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

Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Работа с PNG изображениями в языке C

Студент гр. 9303 Жорже М.А.

Преподаватель Чайка К.В.

Санкт-Петербург

2020

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Жорже М.А.

Группа 9303

Тема работы: Работа с PNG изображениями в языке C

Исходные данные:

Программе на вход подаётся файл PNG.

Содержание пояснительной записки:

– Содержание

– Введение

– Описание работы программы

– Примеры работы программы

– Заключение

– Список использованных источников

– Исходный код программы

Предполагаемый объем пояснительной записки:

Не менее 50 страниц.

Дата выдачи задания: 01.03.2020

Дата сдачи реферата: 01.10.2020

Дата защиты реферата: 01.10.2020

Студент _____ Жорже М.А.

Преподаватель _____ Чайка К.В.

АННОТАЦИЯ

В данной работе реализована программа на языке программирования C, принимающая на вход файл с изображением формата PNG и изменяющая его в зависимости от использованных ключей и введённых значений. Функционал программы включает в себя рисование треугольника с заданными координатами, цветом и толщиной линий, цветом заливки; изменение цвета наибольшего прямоугольника; создание коллажа из одного изображения. Реализован CLI-интерфейс.

SUMMARY

In this work, a program was realized in the programming language C, which receives a file with an image in PNG format and changes it depending on the used keys and the entered values. The functionality of the program includes drawing a triangle with the given coordinates, color and thickness of lines, fill color; change the color of the largest rectangle; creation a collage from a single image. CLI interface was realized.

ВВЕДЕНИЕ

Программа должна реализовывать весь следующий функционал по обработке PNG-файла:

1. Рисование треугольника. Функционал определяется:
    a. координатами вершин треугольника;
    b. толщиной его линий;
    c. цветом его линий;
    d. цветом заливки.

2. Поиск самого большого прямоугольника заданного цвета и перекрашивание его в другой цвет. Функционал определяется:
    a. цветом, прямоугольник которого надо найти;
    b. цветом, в который надо его перекрасить.

3. Создание коллажа размера N x M из одного изображения, повторяющегося N x M раз. Функционал определяется:
    a. количеством изображений по “оси” Y;
    b. количеством изображений по “оси” X.

Цель работы – реализовать заданную задачу, закрепив знания, полученные во втором семестре по предмету «Программирование».

1. ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

1.1. Функция main()

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

        Exec(argc, argv);

        return 0;
}
```

В этой функции запускаем нашу программу, где команды, написанные в командной строке, считываются и отправляются в функции ‘Exec’ который отвечает за обработку команд в терминале.

1.2. Функция Exec(int argc, char* argv[])

```
void Exec(int argc, char * argv[]){
   struct option longOps[] = {
      {"triangle", required_argument, NULL, 't'},
      {"line" , required_argument, NULL, 'l'},
      {"fill", required_argument, NULL, 'f'},
      {"rectangle", required_argument, NULL, 'r'},
      {"collage", required_argument, NULL, 'c'},
      {"help", no_argument, NULL, 'h'},
      {0, 0, 0, 0}
   };
   int IndexLongOps;
   char fill = 'n';
   char old[10];
   char new[10];
   int *rgb_line = (int*)calloc(3, sizeof(int));
   int *rgb_fill = (int*)calloc(3, sizeof(int));
   int *rgb_old = (int*)calloc(3, sizeof(int));
   int *rgb_new = (int*)calloc(3, sizeof(int));
   int *coordinates = (int*)calloc(6, sizeof(int));
   int opt, isTriangle = 0, isRectangle = 0, isCollage = 0, n, m;

   while((opt = getopt_long(argc, argv, "t:l:f:r:c:h", longOps, &IndexLongOps)) > 0){
      switch(opt){
         case 't':
            if((sscanf(optarg, "(%d %d) (%d %d) (%d %d)", &coordinates[0], &coordinates[1],
&coordinates[2], &coordinates[3], &coordinates[4], &coordinates[5])) >= 6){
               isTriangle = 1;
            }else{
               show_help();
               exit(0);
            }
```

```
    break;

case 'l':
    if(setColors(rgb_line, optarg) < 0){
        fprintf(stderr, " Line color not found\n");
        show_colors();
        show_help();
        exit(0);
    }
    break;

case 'f':
    if(setColors(rgb_fill, optarg) < 0){
        fprintf(stderr, "Color to fill triangle not found\n");
        show_colors();
        show_help();
        exit(0);
    }
    fill = 'y';
    break;

case 'r':
    if((sscanf(optarg, "%s %s", old, new)) > 2){
        show_help();
        exit(0);
    }
    if(setColors(rgb_old, old) < 0){
        fprintf(stderr,"input rectangle's color not found\n");
        show_colors();
        show_help();
        exit(0);
    }
    if(setColors(rgb_new, new) < 0 < 0){
        fprintf(stderr,"output rectangle's color not found\n");
        show_colors();
        show_help();
        exit(0);
    }
    isRectangle = 1;
    break;

case 'c':
    if((sscanf(optarg, "%dx%d", &n, &m)) < 2){
        fprintf(stderr, "To make collage, you need add two values\n");
        show_help();
        exit(0);
    }
    isCollage = 1;
    break;
```

```
            default:
                show_help();
                exit(0);
        }
    }

    if(argv[optind] == NULL){
        fprintf(stderr, "maybe, you did not inform the input file name or/and output file name\n");
        show_help();
        exit(0);
    }

    int i = optind;
    char *file_in = argv[i++];
    char *file_out = argv[i];

    if(!file_in){
        fprintf(stderr,"check name file it\n");
        show_help();
        exit(0);
    }
    if(!file_out){
        fprintf(stderr,"check name file out\n");
        show_help();
        exit(0);
    }
    struct Png image;
    int check_file = 1;
    read_png_file(file_in, &image);

    if((check_file > 0) && (isTriangle > 0)){
            triangle(&image, rgb_line, rgb_fill, coordinates[0], coordinates[1], coordinates[2],
coordinates[3], coordinates[4], coordinates[5], fill);
        check_file = -1;
    }
    if((check_file > 0) && (isRectangle > 0)){
        rectangle(&image, rgb_old, rgb_new);
        check_file = -1;
    }
    if((check_file > 0) && (isCollage > 0)){
        collage(&image, n, m);
        check_file = -1;
    }

    printf("well done (*_*), check the file '%s'\n", file_out);
    write_png_file(file_out, &image);
    free(rgb_fill);
    free(rgb_line);
    free(coordinates);
}
```

В данной функции происходит считывание из командной строки данных для дальнейшего использования программой с помощью функции getopt_long() и оператора switch. В случае ввода ключей -t / --triangle, -r / --rectangle и -c / – collage выставляются соответствующие флаги и в дальнейшем выполняются соответствующие функции.

В случае ввода ключа --fill выставляется флаг ‘fill’, который понадобится в дальнейшем в функции drawTriangle(). В том случае, если какие-то из вводимых значений некорректны, то будет выброшено исключение и программа завершится.

1.3. **Функция печати справки**

```
void show_help(){
    fprintf(stderr, "\
            \n\
            -h, --help\t\t\t display help message and exit.\n\
            -t, --triangle\t\t run function to make a triagle.\n\
             -r, --rectangle\t\t run  function that looks for a rectangle with the largest area by the informed color and paint it with the new color.\n\
            -c, --collage\t\t run function that make collage.\n\
            \n\
            Example how to run traingle's function.\n\
            model: -t '(x0 y0) (x1 y1) (x2 y2)' -l 'color' -f 'color' file_in.png file_out.png\n\
            \t line: -t '(100 30) (50, 100) (600 450)' -l 'red' car.png res.png or --triangle '(100 30) (50, 100) (600 450)' -line 'red' car.png res.png\n\
               \t fill: -t '(100 30) (50, 100) (600 450)' -l 'red' -f 'black' car.png res.png or --triangle '(100 30) (50, 100) (600 450)' -line 'red' --fill 'black' car.png res.png\n\
            \n\
            Example how to run traingle's function.\n\
             model: -r 'old_color new_color' file_in.png file_out.png or --rectangle 'old_color new_color' file_in.png file_out.png.\n\
            \t   -r 'red black' car.png res.png  or --rectangle 'red black car.png res.png\n\
            \n\
            Example how to run collage function.\n\
                 model: -c 'linexcolunm' file_in.png file_out.png or --collage 'linexcolunm' file_in.png file_out.png.\n\
            \t   -c '3x5' car.png res.png or --collage '3x5' car.png res.png\n\
            \n");
}
```

В функции show_help() печатается справка, вызываемая в случае ключа -h, или –help.

1.4. **Функция проверки вводимого цвета**

```
int setColors(int *rgb, char *str){

   if(strcmp(str, "black") == 0){
      rgb[0] = rgb[1] = rgb[2] = 0;
      return 1;
   }
   if(strcmp(str, "white") == 0){
      rgb[0] = rgb[1] = rgb[2] = 255;
      return 1;
   }
   if(strcmp(str, "red") == 0){
      rgb[0] = 255; rgb[1] = rgb[2] = 0;
      return 1;
   }
   if(strcmp(str, "green") == 0){
      rgb[0] = 0; rgb[1] = 255; rgb[2] = 0;
      return 1;
   }
   if(strcmp(str, "blue") == 0){
      rgb[0] = rgb[1] = 0; rgb[2] = 255;
      return 1;
   }
   if(strcmp(str, "yellow") == 0){
      rgb[0] = rgb[1] = 255; rgb[2] = 0;
      return 1;
   }
   if(strcmp(str, "pink") == 0){
      rgb[0] = 255; rgb[1] = 193; rgb[2] = 203;
      return 1;
   }
   if(strcmp(str, "silver") == 0){
      rgb[0] = 192; rgb[1] = 192; rgb[2] = 192;
      return 1;
   }
   if(strcmp(str, "orange") == 0){
      rgb[0] = 1255; rgb[1] = 165; rgb[2] = 0;
      return 1;
   }
   if(strcmp(str, "gold") == 0){
      rgb[0] = 255; rgb[1] = 215; rgb[2] = 0;
      return 1;
   }
   if(strcmp(str, "purple") == 0){
      rgb[0] = 128; rgb[1] = 0; rgb[2] = 128;
      return 1;
```

```
    }
    if(strcmp(str, "teal") == 0){
        rgb[0] = 0; rgb[1] = 128; rgb[2] = 128;
        return 1;
    }
    if(strcmp(str, "brown") == 0){
        rgb[0] = 165; rgb[1] = 40; rgb[2] = 40;
        return 1;
    }
    if(strcmp(str, "magenta") == 0){
        rgb[0] = 255; rgb[1] = 0; rgb[2] = 255;
        return 1;
    }
    if(strcmp(str, "lime") == 0){
        rgb[0] = 50; rgb[1] = 110; rgb[2] = 50;
        return 1;
    }
    return -1;
}
```

В функции setColors() проверяется строка, введённая после ключа –line или --fill. Если она соответствует одному из допустимых цветов, то заполняется соответствующая массивы цвета. Если строка представляет собой код RGB,

В случае, если код или строка введены неверно, программа возвращает -1, что печатает сообщение об ошибке, справку печать функцию show_color() которая показывает все цвет в программе, и программа завершается.

1.5. Функция считывания файла

```
void read_png_file(char *file_name, struct Png *image){
    int x,y;
    char header[8];

    FILE *fp = fopen(file_name, "rb");
    if (!fp){
        fprintf(stderr, "file does not open, try again a png file\n");
        exit(1);
    }

    fread(header, 1, 8, fp);
    if (png_sig_cmp(header, 0, 8)){
        fprintf(stderr, "the input file is not a png file, please, try again with a png file\n");
        fclose(fp);
        exit(1);
    }

      image->png_ptr= png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL, NULL,
NULL);
```

```
        if (!image->png_ptr){
            fprintf(stderr, "Error to initialize the struct png_ptr\n");
            fclose(fp);
            exit(1);
        }

        image->info_ptr = png_create_info_struct(image->png_ptr);

        if (!image->info_ptr){
            fprintf(stderr, "Error to allocate the struct info_png\n");
            fclose(fp);
            exit(1);
        }

        if (setjmp(png_jmpbuf(image->png_ptr))){
            fprintf(stderr, "error, to allocate/iniciazate the structs\n");
            png_destroy_read_struct(&image->png_ptr, &image->info_ptr, NULL);
            fclose(fp);
            exit(1);
        }

        png_init_io(image->png_ptr, fp);
        png_set_sig_bytes(image->png_ptr, 8);

        png_read_info(image->png_ptr, image->info_ptr);

        image->width = png_get_image_width(image->png_ptr, image->info_ptr);
        image->height = png_get_image_height(image->png_ptr, image->info_ptr);
        image->color_type = png_get_color_type(image->png_ptr, image->info_ptr);
        image->bit_depth = png_get_bit_depth(image->png_ptr, image->info_ptr);

        image->number_of_passes = png_set_interlace_handling(image->png_ptr);
        png_read_update_info(image->png_ptr, image->info_ptr);

        if (setjmp(png_jmpbuf(image->png_ptr))){
            fprintf(stderr, "error, to allocate/iniciazate the structs\n");
            png_destroy_read_struct(&image->png_ptr, &image->info_ptr, NULL);
            fclose(fp);
            exit(1);
        }

        image->row_pointers = (png_bytep *) malloc(sizeof(png_bytep) * image->height);
        for (y = 0; y < image->height; y++)
                image->row_pointers[y] = (png_byte *) malloc(png_get_rowbytes(image->png_ptr,
image->info_ptr));

        png_read_image(image->png_ptr, image->row_pointers);

        fclose(fp);
    }
```

Данная функция readFile() производит считывание файла и заполнение структуры PNG с помощью функций библиотеки libpng. В случае какой-либо ошибки выводится соответствующее сообщение, функция возвращает 1, после чего программа завершается.

1.6. Функция записи файла

```
void write_png_file(char *file_name, struct Png *image) {
   int x,y;

   FILE *fp = fopen(file_name, "wb");
   if (!fp){
      fprintf(stderr, "file does not open, try again a png file\n");
      exit(1);
   }

    image->png_ptr = png_create_write_struct(PNG_LIBPNG_VER_STRING, NULL, NULL,
NULL);

   if (!image->png_ptr){
      fprintf(stderr, "Error to initialize the struct png_ptr\n");
      fclose(fp);
      exit(1);
   }

   image->info_ptr = png_create_info_struct(image->png_ptr);
   if (!image->info_ptr){
      fprintf(stderr, "Error to allocate the struct info_png\n");
      fclose(fp);
      exit(1);
   }

   if (setjmp(png_jmpbuf(image->png_ptr))){
      fprintf(stderr, "error, to allocate/iniciazate the structs\n");
      png_destroy_read_struct(&image->png_ptr, &image->info_ptr, NULL);
      fclose(fp);
      exit(1);
   }

   png_init_io(image->png_ptr, fp);


   if (setjmp(png_jmpbuf(image->png_ptr))){
      fprintf(stderr, "error during writing the image\n");
      png_destroy_read_struct(&image->png_ptr, &image->info_ptr, NULL);
      fclose(fp);
      exit(1);
   }

   png_set_IHDR(image->png_ptr, image->info_ptr, 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_ptr);

    if (setjmp(png_jmpbuf(image->png_ptr))){
        fprintf(stderr, "error during writing bytes\n");
        png_destroy_read_struct(&image->png_ptr, &image->info_ptr, NULL);
        fclose(fp);
        exit(1);
    }

    png_write_image(image->png_ptr, image->row_pointers);

    if (setjmp(png_jmpbuf(image->png_ptr))){
        fprintf(stderr, "error during end of write\n");
        png_destroy_read_struct(&image->png_ptr, &image->info_ptr, NULL);
        fclose(fp);
        exit(1);
    }

    png_write_end(image->png_ptr, NULL);

    for (y = 0; y < image->height; y++)
        free(image->row_pointers[y]);
    free(image->row_pointers);

    fclose(fp);
}
```

Данная функция writeFile() производит запись файла из структуры PNG с помощью функций библиотеки libpng. В случае какой-либо ошибки выводится соответствующее сообщение, функция возвращает 1, после чего программа завершается.

1.7. Функция рисования линии

```
void drawLine(struct Png *image, int rgb[], int x0, int y0, int x1, int y1, int typecolor){

    int j = x0, i = y0;
    int dx = abs(x1 - x0), dy = abs(y1 - y0);
    int controlx = (x0 < x1)? 1:-1, controly = (y0 < y1)? 1:-1;
    int err = dx - dy;
    int pixels = 4;

    for( ;( j != x1 || i != y1 ); ){

        png_bytep row  = image->row_pointers[i];
        png_bytep ptr = &(row[j * typecolor]);
        settingsColors(ptr, rgb, typecolor);

        for (int k = i - pixels; k < i+ pixels; k++){
```

```
            row  = image->row_pointers[k];
            for (int l = j - pixels; l < j + pixels; l++){
                ptr = &(row[l * typecolor]);
                settingsColors(ptr, rgb, typecolor);
            }
        }

        if (err * 2 > -dy){
            err -= dy;
            j += controlx;
        }
        else if (err * 2 < dx){
            err += dx;
            i += controly;
        }
    }
}
```

Данная функция drawLine() рисует линию с помощью алгоритма Брезенхэма из координаты (x1; y1) в (x2; y2) цветом line.

1.8. Функция заливки области

void filltriangle(struct Png *image, int rgb[], int x0, int y0, int x1, int y1, int x2, int y2, int typecolor){

```
    if ( x0 >= x1 && x0 >= x2 ){
        if (x1 >= x2)
            organizar(&x1, &y1, &x0, &y0);
        else
            organizar(&x2, &y2, &x0, &y0);
    }
    else if (x1 >= x0 && x1 >= x2){
        if (x2 >= x0)
            organizar(&x2, &y2, &x0, &y0);
    }
    else if (x1 >= x0)
        organizar(&x1, &y1, &x0, &y0);

    if (y0 >= y1 && y0 >= y2){
        if (y1 <= y2)
            organizar(&x1, &y1, &x2, &y2);
    }else if (y1 >= y0 && y1 >= y2)
        organizar(&x2, &y2, &x1, &y1);

    int j = x0, i = y0;
    int dx = abs(x1 - x0), dy = abs(y1 - y0);
    int controlx = (x0 < x1) ? 1 : -1, controly = (y0 < y1) ? 1 : -1;
    int err = dx - dy;

    int j1 = x0, i1 = y0;
```

```
int dx1 = abs(x2 - x0), dy1 = abs(y2 - y0);
int controlx1 = (x0 < x2) ? 1 : -1, controly1 = (y0 < y2) ? 1 : -1;
int err1 = dx1 - dy1;
int k = 0, aum = 0, cont = 0 , k1 = 0, k2 = 0;

for (; (i1 != y2 || j1 != x2);){
   if (j > j1)
      aum = -1;
   else
      aum = 1;
   k = j;
   while (k != j1){
      png_byte *row = image->row_pointers[i1];
      png_byte *ptr = &(row[k * typecolor]);
      settingsColors(ptr, rgb, typecolor);
      k += aum;
   }

   if(i > i1)
      k1 = 1;
   else if(i1 > i)
      k2 = 1;
   else
      k2 = 0; k1 =0;

   if(k1 == 0){
      if (err * 2 > -dy){
         err -= dy;
         j += controlx;
      }
      else if (err * 2 < dx){
         err += dx;
         i += controly;
      }
   }

   if(k2 == 0){
      if (err1 * 2 > -dy1){
         err1 -= dy1;
         j1 += controlx1;
      }
      else if (err1 * 2 < dx1){
         err1 += dx1;
         i1 += controly1;
      }
   }

   if((i == y1) && (cont == 0)){
      cont = 2;
      j = x1, i = y1;
```

```
            dx = abs(x2 - x1), dy = abs(y2 - y1);
            controlx = (x1 < x2) ? 1 : -1, controly = (y1 < y2) ? 1 : -1;
            err = dx - dy;
          }
        }
      }
```

В функции filltriangle() закрашивается выбранная точка внутри треугольника, после чего функция вызывает '**organizar**', '**settingColors**' кторые помогают перекресить всё треугольника.

1.9. Функция рисования треугольника

```
void triangle(struct Png *image, int rgb_line[], int rgb_fill[], int x0, int y0, int x1, int y1, int x2, int y2, char fill){

      /* int x0 = 20, y0 = 400;
         int x1 = 350, y1 = 550;
         int x2 = 250, y2 = 130;

         int x0 = 100, y0 = 30;
         int x1 = 50, y1 = 100;
         int x2 = 600, y2 = 450;

         int x0 = 690, y0 = 30;
         int x1 = 30, y1 = 250;
         int x2 = 750, y2 = 690;*/

          if((x0 > image->width || x1 > image->width || x2 > image->width) || (y0 > image->height || y1 > image->height || y2 > image->height)){
              fprintf(stderr, "the cordenates of triangle, must be between width and height of input file\n");
            exit(1);
          }

          if(x0 < 0 || x1 < 1 || x2 < 0 || y0 < 0 || y1 < 0 || y2 < 0){
            fprintf(stderr, "The cordenates of triangle must be > 0\n");
            exit(1);
          }

          int typecolor;

          if(png_get_color_type(image->png_ptr, image->info_ptr) == PNG_COLOR_TYPE_RGB){
            typecolor = 3;
                        }else if(png_get_color_type(image->png_ptr, image->info_ptr) == PNG_COLOR_TYPE_RGBA){
            typecolor = 4;
          }else{
            fprintf(stderr, "The photo, must be to have a color type RGB or RGBA, typecolor\n");
            exit(1);
          }
```

```
    drawLine(image, rgb_line, x0, y0, x1, y1, typecolor);
    drawLine(image, rgb_line, x2, y2, x1, y1, typecolor);
    drawLine(image, rgb_line, x0, y0, x2, y2, typecolor);

    if(fill == 'y' || fill == 'Y')
      filltriangle(image,rgb_fill, x0, y0, x1, y1, x2, y2, typecolor);
  }
```

В функции drawTriangle() трижды вызывается функция рисования линии и, в случае, если был введён ключ --fill, соответственно, флаг flag fill равен ‘y’ или ‘Y’ , то выбирается рандомная точка внутри треугольника и вызывается функция заливки области для неё.

1.10. Функция, которая находит самый большой прямоугольник

```
void rectangle(struct Png *image, int rgb_old[], int rgb_new[]){

    int typecolor;
                    if(png_get_color_type(image->png_ptr, image->info_ptr) == PNG_COLOR_TYPE_RGBA)
      typecolor = 4;
                else if (png_get_color_type(image->png_ptr, image->info_ptr) == PNG_COLOR_TYPE_RGB)
      typecolor = 3;
    else{
      fprintf(stderr, "The photo, must be to have a color type RGB or RGBA, typecolor\n");
      exit(1);
    }

    int A[2] = {0};
    int height = 0, h,hf = 0, heightf  = 0 , lengthk = 0 , lengthj = 0;

    int k = 0, contador = 0;
    for(int i = 0; i < image->height; i++){
      png_bytep row = image->row_pointers[i];
      for(int j = 0; j < image->width; j++){
        png_bytep ptr = &(row[j * typecolor]);
        if(rgb_old[0] == ptr[0] && rgb_old[1] == ptr[1] && rgb_old[2] == ptr[2]){
          k = j;
           while((rgb_old[0] == ptr[0] && rgb_old[1] == ptr[1] && rgb_old[2] == ptr[2]) && (k < image->width)){
            k++;
            ptr = &row[k * typecolor];
          }

          int q = 0;
            if((j != lengthj && k != lengthk) || (j == lengthj && k != lengthk) /*|| (j != lengthj && k == lengthk)*/){
              for(int t = i + 1; t < image->height; t++){
                row = image->row_pointers[t];
                for(int cont = j; cont < k; cont++){
                  ptr = &(row[cont * typecolor]);
```

```
                        if(!(rgb_old[0] == ptr[0] && rgb_old[1] == ptr[1] && rgb_old[2] == ptr[2]))
{
                            h = i; hf = t;
                            t = image->height;
                            q = 1;
                            break;
                        }
                    }
                }
                if(q == 0){
                    h = i; hf = image->height-1;
                }

                if(A[0] == 0){
                    A[0] = (k - j - 1) * (hf - i -1);
                    lengthj = j;
                    lengthk = k;
                    heightf = hf;
                    height = i;

                }else {
                    int altura = 0;
                    if(j >= lengthj && lengthk > k)
                        altura = height;
                    else
                        altura = i;

                    A[1] = (k - j - 1) * (hf - altura - 1);
                    if(A[0] < A[1]){
                        heightf = hf;
                        A[0] = A[1];
                        A[1] = 0;
                        lengthj = j;
                        height = altura;
                        lengthk = k;
                    }
                }
            }j = k;
        }
    }
}

    for(int i = height; i < heightf; i++){
        png_bytep row = image->row_pointers[i];
        for(int j = lengthj; j < lengthk; j++){
            png_bytep ptr = &(row[j * typecolor]);
            settingsColors(ptr, rgb_new, typecolor);
        }
    }
}
```

Функция rectangle() находит самый большой прямоугольник заданного цвета rgb_old и меняет цвет всех пикселей на rgb_new. В случае, если прямоугольник не найден.

1.13. Функция создания коллажа

```
void collage(struct Png *image, int m, int n){
   int height = image->height;
   int width = image->width;

   int typecolor;

                              if(png_get_color_type(image->png_ptr, image->info_ptr) ==
PNG_COLOR_TYPE_RGBA)
      typecolor = 4;
                    else if (png_get_color_type(image->png_ptr, image->info_ptr) ==
PNG_COLOR_TYPE_RGB)
      typecolor = 3;
   else{
      fprintf(stderr, "error\n");
      exit(0);
   }

   png_bytep *rowPointers = (png_bytep *)malloc(height * sizeof(png_bytep));
   for (int y = 0; y < height; y++)
   {
      rowPointers[y] = (png_byte *)malloc(width * typecolor * sizeof(png_byte));
      for (int x = 0; x < width * typecolor; x++)
         rowPointers[y][x] = image->row_pointers[y][x];
   }
   image->height *= m;
   image->width *= n;
   image->row_pointers = (png_bytep *)calloc(image->height, sizeof(png_bytep));

   for (int y = 0; y < height; y++)
                    image->row_pointers[y] = (png_byte *)calloc(image->width * typecolor,
sizeof(png_byte));
   for (int y = height; y < image->height; y++)
                  image->row_pointers[y] = (png_byte *)malloc(image->width * typecolor *
sizeof(png_byte));
   int x = 0, y = 0;
   for (int j = 0; j < m; j++)
      for (int i = 0; i < n; i++)
         for (int y = 0; y < height; y++)
            for (int x = 0; x < width * typecolor; x++)
                        image->row_pointers[y + (j * height)][x + (i * width * typecolor)] =
rowPointers[y][x];
   for (int y = 0; y < height; y++)
      free(rowPointers[y]);
   free(rowPointers);
}
```

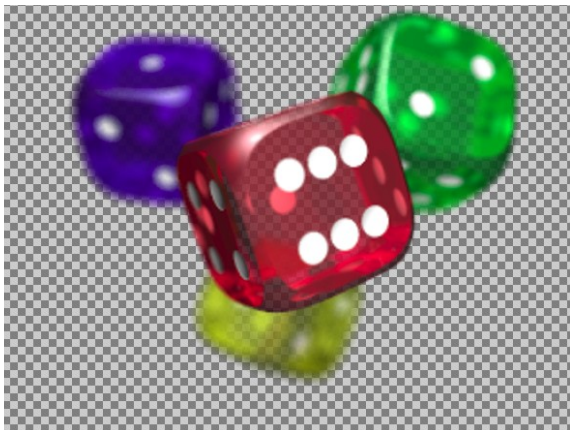
В функции makeCollage() создаётся буфер под уже имеющееся изображение, туда сохраняется изображение. Затем старый буфер увеличивается в длину в N раз, а в высоту – в M. В расширенный буфер помещается старое изображение N x M раз. После всего, выделенный буфер отчищается.

2. ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

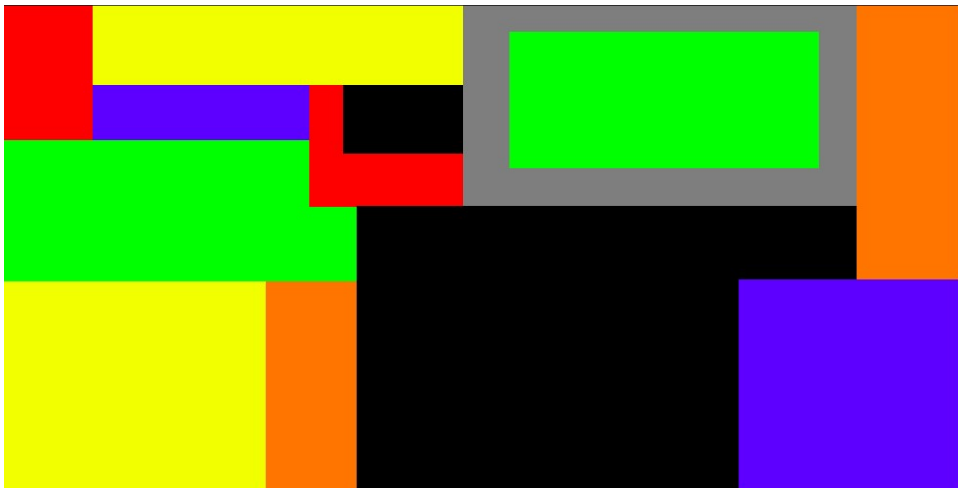
2.1. Примеры корректной работы программы

Входные данные:

1. input1.png



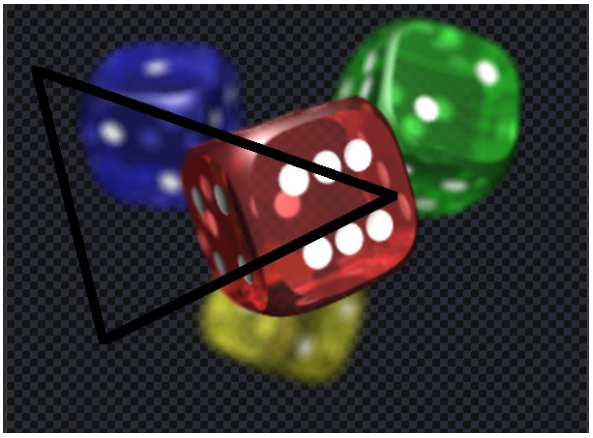
2. input2.png



Примеры использования программы:

1. Рисование треугольника с координатами (30; 70) (100; 350) (400; 200), цевет “balck”:

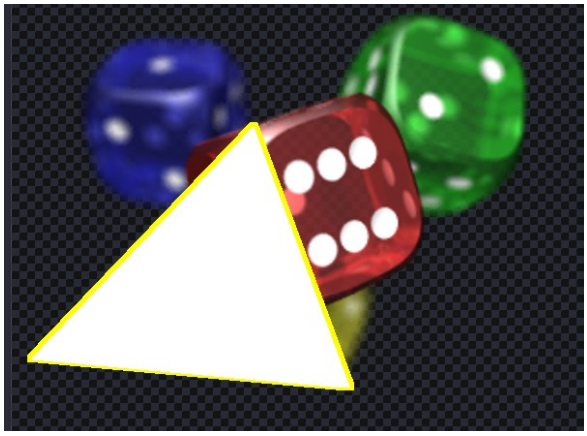
```
gcc main.c -o main -lpng
./main --triangle "(30 70) (100 350) (400 200)" -line "black" test.png out.png
```



2. Нарисуйте треугольник с координатами (20; 370) (350; 400) (250; 130), с цветом линии “желтый” и заполненным цветом “белый”.

gcc main.c -o main -lpng

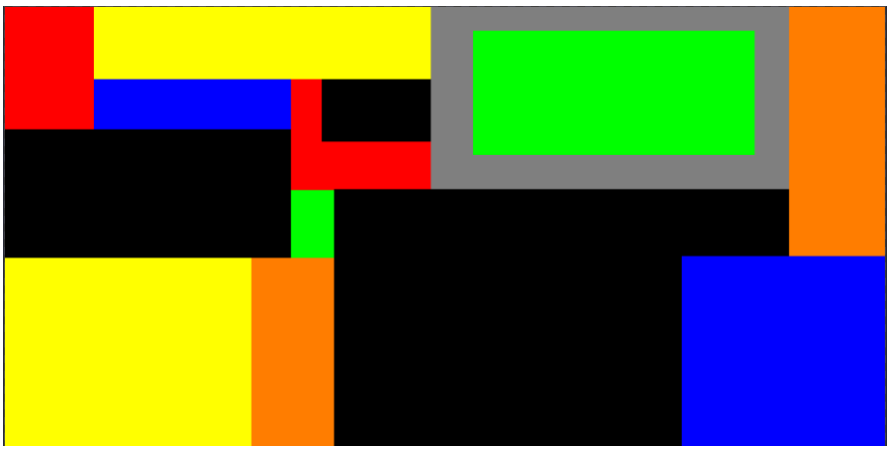
./main --triangle "(20 370) (350 400) (250 130)" --line "yellow" --fill "white" test.png out.png



3. Замена цвета у самого большого зеленого прямоугольника на черный:

gcc main.c -o main -lpng

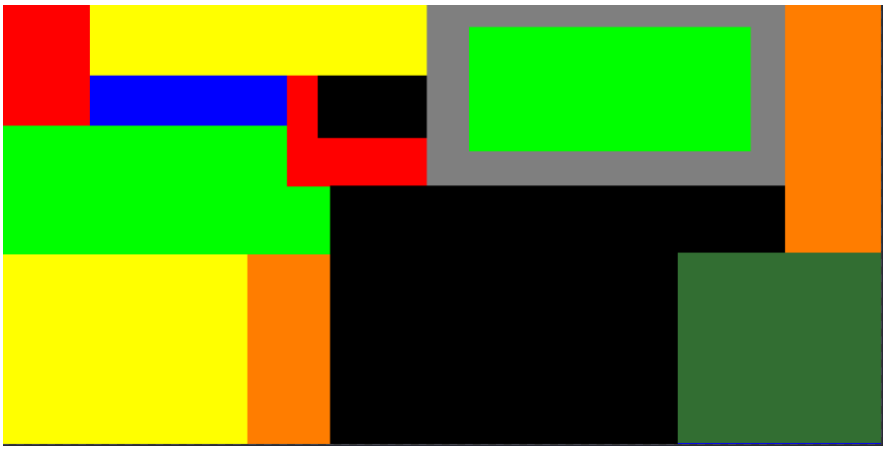
./main --rectangle "green black" input2.png out.png



4. Замена цвета у самого большого синего прямоугольника на Лаймого:

gcc main.c -o main -lpng

./main --rectangle "blue lime" input2.png out.png



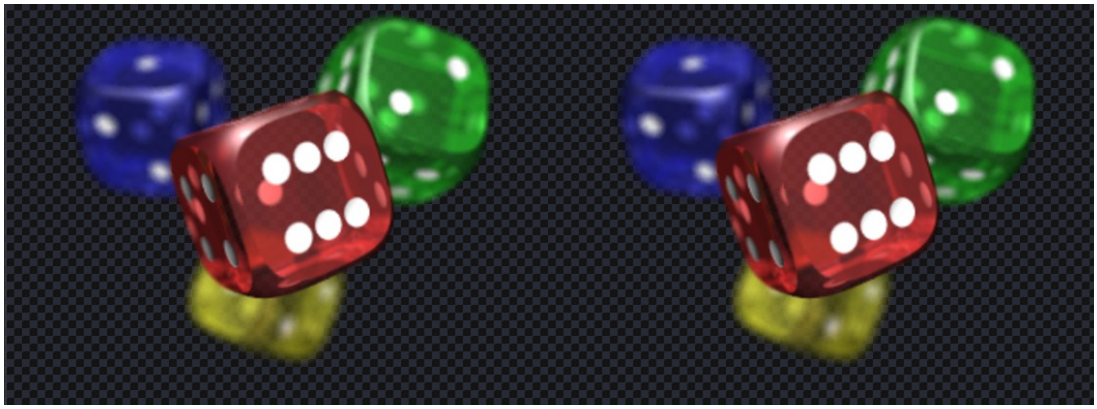
5. Создание коллажа размера 4 на 4 из одной фотографии:

gcc main.c -o main -lpng

./cw --collage 3x4 test.png out.png

6. Создание коллажа размера 4 на 4 из одной фотографии:

```
gcc main.c -o main -lpng
./cw --collage 1x2 test.png out.png
```



ЗАКЛЮЧЕНИЕ

В ходе работы были закреплены практические навыки в работе с изображениями в формате PNG. Была разработана программа с консольным интерфейсом CLI, которая позволяет обрабатывать PNG изображения: рисовать треугольник по заданным координатам, менять цвет самого большого прямоугольника и создавать коллаж из одного изображения.

ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ

main.c

```
#include <stdio.h>
#include "main.h"

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

   Exec(argc, argv);

   return 0;
}
```

main.h

```
#include <unistd.h>
#include <stdlib.h>
#include <getopt.h>
#include <string.h>
#include <png.h>

struct Png{
   int width, height;
   png_byte color_type;
   png_byte bit_depth;
   png_structp png_ptr;
   png_infop info_ptr;
   int number_of_passes;
   png_bytep *row_pointers;
};

void triangle(struct Png *image, int rgb_line[], int rgb_fill[], int x0, int y0, int x1, int y1, int x2, int y2,
char fill);
void filltriangle(struct Png *image, int rgb[], int x0, int y0, int x1, int y1, int x2, int y2, int typecolor);
void drawLine(struct Png *image, int rgb[], int x0, int y0, int x1, int y1, int typecolor);
void rectangle(struct Png *image, int rgb_old[], int rgb_new[]);
void settingsColors(png_bytep ptr, int rgb[], int typecolor);
void write_png_file(char *file_name, struct Png *image);
void read_png_file(char *file_name, struct Png *image);
void organizar(int *x0, int *y0, int *x1, int *y1);
void collage(struct Png *image, int m, int n);
int setColors(int *rgba, char *str);
void Exec(int argc, char * argv[]);
int isColor(char * str);
void show_colors();
void show_help();
```

```
void read_png_file(char *file_name, struct Png *image){
   int x,y;
   char header[8];

   FILE *fp = fopen(file_name, "rb");
   if (!fp){
      fprintf(stderr, "file does not open, try again a png file\n");
      exit(1);
   }

   fread(header, 1, 8, fp);
   if (png_sig_cmp(header, 0, 8)){
      fprintf(stderr, "the input file is not a png file, please, try again with a png file\n");
      fclose(fp);
      exit(1);
   }

   image->png_ptr= png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL, NULL, NULL);

   if (!image->png_ptr){
      fprintf(stderr, "Error to initialize the struct png_ptr\n");
      fclose(fp);
      exit(1);
   }

   image->info_ptr = png_create_info_struct(image->png_ptr);

   if (!image->info_ptr){
      fprintf(stderr, "Error to allocate the struct info_png\n");
      fclose(fp);
      exit(1);
   }

   if (setjmp(png_jmpbuf(image->png_ptr))){
      fprintf(stderr, "error, to allocate/iniciazate the structs\n");
      png_destroy_read_struct(&image->png_ptr, &image->info_ptr, NULL);
      fclose(fp);
      exit(1);
   }

   png_init_io(image->png_ptr, fp);
   png_set_sig_bytes(image->png_ptr, 8);

   png_read_info(image->png_ptr, image->info_ptr);

   image->width = png_get_image_width(image->png_ptr, image->info_ptr);
   image->height = png_get_image_height(image->png_ptr, image->info_ptr);
   image->color_type = png_get_color_type(image->png_ptr, image->info_ptr);
   image->bit_depth = png_get_bit_depth(image->png_ptr, image->info_ptr);
```

```
   image->number_of_passes = png_set_interlace_handling(image->png_ptr);
   png_read_update_info(image->png_ptr, image->info_ptr);

   if (setjmp(png_jmpbuf(image->png_ptr))){
      fprintf(stderr, "error, to allocate/iniciazate the structs\n");
      png_destroy_read_struct(&image->png_ptr, &image->info_ptr, NULL);
      fclose(fp);
      exit(1);
   }

   image->row_pointers = (png_bytep *) malloc(sizeof(png_bytep) * image->height);
   for (y = 0; y < image->height; y++)
         image->row_pointers[y] = (png_byte *) malloc(png_get_rowbytes(image->png_ptr, image->info_ptr));

   png_read_image(image->png_ptr, image->row_pointers);

   fclose(fp);
}

void write_png_file(char *file_name, struct Png *image) {
   int x,y;

   FILE *fp = fopen(file_name, "wb");
   if (!fp){
      fprintf(stderr, "file does not open, try again a png file\n");
      exit(1);
   }

      image->png_ptr = png_create_write_struct(PNG_LIBPNG_VER_STRING, NULL, NULL, NULL);

   if (!image->png_ptr){
      fprintf(stderr, "Error to initialize the struct png_ptr\n");
      fclose(fp);
      exit(1);
   }

   image->info_ptr = png_create_info_struct(image->png_ptr);
   if (!image->info_ptr){
      fprintf(stderr, "Error to allocate the struct info_png\n");
      fclose(fp);
      exit(1);
   }

   if (setjmp(png_jmpbuf(image->png_ptr))){
      fprintf(stderr, "error, to allocate/iniciazate the structs\n");
      png_destroy_read_struct(&image->png_ptr, &image->info_ptr, NULL);
      fclose(fp);
      exit(1);
```

```
    }

    png_init_io(image->png_ptr, fp);


    if (setjmp(png_jmpbuf(image->png_ptr))){
        fprintf(stderr, "error during writing the image\n");
        png_destroy_read_struct(&image->png_ptr, &image->info_ptr, NULL);
        fclose(fp);
        exit(1);
    }

    png_set_IHDR(image->png_ptr, image->info_ptr, 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_ptr);


    if (setjmp(png_jmpbuf(image->png_ptr))){
        fprintf(stderr, "error during writing bytes\n");
        png_destroy_read_struct(&image->png_ptr, &image->info_ptr, NULL);
        fclose(fp);
        exit(1);
    }

    png_write_image(image->png_ptr, image->row_pointers);

    if (setjmp(png_jmpbuf(image->png_ptr))){
        fprintf(stderr, "error during end of write\n");
        png_destroy_read_struct(&image->png_ptr, &image->info_ptr, NULL);
        fclose(fp);
        exit(1);
    }

    png_write_end(image->png_ptr, NULL);

    for (y = 0; y < image->height; y++)
        free(image->row_pointers[y]);
    free(image->row_pointers);

    fclose(fp);
}

void rectangle(struct Png *image, int rgb_old[], int rgb_new[]){

    int typecolor;
    if(png_get_color_type(image->png_ptr, image->info_ptr) == PNG_COLOR_TYPE_RGBA)
        typecolor = 4;
    else if (png_get_color_type(image->png_ptr, image->info_ptr) == PNG_COLOR_TYPE_RGB)
```

```
        typecolor = 3;
    else{
        fprintf(stderr, "The photo, must be to have a color type RGB or RGBA, typecolor\n");
        exit(1);
    }

    int A[2] = {0};
    int height = 0, h,hf = 0, heightf  = 0 , lengthk = 0 , lengthj = 0;

    int k = 0, contador = 0;
    for(int i = 0; i < image->height; i++){
        png_bytep row = image->row_pointers[i];
        for(int j = 0; j < image->width; j++){
            png_bytep ptr = &(row[j * typecolor]);
            if(rgb_old[0] == ptr[0] && rgb_old[1] == ptr[1] && rgb_old[2] == ptr[2]){
                k = j;
                  while((rgb_old[0] == ptr[0] && rgb_old[1] == ptr[1] && rgb_old[2] == ptr[2]) && (k <
image->width)){
                    k++;
                    ptr = &row[k * typecolor];
                }

                int q = 0;
                 if((j != lengthj && k != lengthk) || (j == lengthj && k != lengthk) /*|| (j != lengthj && k ==
lengthk)*/){
                    for(int t = i + 1; t < image->height; t++){
                        row = image->row_pointers[t];
                        for(int cont = j; cont < k; cont++){
                            ptr = &(row[cont * typecolor]);
                            if(!(rgb_old[0] == ptr[0] && rgb_old[1] == ptr[1] && rgb_old[2] == ptr[2])){
                                h = i; hf = t;
                                t = image->height;
                                q = 1;
                                break;
                            }
                        }
                    }
                    if(q == 0){
                         h = i; hf = image->height-1;
                    }

                    if(A[0] == 0){
                        A[0] = (k - j - 1) * (hf - i -1);
                        lengthj = j;
                        lengthk = k;
                        heightf = hf;
                        height = i;

                    }else {
                        int altura = 0;
```

```
                if(j >= lengthj && lengthk > k)
                    altura = height;
                else
                    altura = i;

                A[1] = (k - j - 1) * (hf - altura - 1);
                if(A[0] < A[1]){
                    heightf = hf;
                    A[0] = A[1];
                    A[1] = 0;
                    lengthj = j;
                    height = altura;
                    lengthk = k;
                }
            }
        }j = k;
    }
  }
}

  for(int i = height; i < heightf; i++){
    png_bytep row = image->row_pointers[i];
    for(int j = lengthj; j < lengthk; j++){
      png_bytep ptr = &(row[j * typecolor]);
      settingsColors(ptr, rgb_new, typecolor);
    }
  }
}

void drawLine(struct Png *image, int rgb[], int x0, int y0, int x1, int y1, int typecolor){

  int j = x0, i = y0;
  int dx = abs(x1 - x0), dy = abs(y1 - y0);
  int controlx = (x0 < x1)? 1:-1, controly = (y0 < y1)? 1:-1;
  int err = dx - dy;
  int pixels = 4;

  for( ;( j != x1 || i != y1 ); ){

    png_bytep row  = image->row_pointers[i];
    png_bytep ptr = &(row[j * typecolor]);
    settingsColors(ptr, rgb, typecolor);

    for (int k = i - pixels; k < i+ pixels; k++){
      row  = image->row_pointers[k];
      for (int l = j - pixels; l < j + pixels; l++){
        ptr = &(row[l * typecolor]);
        settingsColors(ptr, rgb, typecolor);
      }
    }
```

```
        if (err * 2 > -dy){
            err -= dy;
            j += controlx;
        }
        else if (err * 2 < dx){
            err += dx;
            i += controly;
        }
    }
}

void organizar(int *x0, int *y0, int *x1, int *y1){
    *x0 += *x1;
    *x1 = *x0 - *x1;
    *x0 = *x0 - *x1;

    *y0 += *y1;
    *y1 = *y0 - *y1;
    *y0 = *y0 - *y1;
}

void filltriangle(struct Png *image, int rgb[], int x0, int y0, int x1, int y1, int x2, int y2, int typecolor){

    if ( x0 >= x1 && x0 >= x2 ){
        if (x1 >= x2)
            organizar(&x1, &y1, &x0, &y0);
        else
            organizar(&x2, &y2, &x0, &y0);
    }
    else if (x1 >= x0 && x1 >= x2){
        if (x2 >= x0)
            organizar(&x2, &y2, &x0, &y0);
    }
    else if (x1 >= x0)
        organizar(&x1, &y1, &x0, &y0);

    if (y0 >= y1 && y0 >= y2){
        if (y1 <= y2)
            organizar(&x1, &y1, &x2, &y2);
    }else if (y1 >= y0 && y1 >= y2)
        organizar(&x2, &y2, &x1, &y1);

    int j = x0, i = y0;
    int dx = abs(x1 - x0), dy = abs(y1 - y0);
    int controlx = (x0 < x1) ? 1 : -1, controly = (y0 < y1) ? 1 : -1;
    int err = dx - dy;

    int j1 = x0, i1 = y0;
    int dx1 = abs(x2 - x0), dy1 = abs(y2 - y0);
```

```
int controlx1 = (x0 < x2) ? 1 : -1, controly1 = (y0 < y2) ? 1 : -1;
int err1 = dx1 - dy1;
int k = 0, aum = 0, cont = 0 , k1 = 0, k2 = 0;

for (; (i1 != y2 || j1 != x2);){
   if (j > j1)
      aum = -1;
   else
      aum = 1;
   k = j;
   while (k != j1){
      png_byte *row = image->row_pointers[i1];
      png_byte *ptr = &(row[k * typecolor]);
      settingsColors(ptr, rgb, typecolor);
      k += aum;
   }

   if(i > i1)
      k1 = 1;
   else if(i1 > i)
      k2 = 1;
   else
      k2 = 0; k1 =0;

   if(k1 == 0){
      if (err * 2 > -dy){
         err -= dy;
         j += controlx;
      }
      else if (err * 2 < dx){
         err += dx;
         i += controly;
      }
   }

   if(k2 == 0){
      if (err1 * 2 > -dy1){
         err1 -= dy1;
         j1 += controlx1;
      }
      else if (err1 * 2 < dx1){
         err1 += dx1;
         i1 += controly1;
      }
   }

   if((i == y1) && (cont == 0)){
      cont = 2;
      j = x1, i = y1;
      dx = abs(x2 - x1), dy = abs(y2 - y1);
```

```
            controlx = (x1 < x2) ? 1 : -1, controly = (y1 < y2) ? 1 : -1;
            err = dx - dy;
        }
    }
}

void triangle(struct Png *image, int rgb_line[], int rgb_fill[], int x0, int y0, int x1, int y1, int x2, int y2,
char fill){
/* int x0 = 20, y0 = 400;
    int x1 = 350, y1 = 550;
    int x2 = 250, y2 = 130;

    int x0 = 100, y0 = 30;
    int x1 = 50, y1 = 100;
    int x2 = 600, y2 = 450;

    int x0 = 690, y0 = 30;
    int x1 = 30, y1 = 250;
    int x2 = 750, y2 = 690;*/

     if((x0 > image->width || x1 > image->width || x2 > image->width) || (y0 > image->height || y1 >
image->height || y2 > image->height)){
        fprintf(stderr, "the cordenates of triangle, must be between width and height of input file\n");
        exit(1);
    }

    if(x0 < 0 || x1 < 1 || x2 < 0 || y0 < 0 || y1 < 0 || y2 < 0){
        fprintf(stderr, "The cordenates of triangle must be > 0\n");
        exit(1);
    }

    int typecolor;

    if(png_get_color_type(image->png_ptr, image->info_ptr) == PNG_COLOR_TYPE_RGB){
        typecolor = 3;
    }else if(png_get_color_type(image->png_ptr, image->info_ptr) == PNG_COLOR_TYPE_RGBA){
        typecolor = 4;
    }else{
        fprintf(stderr, "The photo, must be to have a color type RGB or RGBA, typecolor\n");
        exit(1);
    }

    drawLine(image, rgb_line, x0, y0, x1, y1, typecolor);
    drawLine(image, rgb_line, x2, y2, x1, y1, typecolor);
    drawLine(image, rgb_line, x0, y0, x2, y2, typecolor);

    if(fill == 'y' || fill == 'Y')
        filltriangle(image,rgb_fill, x0, y0, x1, y1, x2, y2, typecolor);
}
```

```
void collage(struct Png *image, int m, int n){

    int height = image->height;
    int width = image->width;

    int typecolor;

    if(png_get_color_type(image->png_ptr, image->info_ptr) == PNG_COLOR_TYPE_RGBA)
        typecolor = 4;
    else if (png_get_color_type(image->png_ptr, image->info_ptr) == PNG_COLOR_TYPE_RGB)
        typecolor = 3;
    else{
        fprintf(stderr, "error\n");
        exit(0);
    }

    png_bytep *rowPointers = (png_bytep *)malloc(height * sizeof(png_bytep));
    for (int y = 0; y < height; y++)
    {
        rowPointers[y] = (png_byte *)malloc(width * typecolor * sizeof(png_byte));
        for (int x = 0; x < width * typecolor; x++)
            rowPointers[y][x] = image->row_pointers[y][x];
    }
    image->height *= m;
    image->width *= n;
    image->row_pointers = (png_bytep *)calloc(image->height, sizeof(png_bytep));

    for (int y = 0; y < height; y++)
        image->row_pointers[y] = (png_byte *)calloc(image->width * typecolor, sizeof(png_byte));
    for (int y = height; y < image->height; y++)
        image->row_pointers[y] = (png_byte *)malloc(image->width * typecolor * sizeof(png_byte));
    int x = 0, y = 0;
    for (int j = 0; j < m; j++)
        for (int i = 0; i < n; i++)
            for (int y = 0; y < height; y++)
                for (int x = 0; x < width * typecolor; x++)
                    image->row_pointers[y + (j * height)][x + (i * width * typecolor)] = rowPointers[y][x];
    for (int y = 0; y < height; y++)
        free(rowPointers[y]);
    free(rowPointers);
}

void settingsColors(png_bytep ptr, int rgb[], int typecolor){
    ptr[0] = rgb[0]; ptr[1] = rgb[1]; ptr[2] = rgb[2];
    if(typecolor == 4)
        ptr[3] = 255;
}

int setColors(int *rgb, char *str){
```

```
if(strcmp(str, "black") == 0){
    rgb[0] = rgb[1] = rgb[2] = 0;
    return 1;
}
if(strcmp(str, "white") == 0){
    rgb[0] = rgb[1] = rgb[2] = 255;
    return 1;
}
if(strcmp(str, "red") == 0){
    rgb[0] = 255; rgb[1] = rgb[2] = 0;
    return 1;
}
if(strcmp(str, "green") == 0){
    rgb[0] = 0; rgb[1] = 255; rgb[2] = 0;
    return 1;
}
if(strcmp(str, "blue") == 0){
    rgb[0] = rgb[1] = 0; rgb[2] = 255;
    return 1;
}
if(strcmp(str, "yellow") == 0){
    rgb[0] = rgb[1] = 255; rgb[2] = 0;
    return 1;
}
if(strcmp(str, "pink") == 0){
    rgb[0] = 255; rgb[1] = 193; rgb[2] = 203;
    return 1;
}
if(strcmp(str, "silver") == 0){
    rgb[0] = 192; rgb[1] = 192; rgb[2] = 192;
    return 1;
}
if(strcmp(str, "orange") == 0){
    rgb[0] = 1255; rgb[1] = 165; rgb[2] = 0;
    return 1;
}
if(strcmp(str, "gold") == 0){
    rgb[0] = 255; rgb[1] = 215; rgb[2] = 0;
    return 1;
}
if(strcmp(str, "purple") == 0){
    rgb[0] = 128; rgb[1] = 0; rgb[2] = 128;
    return 1;
}
if(strcmp(str, "teal") == 0){
    rgb[0] = 0; rgb[1] = 128; rgb[2] = 128;
    return 1;
}
if(strcmp(str, "brown") == 0){
    rgb[0] = 165; rgb[1] = 40; rgb[2] = 40;
```

```
        return 1;
    }
    if(strcmp(str, "magenta") == 0){
        rgb[0] = 255; rgb[1] = 0; rgb[2] = 255;
        return 1;
    }
    if(strcmp(str, "lime") == 0){
        rgb[0] = 50; rgb[1] = 110; rgb[2] = 50;
        return 1;
    }

    return -1;
}

void show_help(){
    fprintf(stderr, "\
            \n\
            -h, --help\t\t\t display help message and exit.\n\
            -t, --triangle\t\t run function to make a triagle.\n\
             -r, --rectangle\t\t run  function that looks for a rectangle with the largest area by the informed color and paint it with the new color.\n\
            -c, --collage\t\t run function that make collage.\n\
            \n\
            Example how to run traingle's function.\n\
            model: -t '(x0 y0) (x1 y1) (x2 y2)' -l 'color' -f 'color' file_in.png file_out.png\n\
             \t line: -t '(100 30) (50, 100) (600 450)' -l 'red' car.png res.png or --triangle '(100 30) (50, 100) (600 450)' -line 'red' car.png res.png\n\
              \t fill: -t '(100 30) (50, 100) (600 450)' -l 'red' -f 'black' car.png res.png or --triangle '(100 30) (50, 100) (600 450)' -line 'red' --fill 'black' car.png res.png\n\
            \n\
            Example how to run traingle's function.\n\
               model: -r 'old_color new_color' file_in.png file_out.png or --rectangle 'old_color new_color' file_in.png file_out.png.\n\
            \t   -r 'red black' car.png res.png  or --rectangle 'red black car.png res.png\n\
            \n\
            Example how to run collage function.\n\
                    model: -c 'linexcolunm' file_in.png file_out.png or --collage 'linexcolunm' file_in.png file_out.png.\n\
            \t   -c '3x5' car.png res.png or --collage '3x5' car.png res.png\n\
            \n");
}

void show_colors(){
    printf("Here are the colors that program works with:\n");
    printf("\tblack\t\twhite\t\tblue\t\tred\t\tgreen\n\n");
    printf("\tyellow\t\torange\t\tpurple\t\tteal\t\tlime\n\n");
    printf("\tbrown\t\tmagenta\t\tpink\t\tsilver\t\tgold\n");
}

void Exec(int argc, char * argv[]){
```

```
   struct option longOps[] = {
      {"triangle", required_argument, NULL, 't'},
      {"line" , required_argument, NULL, 'l'},
      {"fill", required_argument, NULL, 'f'},
      {"rectangle", required_argument, NULL, 'r'},
      {"collage", required_argument, NULL, 'c'},
      {"help", no_argument, NULL, 'h'},
      {0, 0, 0, 0}
   };
   int IndexLongOps;
   char fill = 'n';
   char old[10];
   char new[10];
   int *rgb_line = (int*)calloc(3, sizeof(int));
   int *rgb_fill = (int*)calloc(3, sizeof(int));
   int *rgb_old = (int*)calloc(3, sizeof(int));
   int *rgb_new = (int*)calloc(3, sizeof(int));
   int *coordinates = (int*)calloc(6, sizeof(int));
   int opt, isTriangle = 0, isRectangle = 0, isCollage = 0, n, m;

   while((opt = getopt_long(argc, argv, "t:l:f:r:c:h", longOps, &IndexLongOps)) > 0){
      switch(opt){
         case 't':
                  if((sscanf(optarg, "(%d %d) (%d %d) (%d %d)", &coordinates[0], &coordinates[1],
&coordinates[2], &coordinates[3], &coordinates[4], &coordinates[5])) >= 6){
            isTriangle = 1;
         }else{
            show_help();
            exit(0);
         }
         break;

         case 'l':
            if(setColors(rgb_line, optarg) < 0){
               fprintf(stderr, " Line color not found\n");
               show_colors();
               show_help();
               exit(0);
            }
            break;

         case 'f':
            if(setColors(rgb_fill, optarg) < 0){
               fprintf(stderr, "Color to fill triangle not found\n");
               show_colors();
               show_help();
               exit(0);
            }
            fill = 'y';
```

```
            break;

        case 'r':
            if((sscanf(optarg, "%s %s", old, new)) > 2){
                show_help();
                exit(0);
            }
            if(setColors(rgb_old, old) < 0){
                fprintf(stderr,"input rectangle's color not found\n");
                show_colors();
                show_help();
                exit(0);
            }
            if(setColors(rgb_new, new) < 0 < 0){
                fprintf(stderr,"output rectangle's color not found\n");
                show_colors();
                show_help();
                exit(0);
            }
            isRectangle = 1;
            break;

        case 'c':
            if((sscanf(optarg, "%dx%d", &n, &m)) < 2){
                fprintf(stderr, "To make collage, you need add two values\n");
                show_help();
                exit(0);
            }
            isCollage = 1;
            break;

        default:
            show_help();
            exit(0);
    }
}

if(argv[optind] == NULL){
    fprintf(stderr, "maybe, you did not inform the input file name or/and output file name\n");
    show_help();
    exit(0);
}

int i = optind;
char *file_in = argv[i++];
char *file_out = argv[i];

if(!file_in){
    fprintf(stderr,"check name file it\n");
    show_help();
```

```
        exit(0);
    }
    if(!file_out){
        fprintf(stderr,"check name file out\n");
        show_help();
        exit(0);
    }

    struct Png image;
    int check_file = 1;
    read_png_file(file_in, &image);

    if((check_file > 0) && (isTriangle > 0)){
        triangle(&image, rgb_line, rgb_fill, coordinates[0], coordinates[1], coordinates[2], coordinates[3],
coordinates[4], coordinates[5], fill);
        check_file = -1;
    }
    if((check_file > 0) && (isRectangle > 0)){
        rectangle(&image, rgb_old, rgb_new);
        check_file = -1;
    }
    if((check_file > 0) && (isCollage > 0)){
            collage(&image, n, m);
           check_file = -1;
    }

    printf("well done (*_*), check the file '%s'\n", file_out);
    write_png_file(file_out, &image);
    free(rgb_fill);
    free(rgb_line);
    free(coordinates);
}
```