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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка PNG-изображений

Студент гр. 1382 _____ Прошичев А.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2022

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Прошичев А.В.

Группа 1382

Тема работы: Обработка PNG-изображений

Вариант 15

Общие сведения

1. Формат картинки PNG (рекомендуем использовать библиотеку libpng)
2. файл всегда соответствует формату PNG
3. обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями.
4. все поля стандартных PNG заголовков в выходном файле должны иметь те же значения что и во входном (разумеется кроме тех, которые должны быть изменены).

Программа должна реализовывать следующий функционал по обработке PNG-файла

1. Отражение заданной области. Этот функционал определяется:
    a. выбором оси относительно которой отражать (горизонтальная или вертикальная)
    b. Координатами левого верхнего угла области
    c. Координатами правого нижнего угла области

2. Рисование пентаграммы в круге. Пентаграмма определяется:
    a. либо координатами левого верхнего и правого нижнего угла квадрата, в который вписана окружность пентаграммы, либо координатами ее центра и радиусом окружности
    b. толщиной линий и окружности
    c. цветом линий и окружности

3. Рисование прямоугольника. Он определяется:
    a. Координатами левого верхнего угла
    b. Координатами правого нижнего угла
    c. Толщиной линий
    d. Цветом линий
    e. Прямоугольник может быть залит или нет
    f. цветом которым он залит, если пользователем выбран залитый
4. Рисование правильного шестиугольника. Шестиугольник определяется:
    a. либо координатами левого верхнего и правого нижнего угла квадрата, в который он вписан, либо координатами его центра и радиусом в который он списан
    b. толщиной линий
    c. цветом линий
    d. шестиугольник может быть залит или нет
    e. цветом которым залит шестиугольник, если пользователем выбран залитый

Предполагаемый объем пояснительной записки:

Не менее 22 страниц.

Дата выдачи задания: 22.03.2022

Дата сдачи реферата: 27.05.2022

Дата защиты реферата: 31.05.2022

Студент гр. 1382 _____ Прошичев А.В.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

Курсовая работа заключается в реализации программы для обработки изображений на языке Си. Для хранения и работы с изображениями были использованы структуры и функции библиотек <libpng>.

Программа взаимодействует с пользователем методом CLI. Существует возможность вывода инструкции. Поддерживает функции рисования пентаграммы, шестиугольника и прямоугольника, а также отражение заданной области.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – создать консольное приложение для обработки PNG-изображений согласно запросам пользователя. Для выполнения работы необходимо выполнить следующие задачи:

- Ввод, хранение и вывод изображения.
- Интерфейс для взаимодействия пользователем
- Реализация элементарных функций рисования пикселя, линии и окружности
- Обработка всех исключительных ситуаций

Важной частью программы является реализация интерфейса с взаимодействием пользователя и программы с помощью getopt, а также обработка PNG-файлов на основе библиотеки libpng.

1. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

1.1. Чтение и запись PNG-файлов

Для успешного считывания файла требуется его расположение в текущей директории по отношению к программе. Чтение и вывод происходит с помощью библиотеки libpng, которая позволяет распознать файл такого типа, считать все данные и параметры в специально отведённую структуру под названием png. Точно также функции данной библиотеки позволяют правильно создать PNG-файл, куда будут сохранены изменения изначального файла.

1.2. Выбор операции.

Взаимодействие с пользователем производится по средствам CLI. Для получения справки о всех возможных функциях и операциях программы после вызова исполняемого файла с флагом –h. Чтобы выбрать необходимую функцию, нужно прописать флаг этой функции сразу после исполняемого файла, далее нужно указывать все необходимые данные для корректного исполнения с помощью вспомогательных флагов.

1.3. Первая подзадача

Для реализации функции отражения следует определится с расположением отражаемого участка и осью отражения. Если ось горизонтальная, то программа перебирает половину рядов пикселей на этом прямоугольном участке сверху вниз и меняет их местами с соответствующими строками в нижней части прямоугольного участка. Аналогично происходит с вертикальной осью, только вместо рядов пикселей меняются столбцы.

1.4. Вспомогательные функции

Для решения задач на рисования были написаны вспомогательные функции рисования пикселя, линии и круга. Функция пикселя закрашивает пиксель с заданными координатами, если координаты введены корректно. Функция рисования линии используется алгоритм Брезенхэма и функцию рисования пикселей. Толщина линии воспроизводится за счёт рисования вместо пикселя круга определённого радиуса. Функция рисования окружности также работает с помощью алгоритма Брезенхэма и закрашивания пикселя. Функция заливки реализована на основе алгоритма flood fill на стеке. Для реализации функции создаётся специальный массив-маска, который помнит все свежие изменения в PNG-файле и может правильно залить фигуру, через которую, например, проходит другой контур того же цвета.

1.5. Вторая подзадача

Рисование пентаграммы полностью основано на использовании вспомогательных для рисования функций. Круг с заливкой осуществляется с помощью рисования двух окружностей разных радиусов и заливки с помощью уравнения окружности. Звезда рисуется рисованием линий в верной последовательности из заранее сосчитанных точек правильного пятиугольника.

1.6. Третья подзадача

Рисование прямоугольника происходит за счёт рисования линий по указанным пользователем координатам. Заливка такого прямоугольника производится с помощью функции заливки.

1.7. Четвёртая подзадача

Рисование шестиугольника происходит за счёт рисования линий по указанным пользователем данным. Заливка такого шестиугольника производится с помощью функции заливки.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы было создано консольное предложение для обработки изображения. Программа умеет считывать PNG-файл, создавать новый корректный PNG-file, рисовать отдельные пиксели, линии, окружности, пентаграммы, прямоугольники, шестиугольники и отражать на заданном области. Можно сделать вывод, что итоговая программа и поставленная цель для написания программы совпали.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- Сайт (онлайн курс) Программирование Си, 2 семестр
- Сайт (репозиторий) Репозиторий с примерами кода
- Сайт (сайт) Записи лекций по программированию

ПРИЛОЖЕНИЕ А
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

```
gcc main.c -lpng -lm -o main && ./main -h
```

```
Руководство по использованию программы:
-Программа обрабатывает PNG-файлы.
-Для запуска программы необходимо передать следующие аргументы:
        ./a.out  -- имя исполняемого файла.
        Основная функция обработки изображения вводиться сразу же после имени исполняемого файла с заглавной буквы!
-Список функций:
        --Reflect/-R -- функция отзеркаливания прямоугольной области, заданной в границах обрабатываемого изображения.
                P.S. Чтобы задать вертикальное или горизонтальное отражение, следует использовать флаги --vertical и --horizontal соответственно.
        --Pentagram/-P -- функция рисования пентаграммы в квадратной области.
                P.S. Может задаваться левой верхней и правой нижней вершинами квадрата через флаги --start и --end, а также окружностью через флаги --center и --radius.
                P.S. Может изменять цвет звезды с помощью флага --fill/-f.
        --Square/-S -- функция рисования прямоугольника.
                P.S. Может быть залит с помощью флага --fill/-f.
        --Hexagon/-H -- функция рисования шестиугольника в квадратной области.
                P.S. Может задаваться левой верхней и правой нижней вершинами квадрата через флаги --start и --end, а также окружностью через флаги --center и --radius.
                P.S. Может быть залит с помощью флага --fill/-f.
        --Information/-I -- функция вывода информации о изображении.
        <arg1>,<arg2> ... -- аргументы к ключу, если требуются (указаны в списке ключей, аргументы разделяются запятой).
-Список ключей и их аргументы:
        --input/-i <имя файла> -- имя PNG-файла, который необходимо обработать. Он должен находиться в текущей директории.
        --output/-o <имя файла> -- имя PNG-файла, в который необходимо сохранить изменения. Он должен находиться в текущей директории.
        --help/-h -- вывод руководства по использованию программы.
        --bold/-b <целое число> -- установка значения толщины линий для функций с рисованием.
        --start/-s <целое число,целое число> -- установка значений координат левого верхнего угла прямоугольника.
        --end/-e <целое число,целое число> -- установка значений координат правого нижнего угла прямоугольника.
        --radius/-r <целое число> -- установка значения радиуса для функций, который могут работать в окружностью.
        --center/-c <целое число,целое число> -- установка значений координат центра окружности для функций, которые могут её обрабатывать.
        --color/-p <целое число,целое число,целое число,целое число> -- установка цвета линии для красного, зелёного, синего и альфа канала соответсвенно для фукнций рисования.
        --fill/-f <целое число,целое число,целое число,целое число> -- установка цвета заливки или второго цвета для красного, зелёного, синего и альфа канала соответсвенно для фукнций
 рисования.
```

Пример 1. Справка.

```
gcc main.c -lpng -lm -o main && ./main -I -i img1.png
```

```
Длина изображения: 1200
Ширина изображения: 1039
```

Пример 2. Информаци о файле.

```
gcc main.c -lpng -lm -o main && ./main -R -s 500,500 -e 1829,1090 -i img1.png -o res.png -v
```

Пример 3. Первая подзадача.

```
gcc main.c -lpng -lm -o main && ./main -P -i img3.png -o res.png -p 255,130,0,255 -s 170,70 -e 270,170 -b 10
```

Пример 4. Вторая подзадача.

```
gcc main.c -lpng -lm -o main && ./main -S -i img2.png -o res.png -p 255,0,0,255 -s 1700,1000 -e 1900,1300 -b 30
```

Пример 5. Третья подзадача.

```
gcc main.c -lpng -lm -o main && ./main -H -i img2.png -o res.png -p 255,0,120,255 -c 1000,1000 -r 150 -b 50
```

Пример 6. Четвёртая подзадача.

ПРИЛОЖЕНИЕ В

```
#INCLUDE <UNISTD.H>
#INCLUDE <STDLIB.H>
#INCLUDE <STDIO.H>
#INCLUDE <STDARG.H>
#INCLUDE <MATH.H>
#INCLUDE <GETOPT.H>
#INCLUDE <PNG.H>

#DEFINE MAX_PATH 261

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

   PNG_STRUCTP PNG_PTR;
   PNG_INFOP INFO_PNG;
   INT NUMBER_OF_PASSES;
   PNG_BYTEP *ROW_POINTERS;
} PNG;

TYPEDEF STRUCT{
   INT REF, PNT, SQR, HEX, TYPE;
   INT IN, OUT;
   INT DIR; // FALSE (VERTICAL/NOT FILL) - TRUE (HORIZONTAL/FILL)
   INT XL, YL; // COORDS OF THE LEFT TOP ANGLE
   INT XR, YR; // COORDS OF THE RIGHT BOTTOM ANGLE
   INT R; // RADIUS AND COORDS OF THE CENTER OF THE CIRCLE
   INT BOLD, INFORM;
   CHAR INPUT_FILE[MAX_PATH], OUTPUT_FILE[MAX_PATH];
   PNG_BYTE CLR_LINE[4], CLR_OBJECT[4];
} CONFIG;

VOID PRINT_HELP(){
   PRINTF("РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ:\N");
   PRINTF("-ПРОГРАММА ОБРАБАТЫВАЕТ PNG-ФАЙЛЫ.\N");
   PRINTF("-ДЛЯ ЗАПУСКА ПРОГРАММЫ НЕОБХОДИМО ПЕРЕДАТЬ СЛЕДУЮЩИЕ АРГУМЕНТЫ:\N");
   PRINTF("\T./A.OUT  -- ИМЯ ИСПОЛНЯЕМОГО ФАЙЛА.\N");
   PRINTF("\TОСНОВНАЯ ФУНКЦИЯ ОБРАБОТКИ ИЗОБРАЖЕНИЯ ВВОДИТЬСЯ СРАЗУ ЖЕ ПОСЛЕ ИМЕНИ
ИСПОЛНЯЕМОГО ФАЙЛА С ЗАГЛАВНОЙ БУКВЫ!\N");
   PRINTF("-СПИСОК ФУНКЦИЙ:\N");
   PRINTF("\T--REFLECT/-R -- ФУНКЦИЯ ОТЗЕРКАЛИВАНИЯ ПРЯМОУГОЛЬНОЙ ОБЛАСТИ, ЗАДАННОЙ В ГРАНИЦАХ
ОБРАБАТЫВАЕМОГО ИЗОБРАЖЕНИЯ.\N");
   PRINTF("\T\TP.S. ЧТОБЫ ЗАДАТЬ ВЕРТИКАЛЬНОЕ ИЛИ ГОРИЗОНТАЛЬНОЕ ОТРАЖЕНИЕ, СЛЕДУЕТ ИСПОЛЬЗОВАТЬ
ФЛАГИ --VERTICAL И --HORIZONTAL СООТВЕТСТВЕННО.\N");
   PRINTF("\T--PENTAGRAM/-P -- ФУНКЦИЯ РИСОВАНИЯ ПЕНТАГРАММЫ В КВАДРАТНОЙ ОБЛАСТИ.\N");
   PRINTF("\T\TP.S. МОЖЕТ ЗАДАВАТЬСЯ ЛЕВОЙ ВЕРХНЕЙ И ПРАВОЙ НИЖНЕЙ ВЕРШИНАМИ КВАДРАТА ЧЕРЕЗ ФЛАГИ --
START И --END, А ТАКЖЕ ОКРУЖНОСТЬЮ ЧЕРЕЗ ФЛАГИ --CENTER И --RADIUS.\N");
   PRINTF("\T\TP.S. МОЖЕТ ИЗМЕНЯТЬ ЦВЕТ ЗВЕЗДЫ С ПОМОЩЬЮ ФЛАГА --FILL/-F.\N");
```

```
   PRINTF("\T--SQUARE/-S -- ФУНКЦИЯ РИСОВАНИЯ ПРЯМОУГОЛЬНИКА.\N");
   PRINTF("\T\TP.S. МОЖЕТ БЫТЬ ЗАЛИТ С ПОМОЩЬЮ ФЛАГА --FILL/-F.\N");
   PRINTF("\T--HEXAGON/-H -- ФУНКЦИЯ РИСОВАНИЯ ШЕСТИУГОЛЬНИКА В КВАДРАТНОЙ ОБЛАСТИ.\N");
   PRINTF("\T\TP.S. МОЖЕТ ЗАДАВАТЬСЯ ЛЕВОЙ ВЕРХНЕЙ И ПРАВОЙ НИЖНЕЙ ВЕРШИНАМИ КВАДРАТА ЧЕРЕЗ ФЛАГИ --
START И --END, А ТАКЖЕ ОКРУЖНОСТЬЮ ЧЕРЕЗ ФЛАГИ --CENTER И --RADIUS.\N");
   PRINTF("\T\TP.S. МОЖЕТ БЫТЬ ЗАЛИТ С ПОМОЩЬЮ ФЛАГА --FILL/-F.\N");
   PRINTF("\T--INFORMATION/-I -- ФУНКЦИЯ ВЫВОДА ИНФОРМАЦИИ О ИЗОБРАЖЕНИИ.\N");
   PRINTF("\T<ARG1>,<ARG2> ... -- АРГУМЕНТЫ К КЛЮЧУ, ЕСЛИ ТРЕБУЮТСЯ (УКАЗАНЫ В СПИСКЕ КЛЮЧЕЙ, АРГУМЕНТЫ
РАЗДЕЛЯЮТСЯ ЗАПЯТОЙ).\N");
   PRINTF("-СПИСОК КЛЮЧЕЙ И ИХ АРГУМЕНТЫ:\N");
   PRINTF("\T--INPUT/-I <ИМЯ ФАЙЛА> -- ИМЯ PNG-ФАЙЛА, КОТОРЫЙ НЕОБХОДИМО ОБРАБОТАТЬ. ОН ДОЛЖЕН
НАХОДИТЬСЯ В ТЕКУЩЕЙ ДИРЕКТОРИИ.\N");
   PRINTF("\T--OUTPUT/-O <ИМЯ ФАЙЛА> -- ИМЯ PNG-ФАЙЛА, В КОТОРЫЙ НЕОБХОДИМО СОХРАНИТЬ ИЗМЕНЕНИЯ. ОН
ДОЛЖЕН НАХОДИТЬСЯ В ТЕКУЩЕЙ ДИРЕКТОРИИ.\N");
   PRINTF("\T--HELP/-H -- ВЫВОД РУКОВОДСТВА ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ.\N");
   PRINTF("\T--BOLD/-B <ЦЕЛОЕ ЧИСЛО> -- УСТАНОВКА ЗНАЧЕНИЯ ТОЛЩИНЫ ЛИНИЙ ДЛЯ ФУНКЦИЙ С
РИСОВАНИЕМ.\N");
   PRINTF("\T--START/-S <ЧЕЛОЕ ЧИСЛО,ЦЕЛОЕ ЧИСЛО> -- УСТАНОВКА ЗНАЧЕНИЙ КООРДИНАТ ЛЕВОГО ВЕРХНЕГО УГЛА
ПРЯМОУГОЛЬНИКА.\N");
   PRINTF("\T--END/-E <ЦЕЛОЕ ЧИСЛО,ЦЕЛОЕ ЧИСЛО> -- УСТАНОВКА ЗНАЧЕНИЙ КООРДИНАТ ПРАВОГО НИЖНЕГО УГЛА
ПРЯМОУГОЛЬНИКА.\N");
   PRINTF("\T--RADIUS/-R <ЦЕЛОЕ ЧИСЛО> -- УСТАНОВКА ЗНАЧЕНИЯ РАДИУСА ДЛЯ ФУНКЦИЙ, КОТОРЫЙ МОГУТ
РАБОТАТЬ В ОКРУЖНОСТЬЮ.\N");
   PRINTF("\T--CENTER/-C <ЦЕЛОЕ ЧИСЛО,ЦЕЛОЕ ЧИСЛО> -- УСТАНОВКА ЗНАЧЕНИЙ КООРДИНАТ ЦЕНТРА ОКРУЖНОСТИ
ДЛЯ ФУНКЦИЙ, КОТОРЫЕ МОГУТ ЕЁ ОБРАБАТЫВАТЬ.\N");
   PRINTF("\T--COLOR/-P <ЦЕЛОЕ ЧИСЛО,ЦЕЛОЕ ЧИСЛО,ЦЕЛОЕ ЧИСЛО,ЦЕЛОЕ ЧИСЛО> -- УСТАНОВКА ЦВЕТА ЛИНИИ
ДЛЯ КРАСНОГО, ЗЕЛЁНОГО, СИНЕГО И АЛЬФА КАНАЛА СООТВЕТСВЕННО ДЛЯ ФУКНЦИЙ РИСОВАНИЯ.\N");
   PRINTF("\T--FILL/-F <ЦЕЛОЕ ЧИСЛО,ЦЕЛОЕ ЧИСЛО,ЦЕЛОЕ ЧИСЛО,ЦЕЛОЕ ЧИСЛО> -- УСТАНОВКА ЦВЕТА ЗАЛИВКИ ИЛИ
ВТОРОГО ЦВЕТА ДЛЯ КРАСНОГО, ЗЕЛЁНОГО, СИНЕГО И АЛЬФА КАНАЛА СООТВЕТСВЕННО ДЛЯ ФУКНЦИЙ
РИСОВАНИЯ.\N");
   PRINTF("\T--VERTICAL -- УСТАНОВКА ВЕРТИКАЛЬНОЙ ОСИ ДЛЯ ОТРАЖЕНИЯ.\N");
   PRINTF("\T--HORIZONTAL -- УСТАНОВКА ГОРИЗОНТАЛЬНОЙ ОСИ ДЛЯ ОТРАЖЕНИЯ.\N");
   PRINTF("ПРИМЕР РИСОВАНИЯ ШЕСТИУГОЛЬНИКА:\N");
   PRINTF("\T ./A.OUT -H -C 0,0 -R 100 -B 10 -P 255,0,0,255 -F 255,255,0,255 -I IMG1.PNG -O IMG2.PNG\N");
}

VOID INFORMATION(PNG *IMG){
   PRINTF("ДЛИНА ИЗОБРАЖЕНИЯ: %D\NШИРИНА ИЗОБРАЖЕНИЯ: %D\N", IMG->WIDTH, IMG->HEIGHT);
}

INT PROC_COMMS(CONFIG *DATA, INT OPT){
   SWITCH(OPT){
      CASE 'H':
         PRINTF("ПРИ НЕОБХОДИМОСТИ ИНСТРУКЦИЮ СЛЕДУЕТ ВЫЗЫВАТЬ БЕЗ КАКИХ-ЛИБО ДРУГИХ ФЛАГОВ!\N");
         RETURN 4;
      CASE 'I':
         DATA->IN = SSCANF(OPTARG, "%S", DATA->INPUT_FILE);
         RETURN 0;
      CASE 'O':
         DATA->OUT = SSCANF(OPTARG, "%S", DATA->OUTPUT_FILE);
         RETURN 0;
      CASE 'S':
```

```
        IF(SSCANF(OPTARG, "%D,%D", &DATA->XL, &DATA->YL) < 2){
            PRINTF("НЕ УДАЛОСЬ СЧИТАТЬ КООРДИНАТЫ ЛЕВОЙ ВЕРХНЕЙ ТОЧКИ!\N");
            RETURN 1;
        }
        DATA->TYPE = 1;
        RETURN 0;
    CASE 'E':
        IF(SSCANF(OPTARG, "%D,%D", &DATA->XR, &DATA->YR) < 2){
            PRINTF("НЕ УДАЛОСЬ СЧИТАТЬ КООРДИНАТЫ ПРАВОЙ НИЖНЕЙ ТОЧКИ!\N");
            RETURN 1;
        }
        DATA->TYPE = 1;
        RETURN 0;
    CASE 'R':
        IF(!DATA->HEX && !DATA->PNT)
            RETURN 2;
        IF(SSCANF(OPTARG, "%D", &DATA->R) == 0) {
            PRINTF("НЕ УДАЛОСЬ СЧИТАТЬ ЗНАЧЕНИЕ РАДИУСА!\N");
            RETURN 1;
        }
        IF(DATA->R < 1){
            PRINTF("ДЛЯ КОРРЕКТНОЙ РАБОТЫ ПРОГРАММЫ РАДИУС ДОЛЖЕН БЫТЬ ПОЛОЖИТЕЛЬНЫМ ЦЕЛЫМ
ЧИСЛОМ!\N");
            RETURN 4;
        }
        RETURN 0;
    CASE 'C':
        IF(!DATA->HEX && !DATA->PNT)
            RETURN 2;
        IF(SSCANF(OPTARG, "%D,%D", &DATA->XR, &DATA->YR) < 2){
            PRINTF("НЕ УДАЛОСЬ СЧИТАТЬ КООРДИНАТЫ ЦЕНТРА ОКРУЖНОСТИ!\N");
            RETURN 1;
        }
        RETURN 0;
    CASE 'P':
        IF(DATA->REF)
            RETURN 2;

        IF(SSCANF(OPTARG, "%HHD,%HHD,%HHD,%HHD",
                &DATA->CLR_LINE[0],
                &DATA->CLR_LINE[1],
                &DATA->CLR_LINE[2],
                &DATA->CLR_LINE[3]) < 4){
            PRINTF("НЕ УДАЛОСЬ СЧИТАТЬ ЗНАЧЕНИЯ КОМПОНЕНТ ПЕРЕДАННОГО ЦВЕТА ЛИНИЙ!\N");
            RETURN 1;
        }
        FOR(INT I = 0; I < 4; I++)
            IF(DATA->CLR_LINE[I] < 0 || DATA->CLR_LINE[I] > 255){
                PRINTF("ДЛЯ КОРРЕКТНОЙ РАБОТЫ КОМПОНЕНТЫ, ЗАДАЮЩИЕ ЦВЕТ, ДОЛЖНЫ НАХОДИТЬСЯ В
ДИАПАЗОНЕ ОТ 0 ДО 255");
                RETURN 4;
            }
```

```
        FOR(INT I = 0; I < 4; I++){
            DATA->CLR_OBJECT[I] = DATA->CLR_LINE[I];
        }
        RETURN 0;
    CASE 'F':
        IF(DATA->REF)
            RETURN 2;
        IF(SSCANF(OPTARG, "%HHD,%HHD,%HHD,%HHD",
                &DATA->CLR_OBJECT[0],
                &DATA->CLR_OBJECT[1],
                &DATA->CLR_OBJECT[2],
                &DATA->CLR_OBJECT[3]) < 4){
            PRINTF("НЕ УДАЛОСЬ СЧИТАТЬ ЗНАЧЕНИЯ КОМПОНЕНТ ПЕРЕДАННОГО ЦВЕТА ЗАЛИВКИ!\N");
            RETURN 1;
        }
        FOR(INT I = 0; I < 4; I++)
            IF(DATA->CLR_LINE[I] < 0 || DATA->CLR_LINE[I] > 255){
                PRINTF("ДЛЯ КОРРЕКТНОЙ РАБОТЫ КОМПОНЕНТЫ, ЗАДАЮЩИЕ ЦВЕТ, ДОЛЖНЫ НАХОДИТЬСЯ В
ДИАПАЗОНЕ ОТ 0 ДО 255");
                RETURN 4;
            }
        DATA->DIR = 1;
        RETURN 0;
    CASE 'B':
        IF(DATA->REF)
            RETURN 2;
        IF(SSCANF(OPTARG, "%D", &DATA->BOLD) == 0){
            PRINTF("НЕ УДАЛОСЬ СЧИТАТЬ ЗНАЧЕНИЕ ТОЛЩИНЫ!\N");
            RETURN 1;
        }
        IF(DATA->BOLD < 1){
            PRINTF("ДЛЯ КОРРЕКТНОЙ РАБОТЫ ПРОГРАММЫ ТОЛЩИНА ДОЛЖЕНА БЫТЬ ПОЛОЖИТЕЛЬНЫМ ЦЕЛЫМ
ЧИСЛОМ!\N");
            RETURN 4;
        }
        RETURN 0;
    CASE 'V':
        IF(!DATA->REF)
            RETURN 2;
        DATA->DIR = 0;
        RETURN 0;
    CASE 'L':
        IF(!DATA->REF)
            RETURN 2;
        DATA->DIR = 1;
        RETURN 0;
    CASE 'R':
        RETURN 3;
    CASE 'H':
        RETURN 3;
    CASE 'P':
        RETURN 3;
```

```
        CASE 'S':
            RETURN 3;
        CASE 'I':
            RETURN 0;
        DEFAULT:
            RETURN 1;
    }
}

INT READ_PNG_FILE(CHAR *FILE_NAME, PNG *IMAGE){
    CHAR HEADER[8];
    FILE *FP = FOPEN(FILE_NAME, "RB");
    IF(!FP){
        PRINTF("ФАЙЛ С ИМЕНЕМ <%S> НЕ НАЙДЕН!\N", FILE_NAME);
        RETURN 1;
    }
    FREAD(HEADER, 1, 8, FP);
    IF(PNG_SIG_CMP(HEADER, 0, 8)){
        PRINTF("ФАЙЛ НЕ ОПРЕДЕЛЁН, КАК ФАЙЛ ТИПА PNG!\NСЛЕДУЕТ ПРОВЕРИТЬ ТИП ПЕРЕДАВАЕМОГО ФАЙЛА!\N");
        FCLOSE(FP);
        RETURN 1;
    }
    IMAGE->PNG_PTR = PNG_CREATE_READ_STRUCT(PNG_LIBPNG_VER_STRING, NULL, NULL, NULL);
    IF(!IMAGE->PNG_PTR){
        PRINTF("НЕ УДАЛОСЬ СЧИТАТЬ СТРУКТУРУ ПРЕДЛАГАЕМОГО ФАЙЛА!\N");
        FCLOSE(FP);
        RETURN 1;
    }
    IMAGE->INFO_PNG = PNG_CREATE_INFO_STRUCT(IMAGE->PNG_PTR);
    IF(!IMAGE->INFO_PNG){
        PRINTF("НЕ УДАЛОСЬ СЧИТАТЬ ИНФОРМАЦИЮ ОБ ИЗОБРАЖЕНИИ!\N");
        FCLOSE(FP);
        RETURN 1;
    }
    IF(SETJMP(PNG_JMPBUF(IMAGE->PNG_PTR))){
        PRINTF("ОШИБКА ПОЛУЧЕНИЯ ДАННЫХ!\N");
        FCLOSE(FP);
        RETURN 1;
    }
    PNG_INIT_IO(IMAGE->PNG_PTR, FP);
    PNG_SET_SIG_BYTES(IMAGE->PNG_PTR, 8);

    PNG_READ_INFO(IMAGE->PNG_PTR, IMAGE->INFO_PNG);

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

    IMAGE->NUMBER_OF_PASSES = PNG_SET_INTERLACE_HANDLING(IMAGE->PNG_PTR);

    PNG_READ_UPDATE_INFO(IMAGE->PNG_PTR, IMAGE->INFO_PNG);
```

```
    IF(SETJMP(PNG_JMPBUF(IMAGE->PNG_PTR))){
        PRINTF("ОШИБКА ОБРАБОТКИ ЦВЕТОВ!\N");
        FCLOSE(FP);
        RETURN 1;
    }
    IMAGE->ROW_POINTERS = (PNG_BYTEP *)MALLOC(SIZEOF(PNG_BYTEP) * IMAGE->HEIGHT);
    FOR(INT Y = 0; Y < IMAGE->HEIGHT; Y++){
        IMAGE->ROW_POINTERS[Y] = (PNG_BYTE *) MALLOC(PNG_GET_ROWBYTES(IMAGE->PNG_PTR, IMAGE->INFO_PNG));
    }
    PNG_READ_IMAGE(IMAGE->PNG_PTR, IMAGE->ROW_POINTERS);
    FCLOSE(FP);
    RETURN 0;
}

VOID WRITE_PNG_FILE(CHAR *FILE_NAME, PNG *IMAGE){
    FILE *FP = FOPEN(FILE_NAME, "WB");
    IF(!FP){
        PRINTF("ФАЙЛ С ИМЕНЕМ <%S> НЕ НАЙДЕН!\N", FILE_NAME);
        RETURN;
    }
    IMAGE->PNG_PTR = PNG_CREATE_WRITE_STRUCT(PNG_LIBPNG_VER_STRING, NULL, NULL, NULL);
    IF(!IMAGE->PNG_PTR){
        PRINTF("НЕ УДАЛОСЬ ЗАПИСАТЬ СТРУКТУРУ ПРЕДЛАГАЕМОГО ФАЙЛА!\N");
        FOR(INT Y = 0; Y < IMAGE->HEIGHT; Y++){
            FREE(IMAGE->ROW_POINTERS[Y]);
        }
        FREE(IMAGE->ROW_POINTERS);
        FCLOSE(FP);
        RETURN;
    }

    IMAGE->INFO_PNG = PNG_CREATE_INFO_STRUCT(IMAGE->PNG_PTR);
    IF(!IMAGE->INFO_PNG){
        PRINTF("НЕ УДАЛОСЬ ЗАПИСАТЬ ИНФОРМАЦИЮ ОБ ОБРАБОТОННОМ ФАЙЛЕ!\N");
        FOR(INT Y = 0; Y < IMAGE->HEIGHT; Y++){
            FREE(IMAGE->ROW_POINTERS[Y]);
        }
        FREE(IMAGE->ROW_POINTERS);
        FCLOSE(FP);
        RETURN;
    }
    IF(SETJMP(PNG_JMPBUF(IMAGE->PNG_PTR))){
        PRINTF("НЕ УДАЛОСЬ ПЕРЕДАТЬ ИНФОРМАЦИЮ О ИЗОБРАЖЕНИЕ!\N");
        FOR(INT Y = 0; Y < IMAGE->HEIGHT; Y++){
            FREE(IMAGE->ROW_POINTERS[Y]);
        }
        FREE(IMAGE->ROW_POINTERS);
        FCLOSE(FP);
        RETURN;
    }
    PNG_INIT_IO(IMAGE->PNG_PTR, FP);
```

```
    IF (SETJMP(PNG_JMPBUF(IMAGE->PNG_PTR))){
        PRINTF("НЕ УДАЛОСЬ СОСТАВИТЬ ЗАГАЛОВОК ФАЙЛА!\N");
        FOR(INT Y = 0; Y < IMAGE->HEIGHT; Y++){
            FREE(IMAGE->ROW_POINTERS[Y]);
        }
        FREE(IMAGE->ROW_POINTERS);
        FCLOSE(FP);
        RETURN;
    }
    PNG_SET_IHDR(IMAGE->PNG_PTR, IMAGE->INFO_PNG, IMAGE->WIDTH, IMAGE->HEIGHT,
                 IMAGE->BIT_DEPTH, IMAGE->COLOR_TYPE, PNG_INTERLACE_NONE,
                 PNG_COMPRESSION_TYPE_BASE, PNG_FILTER_TYPE_BASE );
    PNG_WRITE_INFO(IMAGE->PNG_PTR, IMAGE->INFO_PNG);

    IF(SETJMP(PNG_JMPBUF(IMAGE->PNG_PTR))){
        PRINTF("НЕ УДАЛОСЬ ПЕРЕДАТЬ ИНФОРМАЦИЮ О ЦВЕТАХ\N");
        FOR(INT Y = 0; Y < IMAGE->HEIGHT; Y++){
            FREE(IMAGE->ROW_POINTERS[Y]);
        }
        FREE(IMAGE->ROW_POINTERS);
        FCLOSE(FP);
        RETURN;
    }

    PNG_WRITE_IMAGE(IMAGE->PNG_PTR, IMAGE->ROW_POINTERS);

    IF(SETJMP(PNG_JMPBUF(IMAGE->PNG_PTR))){
        PRINTF("НЕКОРРЕКТНОЕ ЗАВЕРШЕНИЕ ЗАПИСИ!\N");
        FOR(INT Y = 0; Y < IMAGE->HEIGHT; Y++){
            FREE(IMAGE->ROW_POINTERS[Y]);
        }
        FREE(IMAGE->ROW_POINTERS);
        FCLOSE(FP);
        RETURN;
    }

    PNG_WRITE_END(IMAGE->PNG_PTR, NULL);

    FOR(INT Y = 0; Y < IMAGE->HEIGHT; Y++){
        FREE(IMAGE->ROW_POINTERS[Y]);
    }
    FREE(IMAGE->ROW_POINTERS);
    FCLOSE(FP);
}

VOID REFLECT(INT LINE, INT X0, INT Y0, INT X1, INT Y1, PNG *IMG){
    INT WDTH = X1 - X0, HGHT = Y1 - Y0;
    IF(LINE){
        FOR(INT Y = Y0; Y < Y0 + HGHT / 2; Y++){
            PNG_BYTE *ROW_STR = IMG->ROW_POINTERS[Y];
            PNG_BYTE *ROW_FNSH = IMG->ROW_POINTERS[Y0 + Y1 - Y - 1];
```

```
            FOR(INT X = X0; X < X1; X++){
               PNG_BYTEP PTR_STR = &(ROW_STR[X * 4]);
               PNG_BYTEP PTR_FNSH = &(ROW_FNSH[X * 4]);
               FOR(INT I = 0; I < 4; I++){
                  PNG_BYTE SW = PTR_STR[I];
                  PTR_STR[I] = PTR_FNSH[I];
                  PTR_FNSH[I] = SW;
               }
            }
         }
      }
      ELSE{
         FOR(INT Y = Y0; Y < Y1; Y++){
            PNG_BYTE *ROW = IMG->ROW_POINTERS[Y];
            FOR(INT X = X0; X < X0 + WDTH / 2; X++){
               PNG_BYTEP PTR_STR = &(ROW[X * 4]);
               PNG_BYTEP PTR_FNSH = &(ROW[(X0 + X1 - X - 1) * 4 ]);

               FOR(INT I = 0; I < 4; I++){
                  PNG_BYTE SW = PTR_STR[I];
                  PTR_STR[I] = PTR_FNSH[I];
                  PTR_FNSH[I] = SW;
               }
            }
         }
      }
   }

VOID DRAWPIXEL(PNG* IMG, INT X0, INT Y0, PNG_BYTEP COLOR, INT **MASK){
   IF(X0 < 0 || X0 >= IMG->WIDTH || Y0 < 0 || Y0 >= IMG->HEIGHT)
      RETURN;
   IMG->ROW_POINTERS[Y0][X0 * 4] = COLOR[0];
   IMG->ROW_POINTERS[Y0][X0 * 4 + 1] = COLOR[1];
   IMG->ROW_POINTERS[Y0][X0 * 4 + 2] = COLOR[2];
   IMG->ROW_POINTERS[Y0][X0 * 4 + 3] = COLOR[3];
   MASK[Y0 + 1][X0 + 1] = 1;
}

VOID DRAWLINE(PNG *IMG, INT X0, INT Y0, INT X1, INT Y1, INT BOLD, PNG_BYTEP COLOR, INT **MASK){
   INT DX, DY, FLAG, FLAGX, FLAGY;
   DY = Y1 - Y0;
   DX = X0 - X1;
   FLAG = ABS(DY) > ABS(DX) ? 1 : -1;
   FLAGX = DX < 0 ? -1 : 1;
   FLAGY = DY < 0 ? -1 : 1;
   INT DERR = 0;
   FOR(INT Y = Y0 - BOLD / 2; Y <= Y0 + BOLD / 2; Y++){
      FOR(INT X = X0 - BOLD / 2; X <= X0 + BOLD / 2; X++){
         IF((X - X0) * (X - X0) + (Y - Y0) * (Y - Y0) <= (BOLD / 2) * (BOLD / 2)){
            DRAWPIXEL(IMG, X, Y, COLOR, MASK);
```

```
            }
        }
    }
    INT XT = X0, YT = Y0;
    IF(FLAG == -1){
        WHILE(XT != X1 || YT != Y1){
            DERR += DY * FLAGY;
            IF(DERR > 0){
                DERR -= DX * FLAGX;
                YT += FLAGY;
            }
            XT -= FLAGX;
            FOR(INT Y = YT - BOLD / 2; Y <= YT + BOLD / 2; Y++){
                FOR(INT X = XT - BOLD / 2; X <= XT + BOLD / 2; X++){
                    IF((X - XT) * (X - XT) + (Y - YT) * (Y - YT) <= (BOLD / 2) * (BOLD / 2)){
                        DRAWPIXEL(IMG, X, Y, COLOR, MASK);
                    }
                }
            }
        }
    }
    ELSE{
        WHILE(XT != X1 || YT != Y1){
            DERR += DX * FLAGX;
            IF(DERR > 0){
                DERR -= DY * FLAGY;
                XT -= FLAGX;
            }
            YT += FLAGY;
            FOR(INT Y = YT - BOLD / 2; Y <= YT + BOLD / 2; Y++){
                FOR(INT X = XT - BOLD / 2; X <= XT + BOLD / 2; X++){
                    IF((X - XT) * (X - XT) + (Y - YT) * (Y - YT) <= (BOLD / 2) * (BOLD / 2)){
                        DRAWPIXEL(IMG, X, Y, COLOR, MASK);
                    }
                }
            }
        }
    }
}

VOID DRAWCIRCLE(PNG *IMG, INT X0, INT Y0, INT R, PNG_BYTEP COLOR, INT **MASK){
    INT DELTA = 1 - 2 * R, ERR = 0, X = 0, Y = R;
    WHILE(Y >= X){
        DRAWPIXEL(IMG, X0 + X, Y0 + Y, COLOR, MASK);
        DRAWPIXEL(IMG, X0 + X, Y0 - Y, COLOR, MASK);
        DRAWPIXEL(IMG, X0 - X, Y0 + Y, COLOR, MASK);
        DRAWPIXEL(IMG, X0 - X, Y0 - Y, COLOR, MASK);
        DRAWPIXEL(IMG, X0 + Y, Y0 + X, COLOR, MASK);
        DRAWPIXEL(IMG, X0 + Y, Y0 - X, COLOR, MASK);
        DRAWPIXEL(IMG, X0 - Y, Y0 + X, COLOR, MASK);
        DRAWPIXEL(IMG, X0 - Y, Y0 - X, COLOR, MASK);
        ERR = 2 * (DELTA + Y) - 1;
```

```
        IF((DELTA < 0) && (ERR <= 0)){
            DELTA += 2 * ++X + 1;
            CONTINUE;
        }
        IF((DELTA > 0) && (ERR > 0)){
            DELTA -= 2 * --Y + 1;
            CONTINUE;
        }
        DELTA += 2 * (++X - --Y);
    }
}

VOID DRAWPENTOGRAM(PNG *IMG, INT TYPEIO, INT X0, INT Y0, INT X1, INT Y1, INT R, INT BOLD, PNG_BYTEP CLR1,
PNG_BYTEP CLR2, INT **MASK){
    IF(TYPEIO){
        R = (X1 - X0)/2;
        X0 = X1 - R;
        Y0 = Y1 - R;
    }
    ELSE{
        X0 = X1, Y0 = Y1;
    }
    IF(BOLD > R/2) // ERROR
        RETURN;
    DRAWCIRCLE(IMG, X0, Y0, R, CLR1, MASK);
    DRAWCIRCLE(IMG, X0, Y0, R + BOLD, CLR1, MASK);
    FOR(INT Y = Y0 - R - BOLD; Y <= Y0 + R + BOLD; Y++){
        FOR(INT X = X0 - R - BOLD; X <= X0 + R + BOLD; X++){
            IF((X - X0) * (X - X0) + (Y - Y0)  * (Y - Y0) < R * R)
                CONTINUE;
            ELSE IF((X - X0) * (X - X0) + (Y - Y0) * (Y - Y0) <= (R + BOLD) * (R + BOLD)){
                DRAWPIXEL(IMG, X, Y, CLR1, MASK);
            }
        }
    }
    INT XT = (INT)CEIL((R + BOLD / 2) * SINL(72.0 / 180.0 * M_PI)),
            YT = (INT)CEIL( (R + BOLD / 2)  * COSL(72.0 / 180.0 * M_PI)),
            XB = (INT)CEIL((R + BOLD / 2) * SINL(144.0 / 180.0 * M_PI)),
            YB = (INT)CEIL((R + BOLD / 2) * COS(144.0 / 180.0 * M_PI));
    DRAWLINE(IMG, X0 - XT, Y0 - YT, X0 + XT, Y0 - YT, BOLD / 2, CLR2, MASK);
    DRAWLINE(IMG, X0 + XT, Y0 - YT, X0 - XB, Y0 - YB, BOLD / 2, CLR2, MASK);
    DRAWLINE(IMG, X0 - XB, Y0 - YB, X0, Y0 - R - BOLD / 2, BOLD / 2, CLR2, MASK);
    DRAWLINE(IMG, X0, Y0 - R - BOLD / 2, X0 + XB, Y0 - YB, BOLD / 2, CLR2, MASK);
    DRAWLINE(IMG, X0 + XB, Y0 - YB, X0 - XT, Y0 - YT, BOLD / 2, CLR2, MASK);
}

INT PTRINSIDE(INT X0, INT Y0, INT PTR1[], INT PTR2[], INT PTR3[]){
    INT SIZE1 = (PTR1[0] - X0) * (PTR2[1] - PTR1[1]) - (PTR2[0] - PTR1[0]) * (PTR1[1] - Y0),
    SIZE2 = (PTR2[0] - X0) * (PTR3[1] - PTR2[1]) - (PTR3[0] - PTR2[0]) * (PTR2[1] - Y0),
    SIZE3 = (PTR3[0] - X0) * (PTR1[1] - PTR3[1]) - (PTR1[0] - PTR3[0]) * (PTR3[1] - Y0);
    RETURN (SIZE1 > 0 && SIZE2 > 0 && SIZE3 > 0) || (SIZE1 < 0 && SIZE2 < 0 && SIZE3 < 0);
}
```

```
VOID FILLING(PNG *IMG, INT X0, INT Y0, PNG_BYTEP COLOR, INT **MASK){
   INT X = X0, Y = Y0;
   IF(MASK[Y][X])
      RETURN;
   INT **STACK = (INT **) MALLOC(SIZEOF(INT *) * IMG->HEIGHT * IMG->WIDTH);
   INT IDX = 0;
   STACK[IDX++] = (INT *) MALLOC(SIZEOF(INT) * 4);
   STACK[IDX - 1][0] = X, STACK[IDX - 1][1] = X, STACK[IDX - 1][2] = Y, STACK[IDX - 1][3] = 1;
   STACK[IDX++] = (INT *) MALLOC(SIZEOF(INT) * 4);
   STACK[IDX - 1][0] = X, STACK[IDX - 1][1] = X, STACK[IDX - 1][2] = Y - 1, STACK[IDX - 1][3] = -1;
   WHILE(IDX > 0){
      Y = STACK[IDX - 1][2];
      INT X1 = STACK[IDX - 1][0], X2 = STACK[IDX - 1][1], DY = STACK[IDX - 1][3];
      X = X1;
      FREE(STACK[--IDX]);
      IF(!MASK[Y][X])
         WHILE(!MASK[Y][X - 1]) {
            DRAWPIXEL(IMG, X - 2, Y - 1, COLOR, MASK);
            X--;
         }
      IF(X < X1){
         STACK[IDX++] = (INT *)MALLOC(SIZEOF(INT) * 4);
         STACK[IDX - 1][0] = X, STACK[IDX - 1][1] = X1 - 1, STACK[IDX - 1][2] = Y - DY, STACK[IDX - 1][3] = -DY;
      }
      WHILE(X1 <= X2){
         WHILE(!MASK[Y][X1]){
            DRAWPIXEL(IMG, X1 - 1, Y - 1, COLOR, MASK);
            X1++;
            STACK[IDX++] = (INT *)MALLOC(SIZEOF(INT) * 4);
            STACK[IDX - 1][0] = X, STACK[IDX - 1][1] = X1 - 1, STACK[IDX - 1][2] = Y + DY, STACK[IDX - 1][3] = DY;
            IF(X1 - 1 > X2){
               STACK[IDX++] = (INT *)MALLOC(SIZEOF(INT) * 4);
               STACK[IDX - 1][0] = X2 + 1, STACK[IDX - 1][1] = X1 - 1, STACK[IDX - 1][2] = Y - DY, STACK[IDX - 1][3] = -DY;
            }
         }
         X1++;
         WHILE(X1 < X2 && MASK[Y][X1])
            X1++;
         X = X1;
      }
   }
   FREE(STACK);
}

VOID DRAWSQUARE(PNG *IMG, INT FILL, INT X0, INT Y0, INT X1, INT Y1, INT BOLD, PNG_BYTEP CLR_LINE, PNG_BYTEP
CLR_FILL, INT **MASK) {
   IF (BOLD > IMG->HEIGHT / 2 || BOLD > IMG->WIDTH / 2)
      RETURN;
   FOR (INT I = 0; I < BOLD; I++) {
      DRAWLINE(IMG, X0 - I, Y0 - I, X1 + I, Y0 - I, 1, CLR_LINE, MASK);
      DRAWLINE(IMG, X1 + I, Y0 - I, X1 + I, Y1 + I, 1, CLR_LINE, MASK);
```

```
        DRAWLINE(IMG, X1 + I, Y1 + I, X0 - I, Y1 + I, 1, CLR_LINE, MASK);
        DRAWLINE(IMG, X0 - I, Y1 + I, X0 - I, Y0 - I, 1, CLR_LINE, MASK);
    }

    IF (FILL)

        FOR (INT Y = Y0 + 1; Y < Y1; Y++) {
            FOR (INT X = X0 + 1; X < X1; X++){
                IF(X >= 0 && X < IMG->WIDTH && Y >= 0 && Y < IMG->HEIGHT && !MASK[Y][X]) {
                    FILLING(IMG, X, Y, CLR_FILL, MASK);
                    RETURN;
                }
        }
    }
}

VOID DRAWHEXAGON(PNG* IMG, INT TYPEIO, INT FILL, INT X0, INT Y0, INT X1, INT Y1, INT R, INT BOLD, PNG_BYTEP COLOR,
PNG_BYTEP CLR_FILL, INT **MASK) {
    //NEED SQUARE
    INT PTRS[6][2];
    IF(TYPEIO){
        R = (X1 - X0)/2;
        X1 -= R;
        Y1 -= R;
    }

    X0 = X1 + ((R + BOLD / 2) * COSL(30.0 / 180.0 * M_PI));
    Y0 = Y1 + ((R + BOLD / 2) * SINL(30.0 / 180.0 * M_PI));
    FOR (INT I = 1; I <= 6; I++) {
        DOUBLE ANGL = (60.0 * I + 30.0) / 180.0 * M_PI;
        DRAWLINE(IMG, X0, Y0, X1 + (R + BOLD / 2) * COSL(ANGL), Y1 + (R + BOLD / 2) * SINL(ANGL), BOLD, COLOR, MASK);
        X0 = X1 + ((R + BOLD / 2) * COSL(ANGL));
        Y0 = Y1 + ((R + BOLD / 2) * SINL(ANGL));
        PTRS[I - 1][0] = X0 - (BOLD / 2) * COSL(ANGL),
        PTRS[I - 1][1] = Y0 - (BOLD / 2) * SINL(ANGL);
    }
    IF(FILL)
        FOR(INT Y = PTRS[3][1] + 1; Y < PTRS[0][1]; Y++)
            FOR(INT X = PTRS[1][0] + 1; X < PTRS[4][0]; X++)
                FOR(INT I = 1; I < 5; I++)
                    IF(X >= 0 && Y >= 0 && X < IMG->WIDTH && Y < IMG->HEIGHT && !MASK[Y][X]
                    && PTRINSIDE(X, Y, PTRS[0], PTRS[I], PTRS[I + 1]))
                        FILLING(IMG, X, Y, CLR_FILL, MASK);


}



INT MAIN(INT ARGC, CHAR **ARGV){
    CONFIG VALUES = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 1, 0, "\0", "\0"};
    FOR(INT I = 0; I < 4; I++){
```

```
    VALUES.CLR_LINE[I] = 0;
    VALUES.CLR_OBJECT[I] = 0;
}
CHAR *OPTS = "HI:O:B:S:E:R:C:P:F:VLRPHSI";
STRUCT OPTION LONGOPTS[]={
        {"INPUT", REQUIRED_ARGUMENT, NULL, 'I'},
        {"OUTPUT", REQUIRED_ARGUMENT, NULL, 'O'},
        {"START", REQUIRED_ARGUMENT, NULL, 'S'},
        {"END", REQUIRED_ARGUMENT, NULL, 'E'},
        {"RADIUS", REQUIRED_ARGUMENT, NULL, 'R'},
        {"CENTER", REQUIRED_ARGUMENT, NULL, 'C'},
        {"COLOR", REQUIRED_ARGUMENT, NULL, 'P'},
        {"FILL", REQUIRED_ARGUMENT, NULL, 'F'},
        {"VERTICAL", NO_ARGUMENT, NULL, 'V'},
        {"HORIZONTAL", NO_ARGUMENT, NULL, 'L'},
        {"HELP", NO_ARGUMENT, NULL, 'H'},
        {"THICKNESS", REQUIRED_ARGUMENT, NULL, 'B'},
        {"REFLECT", NO_ARGUMENT, NULL, 'R'},
        {"SQUARE", NO_ARGUMENT, NULL, 'S'},
        {"HEXAGON", NO_ARGUMENT, NULL, 'H'},
        {"PENTAGRAM", NO_ARGUMENT, NULL, 'P'},
        {"INFORMATION", NO_ARGUMENT, NULL, 'I'}
};
INT LONG_IDX;
INT OPT = GETOPT_LONG(ARGC, ARGV, OPTS, LONGOPTS, &LONG_IDX);
IF(OPT == -1){
    PRINTF("НЕ ОБНАРУЖЕННО КОРРЕКТНЫХ ФЛАГОВ!\N");
    RETURN 0;
}
INT CODE_FUNC;
SWITCH(OPT){
    CASE 'R':
        VALUES.REF = 1;
        BREAK;
    CASE 'S':
        VALUES.SQR = 1;
        BREAK;
    CASE 'H':
        VALUES.HEX = 1;
        BREAK;
    CASE 'P':
        VALUES.PNT = 1;
        BREAK;
    CASE 'I':
        VALUES.INFORM = 1;
        BREAK;
    CASE 'H':
        PRINT_HELP();
        RETURN 0;
    DEFAULT:
        PRINTF("НЕ ОБНАРУЖЕНО ФЛАГА ФУНКЦИИ ОБРАБОТКИ ИЗОБРАЖЕНИЯ!\N");
        PRINTF("ДЛЯ КОРРЕКТНОЙ РАБОТЫ ПЕРВЫМ ФЛАГОМ СЛЕДУЕТ ПОДАВАТЬ ФЛАГ НА ТАКУЮ ФУНКЦИЮ!\N");
```

```
        RETURN 0;
    }
    OPT = GETOPT_LONG(ARGC, ARGV, OPTS, LONGOPTS,&LONG_IDX);
    WHILE(OPT != -1){
       CODE_FUNC = PROC_COMMS(&VALUES, OPT);
       IF(CODE_FUNC){
          SWITCH(CODE_FUNC){
             CASE 1:
                PRINTF("ПРОВЕРЬТЕ КОРРЕКТНОСТЬ ФОРМАТА ВВОДА.\N");
                RETURN 0;
             CASE 2:
                PRINTF("ДЛЯ ВЫЗВАННОЙ ФУНКЦИИ БЫЛИ ДОБАВЛЕНЫ ЛИШНИЕ АРГУМЕНТЫ!\N");
                RETURN 0;
             CASE 3:
                PRINTF("ОШИБКА ПРИ ПОПЫТКЕ ВЫЗВАТЬ НЕСКОЛЬКО ФУНКЦИЙ ОБРАБОТКИ ИЗОБРАЖЕНИЯ!\N");
                PRINTF("ДЛЯ КОРРЕКТНОЙ РАБОТЫ ПРОГРАММЫ ТРЕБУЕТСЯ ВВЕСТИ ФЛАГ ОДНОЙ ФУНКЦИИ!\N");
                RETURN 0;
             DEFAULT:
                RETURN 0;
          }
       }
       OPT = GETOPT_LONG(ARGC, ARGV, OPTS, LONGOPTS, &LONG_IDX);
    }
    IF(!VALUES.IN){
       PRINTF("НЕ УДАЛОСЬ ПОЛУЧИТЬ НАЗВАНИЕ СЧИТЫВАЕМОГО ФАЙЛА!\N");
       RETURN 0;
    }
    PNG *IMG = (PNG*)MALLOC(SIZEOF(PNG));
    CODE_FUNC = READ_PNG_FILE(VALUES.INPUT_FILE, IMG);
    IF(CODE_FUNC){
       FOR(INT Y = 0; Y < IMG->HEIGHT; Y++){
          FREE(IMG->ROW_POINTERS[Y]);
       }
       FREE(IMG->ROW_POINTERS);
      FREE(IMG);
      RETURN 0;
    }
    IF(VALUES.INFORM){
       INFORMATION(IMG);
       FOR(INT Y = 0; Y < IMG->HEIGHT; Y++){
          FREE(IMG->ROW_POINTERS[Y]);
       }
       FREE(IMG->ROW_POINTERS);
       FREE(IMG);
       RETURN 0;
    }
    IF(!VALUES.OUT){
       PRINTF("НЕ УДАЛОСЬ ПОЛУЧИТЬ НАЗВАНИЕ ФАЙЛА ДЛЯ ЗАПИСИ!\N");
       FOR(INT Y = 0; Y < IMG->HEIGHT; Y++){
          FREE(IMG->ROW_POINTERS[Y]);
       }
       FREE(IMG->ROW_POINTERS);
```

```
        FREE(IMG);
        RETURN 0;
    }
    INT **MASK = (INT **) MALLOC(SIZEOF(INT *) * (IMG->HEIGHT + 2));
    FOR(INT Y = 0; Y < (IMG->HEIGHT + 2); Y++){
        MASK[Y] = (INT *) CALLOC((IMG->WIDTH + 2), SIZEOF(INT));
        MASK[Y][0] = 1, MASK[Y][IMG->WIDTH + 1] = 1;
    }
    FOR(INT X = 0; X < (IMG->WIDTH + 2); X++){
        MASK[0][X] = 1, MASK[IMG->HEIGHT + 1][X] = 1;
    }

    IF(VALUES.REF){
        IF(VALUES.XL > VALUES.XR || VALUES.YL > VALUES.YR){
            PRINTF("ВВЕДЁННЫЕ ТОЧКИ НЕ МОГУТ ЯВЛЯТЬСЯ КООРДИНАТАМИ ЛЕВОГО ВЕРХНЕГО И ПРАВОГО НИЖНЕГО
УГЛА ПРЯМОУГОЛЬНИКА!\N");
            FOR(INT X = 0; X < IMG->HEIGHT + 2; X++)
                FREE(MASK[X]);
            FREE(MASK);
            FOR(INT Y = 0; Y < IMG->HEIGHT; Y++){
                FREE(IMG->ROW_POINTERS[Y]);
            }
            FREE(IMG->ROW_POINTERS);
            FREE(IMG);
            RETURN 0;
        }
        IF(VALUES.XL < 0 || VALUES.XR >= IMG->WIDTH || VALUES.YL < 0 || VALUES.YR >= IMG->HEIGHT){
            PRINTF("НЕВОЗМОЖНО ОТРАЗИТЬ ОБЛАСТЬ, ВЫХОДЯЩУЮ ЗА ПРЕДЕЛЫ ИЗОБРАЖЕНИЯ!\N");
            FOR(INT X = 0; X < IMG->HEIGHT + 2; X++)
                FREE(MASK[X]);
            FREE(MASK);
            FOR(INT Y = 0; Y < IMG->HEIGHT; Y++){
                FREE(IMG->ROW_POINTERS[Y]);
            }
            FREE(IMG->ROW_POINTERS);
            FREE(IMG);
            RETURN 0;
        }
        REFLECT(VALUES.DIR, VALUES.XL, VALUES.YL, VALUES.XR, VALUES.YR, IMG);
    }
    ELSE IF(VALUES.PNT){
        IF(VALUES.TYPE && (VALUES.XL > VALUES.XR || VALUES.YL > VALUES.YR || VALUES.XR - VALUES.XL != VALUES.YR -
VALUES.YL)){
            PRINTF("ВВЕДЁННЫЕ ТОЧКИ НЕ МОГУТ ЯВЛЯТЬСЯ КООРДИНАТАМИ ЛЕВОГО ВЕРХНЕГО И ПРАВОГО НИЖНЕГО
УГЛА КВАДРАТА!\N");
            FOR(INT X = 0; X < IMG->HEIGHT + 2; X++)
                FREE(MASK[X]);
            FREE(MASK);
            FOR(INT Y = 0; Y < IMG->HEIGHT; Y++){
                FREE(IMG->ROW_POINTERS[Y]);
            }
            FREE(IMG->ROW_POINTERS);
```

```
            FREE(IMG);
            RETURN 0;
        }
        DRAWPENTOGRAM(IMG, VALUES.TYPE, VALUES.XL, VALUES.YL,
                VALUES.XR, VALUES.YR, VALUES.R, VALUES.BOLD,
                VALUES.CLR_LINE, VALUES.CLR_OBJECT, MASK);
    }
    ELSE IF(VALUES.HEX){
        IF(VALUES.TYPE && (VALUES.XL > VALUES.XR || VALUES.YL > VALUES.YR || VALUES.XR - VALUES.XL != VALUES.YR -
VALUES.YL) ){
            PRINTF("ВВЕДЁННЫЕ ТОЧКИ НЕ МОГУТ ЯВЛЯТЬСЯ КООРДИНАТАМИ ЛЕВОГО ВЕРХНЕГО И ПРАВОГО НИЖНЕГО
УГЛА КВАДРАТА!\N");
            FOR(INT X = 0; X < IMG->HEIGHT + 2; X++)
                FREE(MASK[X]);
            FREE(MASK);
            FOR(INT Y = 0; Y < IMG->HEIGHT; Y++){
                FREE(IMG->ROW_POINTERS[Y]);
            }
            FREE(IMG->ROW_POINTERS);
            FREE(IMG);
            RETURN 0;
        }

        DRAWHEXAGON(IMG, VALUES.TYPE, VALUES.DIR, VALUES.XL, VALUES.YL,
                VALUES.XR, VALUES.YR, VALUES.R, VALUES.BOLD,
                VALUES.CLR_LINE, VALUES.CLR_OBJECT, MASK);
    }
    ELSE IF(VALUES.SQR){
        IF(VALUES.XL > VALUES.XR || VALUES.YL > VALUES.YR){
            PRINTF("ВВЕДЁННЫЕ ТОЧКИ НЕ МОГУТ ЯВЛЯТЬСЯ КООРДИНАТАМИ ЛЕВОГО ВЕРХНЕГО ИЛИ ПРАВОГО
НИЖНЕГО УГЛА ПРЯМОУГОЛЬНИКА!\N");
            FOR(INT X = 0; X < IMG->HEIGHT + 2; X++)
                FREE(MASK[X]);
            FREE(MASK);
            FOR(INT Y = 0; Y < IMG->HEIGHT; Y++){
                FREE(IMG->ROW_POINTERS[Y]);
            }
            FREE(IMG->ROW_POINTERS);
            FREE(IMG);
            RETURN 0;
        }
        DRAWSQUARE(IMG, VALUES.DIR, VALUES.XL, VALUES.YL, VALUES.XR, VALUES.YR,
                VALUES.BOLD, VALUES.CLR_LINE, VALUES.CLR_OBJECT, MASK);
    }
    WRITE_PNG_FILE(VALUES.OUTPUT_FILE, IMG);
    FOR(INT X = 0; X < IMG->HEIGHT + 2; X++)
        FREE(MASK[X]);
    FREE(MASK);
    FREE(IMG);
}
```