\N");
                        CLEANUP(&IMG, &CONFIG);
                        RETURN 0;
                    }
                    BREAK;
                CASE 'M':
                    IF(OPTARG)
                        CONFIG.MANIPULATION =
ATOI(OPTARG);
                    IF(CONFIG.MANIPULATION > 4 &&
CONFIG.MANIPULATION < 0){
                        PRINTF("\NCHOSEN MANIPULATION
IS INCORRECT.\N");
```

```
                                CLEANUP(&IMG, &CONFIG);
                                RETURN 0;
                        }
                    CASE 'I':
                        IF(OPTARG)
                            CONFIG.INPUT =
GET_FILE_NAME(OPTARG);
                        BREAK;
                    CASE 'R':
                        IF(OPTARG)
                            CONFIG.RESULT =
GET_FILE_NAME(OPTARG);
                        BREAK;
                    CASE 'H':
                    DEFAULT:
                        CLEANUP(&IMG, &CONFIG);
                        RETURN 0;
                }
                OPT = GETOPT_LONG(ARGC, ARGV, OPTS,
LONGOPTS, &LONGINDEX);
            }

            ARGC -= OPTIND;
            ARGV += OPTIND;
        INT ANORMAL = RFILE(&IMG, CONFIG.INPUT);
        IF(ANORMAL == -1){
            CLEANUP(&IMG, &CONFIG);
            RETURN 0;
        }
        IF(ANORMAL){
```

```
            IF((CONFIG.END_X > IMG.WIDTH  || CONFIG.END_X < 0) ||
(CONFIG.END_Y > IMG.HEIGHT || CONFIG.END_Y < 0)){
                PRINTF("\NLOWER RIGHT ANGLE COORDINATES
ARE INCORRECT.\N");
                CLEANUP(&IMG, &CONFIG);
                RETURN 0;
            }
            IF((CONFIG.START_X > IMG.WIDTH || CONFIG.START_X < 0)
|| (CONFIG.START_Y > IMG.HEIGHT || CONFIG.START_Y < 0)){
                PRINTF("\NUPPER LEFT ANGLE COORDINATES ARE
INCORRECT.\N");
                CLEANUP(&IMG, &CONFIG);
                RETURN 0;
            }
        IF(ANORMAL && FLAG)
            PNG_INFO(&IMG);
        }
        IF(ANORMAL){
            SWITCH(CONFIG.MANIPULATION){
                CASE 1:
                    IF((CONFIG.X_PARTITION == 0) ||
(CONFIG.Y_PARTITION == 0) || (CONFIG.X_PARTITION > 20) ||
(CONFIG.Y_PARTITION > 20)){
                        PRINTF("\NPARTITION VALUE IS
INCORRECT.\N");
                        CLEANUP(&IMG, &CONFIG);
                        RETURN 0;
                    }
```

```
                ANORMAL = DGRID(&IMG,
CONFIG.X_PARTITION, CONFIG.Y_PARTITION, CONFIG.THICK,
CONFIG.RED, CONFIG.GREEN, CONFIG.BLUE, CONFIG.ALPHA);
                BREAK;
            CASE 2:
                IF(((CONFIG.END_X - CONFIG.START_X) < 0) ||
((CONFIG.END_Y - CONFIG.START_Y) < 0)){
                    PRINTF("\NINCORRECT
COORDINATES.\N");
                    CLEANUP(&IMG, &CONFIG);
                    RETURN 0;
                }
                ANORMAL = DRECTANGLE(&IMG,
CONFIG.START_X, CONFIG.START_Y, CONFIG.END_X,
CONFIG.END_Y, CONFIG.THICK, CONFIG.FILLED,CONFIG.RED,
CONFIG.GREEN, CONFIG.BLUE, CONFIG.ALPHA ,CONFIG.D_RED,
CONFIG.D_GREEN, CONFIG.D_BLUE, CONFIG.D_ALPHA);
                BREAK;
            CASE 3:
                ANORMAL = DBORDER(&IMG, CONFIG.OPTION,
CONFIG.THICK, CONFIG.RED, CONFIG.GREEN, CONFIG.BLUE,
CONFIG.ALPHA);
                BREAK;
            CASE 4:
                IF(CONFIG.ANGLE == 0){
                    PRINTF("\NANGLE VALUE IS INCORRECT.\
N");
                    CLEANUP(&IMG, &CONFIG);
                    RETURN 0;
                }
```

```
                    IF((CONFIG.START_X == CONFIG.END_X) &&
(CONFIG.START_Y == CONFIG.END_Y)){
                        PRINTF("\NINCORRECT COORDINATES.\
N");
                        CLEANUP(&IMG, &CONFIG);
                        RETURN 0;
                    }
                    IF(((CONFIG.END_X - CONFIG.START_X) < 0) ||
((CONFIG.END_Y - CONFIG.START_Y) < 0)){
                        PRINTF("\NINCORRECT COORDINATES.\
N");
                        CLEANUP(&IMG, &CONFIG);
                        RETURN 0;
                    }
                    ANORMAL = ROTATE(&IMG, CONFIG.START_X,
CONFIG.START_Y, CONFIG.END_X, CONFIG.END_Y, CONFIG.ANGLE);
                    BREAK;
                DEFAULT:
                    IF(CONFIG.MANIPULATION == 0 && FLAG == 1){
                        BREAK;
                    }
                    PRINTF("\NINVALID MANIPULATION INPUT.\N");
                    CLEANUP(&IMG, &CONFIG);
                    RETURN 0;
                    BREAK;
            }
        }
        //
        IF(ANORMAL && CONFIG.MANIPULATION != 0)
            WFILE(&IMG, CONFIG.RESULT);
```

```
        IF(CONFIG.INPUT !=  NULL)
                FREE(CONFIG.INPUT);
        IF(CONFIG.RESULT != NULL)
                FREE(CONFIG.RESULT);
        RETURN 0;
}
```

PNGIO.C

```
#INCLUDE <STDIO.H>
#INCLUDE <STDLIB.H>
#INCLUDE <PNG.H>
#INCLUDE "STRUCTS.H"
#INCLUDE "PNGIO.H"

#DEFINE PNGSIGSIZE 8



INT RFILE(STRUCT IMG* IMG, CHAR* FNAME){

        FILE *FP = FOPEN(FNAME, "RB");
        INT TYPE_FLAG = 0;
   PNG_BYTE BUFF[PNGSIGSIZE];

   IF(!FP){
      IF(FNAME == NULL)
         RETURN -1;
      ELSE
         PRINTF("CHOSEN FILE DOESN'T EXIST.\N");
      RETURN 0;
```

```
   }
   FREAD(BUFF, 1, PNGSIGSIZE, FP);
   TYPE_FLAG = !PNG_SIG_CMP(BUFF, 0, PNGSIGSIZE);
   IF (!TYPE_FLAG){
      PRINTF("FILE IS NOT A PNG IMAGE.\N");
        FCLOSE(FP);
      RETURN 0;
   }
   ELSE{
        IMG->PNG_PTR =
PNG_CREATE_READ_STRUCT(PNG_LIBPNG_VER_STRING, NULL,
NULL, NULL);
      IF (!IMG->PNG_PTR){
        FCLOSE(FP);
          PRINTF("ERROR DURING CREATION OF PNG_READ_STRUCT\
N");
          RETURN 0;
      }

      IMG->INFO_PTR = PNG_CREATE_INFO_STRUCT(IMG->PNG_PTR);
      IF(!IMG->INFO_PTR) {
         PNG_DESTROY_READ_STRUCT(&IMG->PNG_PTR, NULL, NULL);
         FCLOSE(FP);
         PRINTF("ERROR DURING CREATION OF PNG_INFO_STRUCT\
N");
         RETURN 0;
      }
      IF(SETJMP(PNG_JMPBUF(IMG->PNG_PTR))){
         PNG_DESTROY_READ_STRUCT(&IMG->PNG_PTR, &IMG-
>INFO_PTR, NULL);
```

```
        FCLOSE(FP);
        PRINTF("IMAGE READING ERROR.\N");
        RETURN 0;
    }
     PNG_INIT_IO(IMG->PNG_PTR, FP);
    PNG_SET_SIG_BYTES(IMG->PNG_PTR, PNGSIGSIZE);
    PNG_READ_INFO(IMG->PNG_PTR, IMG->INFO_PTR);

    IMG->WIDTH = PNG_GET_IMAGE_WIDTH(IMG->PNG_PTR, IMG->INFO_PTR);
        IMG->HEIGHT = PNG_GET_IMAGE_HEIGHT(IMG->PNG_PTR, IMG->INFO_PTR);
        IMG->COLOR_TYPE = PNG_GET_COLOR_TYPE(IMG->PNG_PTR, IMG->INFO_PTR);
        IMG->BIT_DEPTH = PNG_GET_BIT_DEPTH(IMG->PNG_PTR, IMG->INFO_PTR);


        IMG->NOP = PNG_SET_INTERLACE_HANDLING(IMG->PNG_PTR);
        PNG_READ_UPDATE_INFO(IMG->PNG_PTR, IMG->INFO_PTR);

        IF (SETJMP(PNG_JMPBUF(IMG->PNG_PTR))){
        PRINTF("IMAGE READING ERROR.\N");
            RETURN 0;
    }

        IMG->ROW_PTS = (PNG_BYTEP *) MALLOC(SIZEOF(PNG_BYTEP) * IMG->HEIGHT);
        FOR (INT Y = 0; Y < IMG->HEIGHT; Y++)
```

```
            IMG->ROW_PTS[Y] = (PNG_BYTE *)
MALLOC(PNG_GET_ROWBYTES(IMG->PNG_PTR, IMG->INFO_PTR));

        PNG_READ_IMAGE(IMG->PNG_PTR, IMG->ROW_PTS);

        FCLOSE(FP);

    RETURN 1;
  }
}

INT WFILE(STRUCT IMG *IMG, CHAR* FNAME){
  INT X,Y;
  FILE *FP;
  IF(FNAME == NULL)
    FP = FOPEN("RESULT.PNG", "WB");
  ELSE
    FP = FOPEN(FNAME, "WB");

  IF (!FP){
    PRINTF("FILE OPENNING ERROR.\N");
    RETURN 0;
  }

  IMG->PNG_PTR =
PNG_CREATE_WRITE_STRUCT(PNG_LIBPNG_VER_STRING, NULL,
NULL, NULL);

  IF (!IMG->PNG_PTR){
```

```
      PRINTF("ERROR DURING CREATION OF PNG_WRITE_STRUCT\
N");
      RETURN 0;
   }

   IMG->INFO_PTR = PNG_CREATE_INFO_STRUCT(IMG->PNG_PTR);
   IF (!IMG->INFO_PTR){
      PRINTF("ERROR DURING CREATION OF PNG_INFO_STRUCT\N");
      RETURN 0;
   }

   IF (SETJMP(PNG_JMPBUF(IMG->PNG_PTR))){
      PRINTF("ERROR IN PNG_INIT_IO\N");
      RETURN 0;
   }

   PNG_INIT_IO(IMG->PNG_PTR, FP);


   IF (SETJMP(PNG_JMPBUF(IMG->PNG_PTR))){
      PRINTF("HEADER WRITING  ERROR\N");
      RETURN 0;
   }

   PNG_SET_IHDR(IMG->PNG_PTR, IMG->INFO_PTR, IMG->WIDTH,
IMG->HEIGHT,
            IMG->BIT_DEPTH, IMG->COLOR_TYPE,
PNG_INTERLACE_NONE,
            PNG_COMPRESSION_TYPE_BASE, PNG_FILTER_TYPE_BASE);
```

```
   PNG_WRITE_INFO(IMG->PNG_PTR, IMG->INFO_PTR);


   IF (SETJMP(PNG_JMPBUF(IMG->PNG_PTR))){
      PRINTF("BYTES WRITING  ERROR\N");
      RETURN 0;
   }

   PNG_WRITE_IMAGE(IMG->PNG_PTR, IMG->ROW_PTS);

   IF (SETJMP(PNG_JMPBUF(IMG->PNG_PTR))){
      PRINTF("END OF WRITE ERROR\N");
      RETURN 0;
   }

   PNG_WRITE_END(IMG->PNG_PTR, NULL);

   FOR (Y = 0; Y < IMG->HEIGHT; Y++)
      FREE(IMG->ROW_PTS[Y]);
   FREE(IMG->ROW_PTS);
   FCLOSE(FP);
   PNG_DESTROY_WRITE_STRUCT(&IMG->PNG_PTR, &IMG-
>INFO_PTR);
   RETURN 1;
}

PNGIO.H
INT RFILE(STRUCT IMG* IMG, CHAR* FNAME);
INT WFILE(STRUCT IMG* IMG, CHAR* FNAME);
```

RENDER.H

INT DRECTANGLE(STRUCT IMG *IMG, INT X1, INT Y1, INT X2, INT Y2, INT THICK, INT FILLED, INT RED, INT GREEN, INT BLUE, INT ALPHA, INT D_RED, INT D_GREEN, INT D_BLUE, INT D_ALPHA);

INT DGRID(STRUCT IMG *IMG, INT X_PARTITION, INT Y_PARTITION,INT THICK, INT RED, INT GREEN, INT BLUE, INT ALPHA);

INT DBORDER(STRUCT IMG *IMG, INT OPTION, INT THICK, INT RED, INT GREEN, INT BLUE, INT ALPHA);

INT ROTATE(STRUCT IMG *IMG, INT X1, INT Y1, INT X2, INT Y2, INT ANGLE);