Course work v5.16

Course work for option 5.16, created by Matvei Kolesnichenko.

Chapter 1

Main Page

1.1 Description

This project is the coursework for option 5.16, created by Matvei Kolesnichenko. It includes several components for processing PNG files, including functionalities like copying, color replacement, ornamentation, and drawing borders around filled rectangles.

1.2 Project Structure

The project is structured as follows:

- `src`: Contains source code files.
- `include`: Contains header files.
- `docs`: Contains documentation generated by Doxygen.
- `Makefile`: Makefile for building the project.
- `README.md`: This file.

1.3 Getting Started

To build the project, navigate to the project directory and run `make`. This will compile the source code and generate the executable.

```
make
```

To clean the project directory and remove generated files, run make clean.

```
make clean
```

1.4 About docs and Doxygen

To generate documentation, run make docs. This will use Doxygen to generate HTML documentation in the docs directory and a PDF file with documentation in the same directory. You can open the HTML documentation by navigating to docs/html and opening index.html.

```
make docs
```

Please note that this command will download the necessary files from Github.

When generating documentation with Doxygen, make sure you have Graphviz installed for generating graphs and LaTeX installed for generating PDF documentation.

1.5 Usage

After building the project, you can execute the generated executable to perform various operations on PNG files. Use command-line arguments to specify the desired functionality and input/output files.

```
./cw [options]
```

1.6 Dependencies

The project depends on the following libraries:

- libpng
- math library (libm)

Ensure these libraries are installed on your system before building the project.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 Area Struct Reference

Structure representing an area copied during the execution of the 'copy_area' function.

`#include <structures.h>`

Public Attributes

- int width
- int height
- png_bytep * row_pointers

4.1.1 Detailed Description

Structure representing an area copied during the execution of the 'copy_area' function.

4.1.2 Member Data Documentation

4.1.2.1 height

`int Area::height`

Height of the area in pixels

4.1.2.2 row_pointers

`png_bytep* Area::row_pointers`

Pointer to an array of pointers, each pointing to a row of area data

4.1.2.3 width

```
int Area::width
```

Width of the area in pixels

The documentation for this struct was generated from the following file:

- include/structures.h

4.2 Options Struct Reference

Structure representing options provided to the program.

```
#include <structures.h>
```

Public Attributes

- char * input_file
- char * output_file
- int flag_help
- int flag_input
- int flag_output
- int flag_copy
- int flag_color_replace
- int flag_ornament
- int flag_filled_rects
- int flag_left_up
- int flag_right_down
- int flag_dest_left_up
- int flag_old_color
- int flag_new_color
- int flag_pattern
- int flag_color
- int flag_thickness
- int flag_count
- int flag_border_color
- int flag_info
- char * left_up_value
- char * right_down_value
- char * dest_left_up_value
- char * old_color_value
- char * new_color_value
- char * pattern_value
- char * color_value
- char * thickness_value
- char * count_value
- char * border_color_value

4.2.1 Detailed Description

Structure representing options provided to the program.

4.2.2 Member Data Documentation

4.2.2.1 border_color_value

```
char* Options::border_color_value
```

Value of the border color for filled rectangles

4.2.2.2 color_value

```
char* Options::color_value
```

Value of the color for ornamentation or filled rectangles

4.2.2.3 count_value

```
char* Options::count_value
```

Value of the count for ornamentation

4.2.2.4 dest_left_up_value

```
char* Options::dest_left_up_value
```

Value of the top-left coordinate of the destination area

4.2.2.5 flag_border_color

```
int Options::flag_border_color
```

Flag indicating if the border color for filled rectangles has been specified

4.2.2.6 flag_color

```
int Options::flag_color
```

Flag indicating if the color for ornamentation or filled rectangles has been specified

4.2.2.7 flag_color_replace

```
int Options::flag_color_replace
```

Flag indicating if the 'color_replace' function should be executed

4.2.2.8 flag_copy

```
int Options::flag_copy
```

Flag indicating if the 'copy' function should be executed

4.2.2.9 flag_count

```
int Options::flag_count
```

Flag indicating if the count for ornamentation has been specified

4.2.2.10 flag_dest_left_up

```
int Options::flag_dest_left_up
```

Flag indicating if the top-left coordinate of the destination area has been specified

4.2.2.11 flag_filled_rects

```
int Options::flag_filled_rects
```

Flag indicating if the 'filled_rects' function should be executed

4.2.2.12 flag_help

```
int Options::flag_help
```

Flag indicating if the help message should be displayed

4.2.2.13 flag_info

```
int Options::flag_info
```

Flag indicating if detailed information about the input PNG file should be printed

4.2.2.14 flag_input

```
int Options::flag_input
```

Flag indicating if the input file has been specified

4.2.2.15 flag_left_up

```
int Options::flag_left_up
```

Flag indicating if the top-left coordinate of the source area has been specified

4.2.2.16 flag_new_color

```
int Options::flag_new_color
```

Flag indicating if the new color for color replacement has been specified

4.2.2.17 flag_old_color

```
int Options::flag_old_color
```

Flag indicating if the old color for color replacement has been specified

4.2.2.18 flag_ornament

```
int Options::flag_ornament
```

Flag indicating if the 'ornament' function should be executed

4.2.2.19 flag_output

```
int Options::flag_output
```

Flag indicating if the output file has been specified

4.2.2.20 flag_pattern

```
int Options::flag_pattern
```

Flag indicating if the pattern for ornamentation has been specified

4.2.2.21 flag_right_down

```
int Options::flag_right_down
```

Flag indicating if the bottom-right coordinate of the source area has been specified

4.2.2.22 flag_thickness

```
int Options::flag_thickness
```

Flag indicating if the thickness for ornamentation or filled rectangles has been specified

4.2.2.23 input_file

```
char* Options::input_file
```

Filename of the input PNG file

4.2.2.24 left_up_value

```
char* Options::left_up_value
```

Value of the top-left coordinate of the source area

4.2.2.25 new_color_value

```
char* Options::new_color_value
```

Value of the new color for color replacement

4.2.2.26 old_color_value

```
char* Options::old_color_value
```

Value of the old color for color replacement

4.2.2.27 output_file

```
char* Options::output_file
```

Filename of the output PNG file

4.2.2.28 pattern_value

```
char* Options::pattern_value
```

Value of the pattern for ornamentation

4.2.2.29 right_down_value

```
char* Options::right_down_value
```

Value of the bottom-right coordinate of the source area

4.2.2.30 thickness_value

```
char* Options::thickness_value
```

Value of the thickness for ornamentation or filled rectangles

The documentation for this struct was generated from the following file:

- include/structures.h

4.3 Png Struct Reference

Structure representing a PNG image.

```
#include <structures.h>
```

Public Attributes

- int width
- int 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

4.3.1 Detailed Description

Structure representing a PNG image.

4.3.2 Member Data Documentation

4.3.2.1 bit_depth

```
png_byte Png::bit_depth
```

Bit depth of the image

4.3.2.2 color_type

```
png_byte Png::color_type
```

Color type of the image (e.g., RGB, Grayscale)

4.3.2.3 height

```
int Png::height
```

Height of the image in pixels

4.3.2.4 info_ptr

```
png_infop Png::info_ptr
```

Pointer to the libpng structure for storing PNG information

4.3.2.5 number_of_passes

```
int Png::number_of_passes
```

Number of passes required for interlacing (typically used for progressive rendering)

4.3.2.6 png_ptr

```
png_structp Png::png_ptr
```

Pointer to the libpng structure for reading/writing PNG data

4.3.2.7 row_pointers

```
png_bytep* Png::row_pointers
```

Pointer to an array of pointers, each pointing to a row of image data

4.3.2.8 width

```
int Png::width
```

Width of the image in pixels

The documentation for this struct was generated from the following file:

- include/structures.h

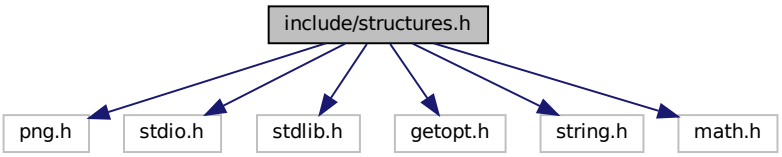
Chapter 5

File Documentation

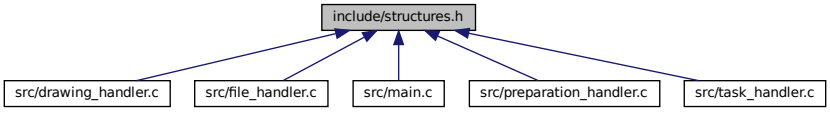
5.1 include/structures.h File Reference

```
#include <png.h>
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include <string.h>
#include <math.h>
```

Include dependency graph for structures.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct Png

  Structure representing a PNG image.
- struct Area

  Structure representing an area copied during the execution of the 'copy_area' function.
- struct Options

  Structure representing options provided to the program.

Typedefs

- typedef struct Png Png

  Structure representing a PNG image.
- typedef struct Area Area

  Structure representing an area copied during the execution of the 'copy_area' function.

5.1.1 Typedef Documentation

5.1.1.1 Area

```
typedef struct Area Area
```

Structure representing an area copied during the execution of the 'copy_area' function.

5.1.1.2 Png

```
typedef struct Png Png
```

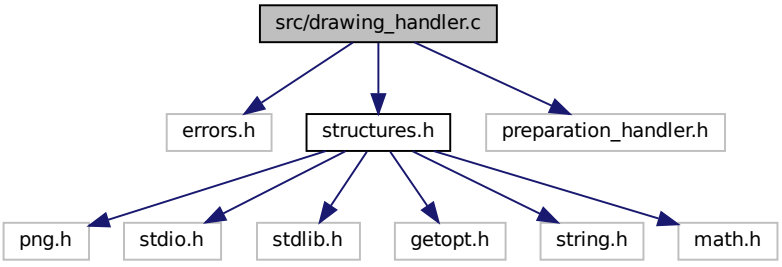
Structure representing a PNG image.

5.2 README.md File Reference

5.3 src/drawing_handler.c File Reference

```
#include "errors.h"
#include "structures.h"
#include "preparation_handler.h"
```

Include dependency graph for drawing_handler.c:



Functions

- void draw_pixel (png_bytep ptr, int ∗color_values)

  Draws a single pixel with the specified color values.
- void draw_border (Png ∗image, int x1, int y1, int x2, int y2, int ∗border_color, char ∗thickness)

  Draws a border around the specified rectangle in the image.
- void rectangle_ornament (Png ∗image, int ornament_thickness, int ornament_count, int ∗color_values, char ∗thickness)

  Draws rectangle ornaments on the image.
- void circle_ornament (Png ∗image, int ∗color_values)

  Draws a circle ornament on the image.
- void semicircles_ornament (Png ∗image, int ornament_thickness, int ornament_count, int ∗color_values)

  Draws semicircle ornaments on the image.

5.3.1 Function Documentation

5.3.1.1 circle_ornament()

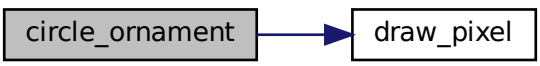
```
void circle_ornament (
            Png * image,
            int * color_values )
```

Draws a circle ornament on the image.

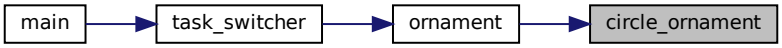
Parameters

image	Pointer to the Png structure representing the image.
color_values	Array containing the RGB values of the ornament color.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.1.2 draw_border()

```
void draw_border (
            Png * image,
            int x1,
            int y1,
            int x2,
            int y2,
            int * border_color,
            char * thickness )
```

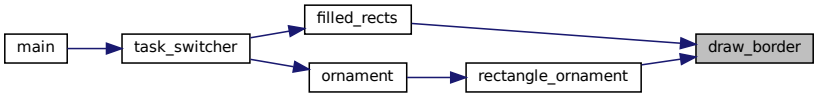
Draws a border around the specified rectangle in the image.

Parameters

image	Pointer to the Png structure representing the image.
x1	The x-coordinate of the top-left corner of the rectangle.
y1	The y-coordinate of the top-left corner of the rectangle.
x2	The x-coordinate of the bottom-right corner of the rectangle.
y2	The y-coordinate of the bottom-right corner of the rectangle.
border_color	Array containing the RGB values of the border color.
thickness	String representing the thickness of the border.

Here is the call graph for this function:

Here is the caller graph for this function:



5.3.1.3 draw_pixel()

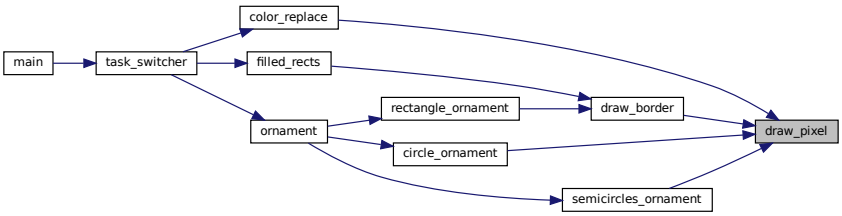
```
void draw_pixel (
            png_bytep ptr,
            int * color_values )
```

Draws a single pixel with the specified color values.

Parameters

ptr	Pointer to the pixel in the image.
color_values	Array containing the RGB values of the pixel color.

Here is the caller graph for this function:



5.3.1.4 rectangle_ornament()

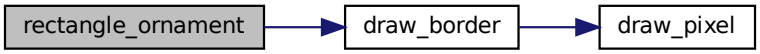
```
void rectangle_ornament (
            Png * image,
            int ornament_thickness,
            int ornament_count,
            int * color_values,
            char * thickness )
```

Draws rectangle ornaments on the image.

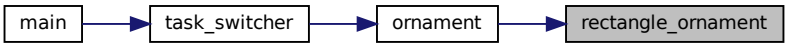
Parameters

image	Pointer to the Png structure representing the image.
ornament_thickness	Thickness of the ornament rectangles.
ornament_count	Number of ornament rectangles to draw.
color_values	Array containing the RGB values of the ornament color.
thickness	String representing the thickness of the border.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.1.5 semicircles_ornament()

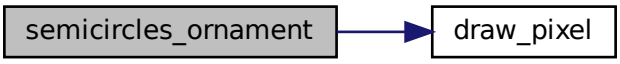
```
void semicircles_ornament (
            Png * image,
            int ornament_thickness,
            int ornament_count,
            int * color_values )
```

Draws semicircle ornaments on the image.

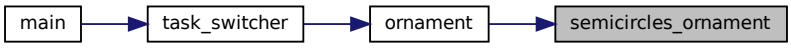
Parameters

image	Pointer to the Png structure representing the image.
ornament_thickness	Thickness of the semicircle ornaments.
ornament_count	Number of semicircle ornaments to draw.
color_values	Array containing the RGB values of the ornament color.

Here is the call graph for this function:



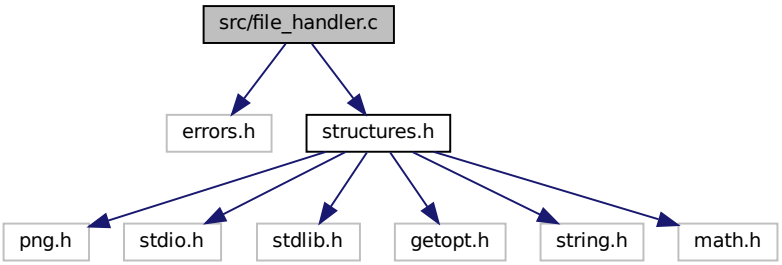
Here is the caller graph for this function:



5.4 src/file_handler.c File Reference

```
#include "errors.h"
#include "structures.h"
```

Include dependency graph for file_handler.c:



Functions

- void read_png_file (char *file_name, Png *image)

  Reads a PNG file and stores its information and pixel data in a Png structure.
- void write_png_file (char *file_name, Png *image)

  Writes a PNG image to a file.

5.4.1 Function Documentation

5.4.1.1 read_png_file()

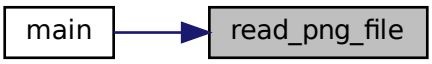
```
void read_png_file (
            char * file_name,
            Png * image )
```

Reads a PNG file and stores its information and pixel data in a Png structure.

Parameters

file_name	A string representing the file name/path of the PNG image to be read.
image	A pointer to the Png structure where the image data and information will be stored.

Here is the caller graph for this function:



5.4.1.2 write_png_file()

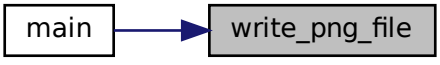
```
void write_png_file (
            char * file_name,
            Png * image )
```

Writes a PNG image to a file.

Parameters

file_name	A string representing the file name/path where the PNG image will be saved.
image	A pointer to the Png structure containing information about the PNG image.

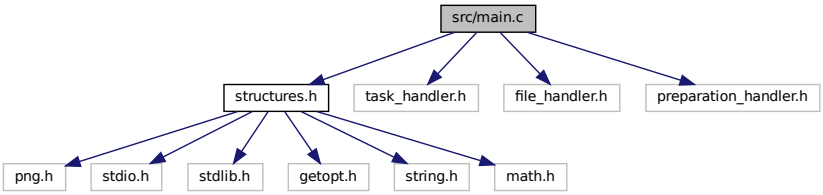
Here is the caller graph for this function:



5.5 src/main.c File Reference

```
#include "structures.h"
#include "task_handler.h"
#include "file_handler.h"
#include "preparation_handler.h"
```

Include dependency graph for main.c:



Functions

- int main (int argc, char ∗argv[ ])

  Main function to handle command-line arguments and process image tasks.

5.5.1 Function Documentation

5.5.1.1 main()

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

Main function to handle command-line arguments and process image tasks.

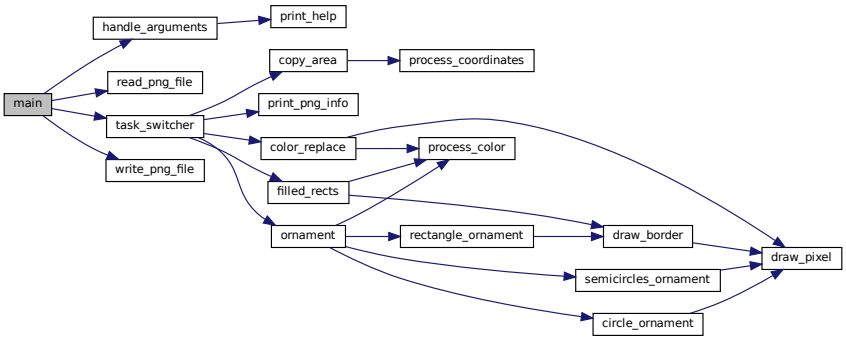
Parameters

argc	The number of command-line arguments.
argv	An array of strings containing the command-line arguments.

Returns

int An integer representing the exit status of the program.

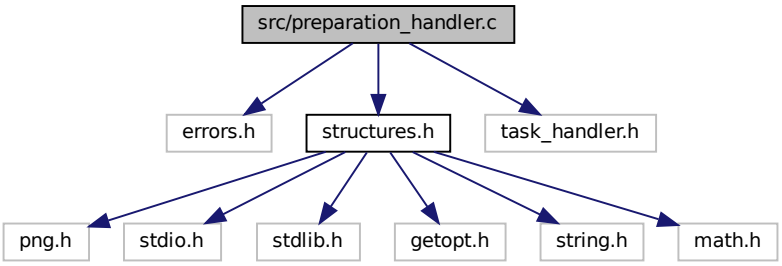
Here is the call graph for this function:



5.6 src/preparation_handler.c File Reference

```
#include "errors.h"
#include "structures.h"
#include "task_handler.h"
```

Include dependency graph for preparation_handler.c:



Functions

- void handle_arguments (int argc, char ∗argv[ ], Options ∗options)

  Handles command-line arguments passed to the program and populates the Options structure accordingly.
- int ∗ process_color (char ∗string_color)

  Processes color provided as a string and returns it as an integer array.
- int ∗ process_coordinates (char ∗string_coordinates)

  Processes coordinates provided as a string and returns them as an integer array.

5.6.1 Function Documentation

5.6.1.1 handle_arguments()

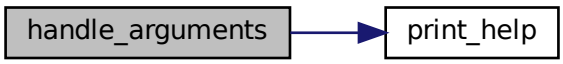
```
void handle_arguments (
            int argc,
            char * argv[],
            Options * options )
```

Handles command-line arguments passed to the program and populates the Options structure accordingly.

Parameters

argc	An integer representing the number of command-line arguments.
argv	An array of strings containing the command-line arguments.
options	A pointer to the Options structure where the parsed arguments will be stored.

Here is the call graph for this function:



Here is the caller graph for this function:

5.6.1.2 process_color()

```
int* process_color (
            char * string_color )
```

Processes color provided as a string and returns it as an integer array.

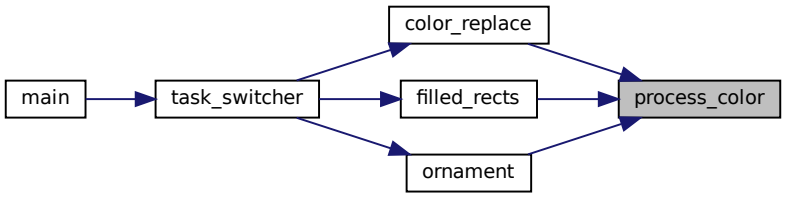
Parameters

string_color	A string representing color in the format "R.G.B".

Returns

int* An integer array containing the red, green, and blue components of the color. NULL if the input string is invalid or if memory allocation fails.

Here is the caller graph for this function:



5.6.1.3 process_coordinates()

```
int* process_coordinates (
            char * string_coordinates )
```

Processes coordinates provided as a string and returns them as an integer array.

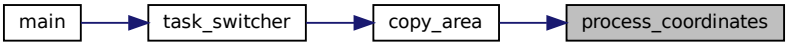
Parameters

string_coordinates	A string representing coordinates in the format "X.Y".

Returns

int* An integer array containing the X and Y coordinates. NULL if the input string is invalid or if memory allocation fails.

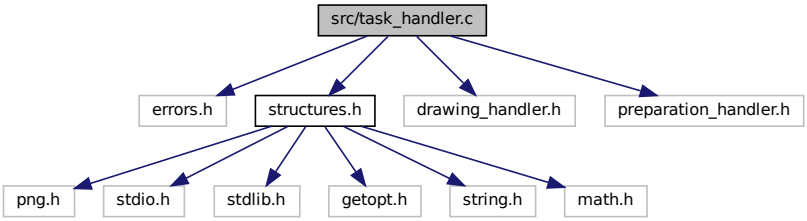
Here is the caller graph for this function:



5.7 src/task_handler.c File Reference

```
#include "errors.h"
#include "structures.h"
#include "drawing_handler.h"
#include "preparation_handler.h"
```

Include dependency graph for task_handler.c:



Functions

- void print_help ()

  Prints the help message explaining the usage of the program and its options.
- void print_png_info (Png *image)

  Prints information about a PNG image.
- void color_replace (Png *image, char *old_color, char *new_color)

  Replaces all pixels of the specified old color with the new color.
- void copy_area (Png *image, char *left_up, char *right_down, char *dest_left_up)

  Copies the specified area from the original image to a new structure, then copies it back to the original image at a different location.
- void filled_rects (Png *image, char *string_color, char *string_border_color, char *thickness)

  Finds all filled rectangles in the image and draws borders around them.
- void ornament (Png *image, char *pattern, char *string_color, char *thickness, char *count)

  Draws an ornament pattern on the given image.
- void task_switcher (Options options, Png *image)

  Handles task switching based on provided options.

5.7.1 Function Documentation

5.7.1.1 color_replace()

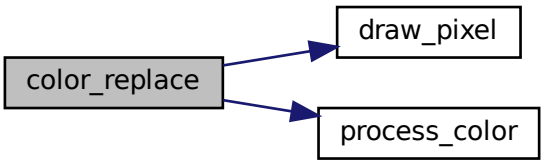
```
void color_replace (
            Png * image,
            char * old_color,
            char * new_color )
```

Replaces all pixels of the specified old color with the new color.

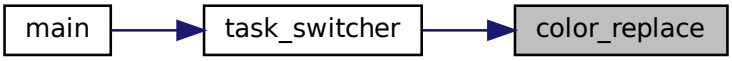
Parameters

image	A pointer to the Png structure representing the image.
old_color	A string representing the old color in the format "R,G,B".
new_color	A string representing the new color in the format "R,G,B".

This function does not return a value. Here is the call graph for this function:



Here is the caller graph for this function:



5.7.1.2 copy_area()

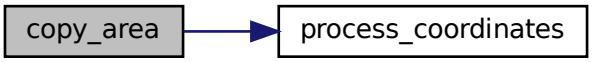
```
void copy_area (
            Png * image,
            char * left_up,
            char * right_down,
            char * dest_left_up )
```

Copies the specified area from the original image to a new structure, then copies it back to the original image at a different location.

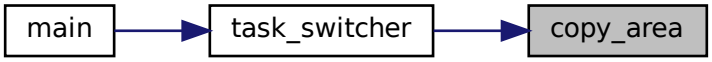
Parameters

image	A pointer to the Png structure representing the original image.
left_up	A string containing the coordinates of the top-left corner of the area to be copied in the format "x,y".
right_down	A string containing the coordinates of the bottom-right corner of the area to be copied in the format "x,y".
dest_left_up	A string containing the coordinates of the top-left corner of the destination location in the original image for the copied area.

This function does not return a value. Here is the call graph for this function:



Here is the caller graph for this function:



5.7.1.3 filled_rects()

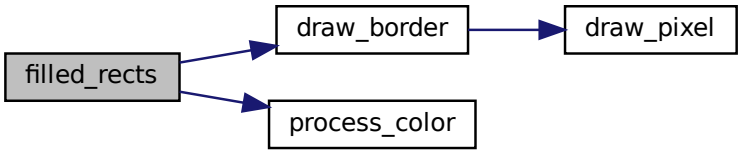
```
void filled_rects (
            Png * image,
            char * string_color,
            char * string_border_color,
            char * thickness )
```

Finds all filled rectangles in the image and draws borders around them.

Parameters

image	A pointer to the Png structure representing the image.
string_color	A string representing the color of the filled rectangles in the format "rrr.ggg.bbb".
string_border_color	A string representing the color of the border in the format "rrr.ggg.bbb".
thickness	A string representing the thickness of the border.

This function does not return a value. Here is the call graph for this function:



Here is the caller graph for this function:

5.7.1.4 ornament()

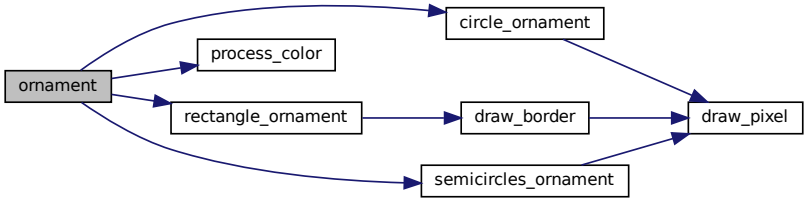
```
void ornament (
            Png * image,
            char * pattern,
            char * string_color,
            char * thickness,
            char * count )
```

Draws an ornament pattern on the given image.

Parameters

image	A pointer to the Png structure representing the image.
pattern	A string specifying the type of ornament pattern ("rectangle", "circle", "semicircles").
string_color	A string representing the color of the ornament in the format "rrr.ggg.bbb".
thickness	A string representing the thickness of the ornament.
count	A string representing the number of ornaments to be drawn.

This function does not return a value. Here is the call graph for this function:



Here is the caller graph for this function:

5.7.1.5 print_help()

```
void print_help ( )
```

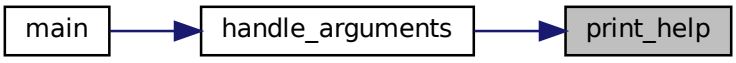
Prints the help message explaining the usage of the program and its options.

This function does not return a value.

The help message includes:

- Information about the course work and its creator.
- Usage syntax.
- Description of available options, including short and long forms, along with their corresponding explanations.

Here is the caller graph for this function:



5.7.1.6 print_png_info()

```
void print_png_info (
            Png * image )
```

Prints information about a PNG image.

Parameters

image	A pointer to the Png structure containing information about the PNG image.

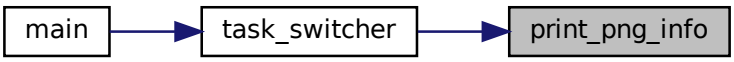
This function does not return a value.

Note

This function prints various details about the PNG image, including its width, height, color type, bit depth, and number of passes.

- The color type is printed as a string representation.
- Bit depth indicates the number of bits per sample or per channel in the image.
- Number of passes refers to the number of passes required for interlaced PNG images.

Here is the caller graph for this function:



5.7.1.7 task_switcher()

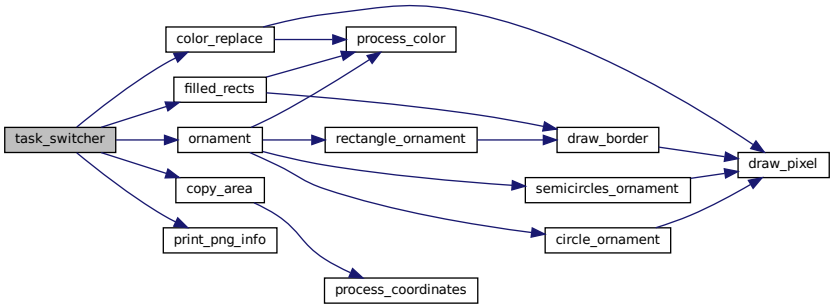
```
void task_switcher (
            Options options,
            Png * image )
```

Handles task switching based on provided options.

Parameters

options	Options structure containing flags and values for various tasks.
image	Pointer to the Image structure representing the image.

This function does not return a value. Here is the call graph for this function:



Here is the caller graph for this function:

Index