Problem Set #4: BMP Transform!

Goals

- Learn to work with structured data types.
- Learn to use standard streams in your program.
- Deepen your knowledge of dynamic memory allocation.
- Learn to work with binaries.
- Directional arithmetic
- Get to know the BMP graphic format and its structure.
- Understand little and big endian.
- Learn to work with Base64 encoded data .

Warning

This assignment must be submitted by **April 24. 2022 23:59:59** . Use the slack channel for discussion `#bmp`.

Warning

This year the tests will not check the use of the command line tool. Only functions from module `bmp`a will be tested `transformations`.

BMP graphic format

BMP (BitMap) is a simple lossless image storage format. It has been with us for over 30 years (it was created around 1987) and nowadays it is of course superseded and replaced by other formats. During its existence, it has seen 5 versions and allows storing images in both compressed and uncompressed form. It was very popular and used mainly in the Windows OS, as it was created as a product of IBM and Microsoft. This is a format that was very often used, but at the same time very unpopular among users, because it was able to compress some images so much that they were larger after compression than before.

An image file in this format consists of:

- from the **header** , which contains metadata, such as file size, dimensions
- and from **image data** , i.e. pixels.

Your task this time will be to create a program that will be able to work with the BMP graphic format - it will be able to load and save files of this type and perform several transformations with them. Specifically, you create

- module `BMP`in which you load and save a file in BMP format,
- a module `transformations`in which you implement individual transformations with a file, and
- file `main.c`, which, based on command line parameters, will perform appropriate operations with the file.

You can find detailed information about the format and its structure, e.g. on [Wikipedia](#) or in the series on the pages of the [root.cz](#) server . In this assignment, you will work with 24-bit color depth. This means that 3B are used to express the color of each pixel. You can find the file header defined in the header file `bmp.h`.

Base64

The BMP graphics format is a binary format. Since we want to represent the binary data in an understandable way in the Arena results, it will be encoded using the [Base64](#) format . You can create an original file from the data encoded in this way at any time using the tool `base64`. For example if he is `stream`represented in the Arena as follows:

```
FILE* stream =
"Qk1WAAAAAAAAADYAAAAoAAAAAgAAAAQAAAABABgAAAAAACAAAAAjLgAAIy4AAAAAAAAAAAAAA////AAAAAAD/AAAA/wAAAA
 // base64 encoded stream
```

You can create a reverse file from it with the following command:

```
$ echo
"Qk1WAAAAAAAAADYAAAAoAAAAAgAAAAQAAAABABgAAAAAACAAAAAjLgAAIy4AAAAAAAAAAAAAA////AAAAAAD/AAAA/wAAAA
 | base64 -d > file.bmp
```

Subsequently, you can work with the file on which the tests took place.

BMP mode

This module contains basic functions for loading/saving the BMP graphic format from/to a file, as well as the structured types with which the BMP graphic format will be represented.

Structure`bmp_header`

This structure represents the header of the BMP graphics format. It occupies 54 flats in memory. The meaning of the individual structure items as well as their permissible values can be found in the header file `bmp.h`.

> Warning
>
> However, not all header items are used in practice. Since in the assignment we will only work with BMP images that use 24-bit depth, we will only be interested in items that depend on the content of the image (they are not constant). These are `.size`, `.width`and `.height`. However, the expected constants will of course also be checked in the tests!

Structure`pixel`

The structure contains three items representing one pixel as a combination of red, green and blue colors. Each of them has a size of 8 bits, which means that one pixel will be represented by 24 bits.

A structured data type`struct bmp_image`

This structure represents the BMP image as a whole, which consists of

- **headers** (item `.header`), az
- **sequences of pixels** (item `.data`).

Data, and thus a sequence of pixels, is represented as a one-dimensional array of pixels that are linearly stored in memory one after the other. This means that the number of elements of this array is given as the product of the height and the width of the image.

Task #1: Load a BMP image from an input stream

To load a BMP file either from a file or standard input, it will be necessary to use a composition of several functions:

1. A function `read_bmp_header()`that reads the header of a BMP file. The header is located at the beginning of the stream. `type`When loading, the value must be in the item `BM`.
2. A function `read_data()`that reads a sequence of pixels representing the contents of a file.
3. A function `read_bmp()`that returns a reference to the loaded complete image consisting of a header and data.

In case the stream is not open (it will have the value `NULL`) or the reference to the header is not valid (it will have the value `NULL`), the function will also return the value `NULL`.

In case the function `read_bmp()`does not load the header, in addition to returning a value, it `NULL`prints a message on the standard error output:

```
Error: This is not a BMP file.
```

If the function reads the header correctly, but not the data, it returns a value `NULL`and writes a message to the standard error output:

```
Error: Corrupted BMP file.
```

Task #2: Write a BMP file to the output stream

Create a function to write the BMP graphic format to the file `write_bmp()`. If the write is successful, it returns the value `true`, otherwise it returns the value `false`.

> Note
>
> You can check whether you have successfully saved the image to the file at any time by opening it in the image browser. However, to test specific values in the header, use one of the **hex editors** . The editor from Midnight Commander , which handles hex mode, is also sufficient for this task .

Task #3: Freeing memory

Create a function `free_bmp_image()`that frees the occupied memory reserved for keeping the BMP image in memory.

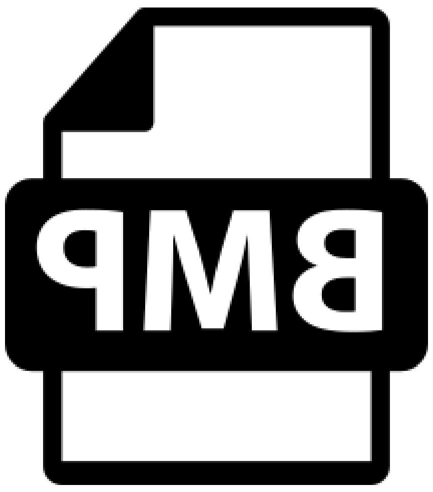
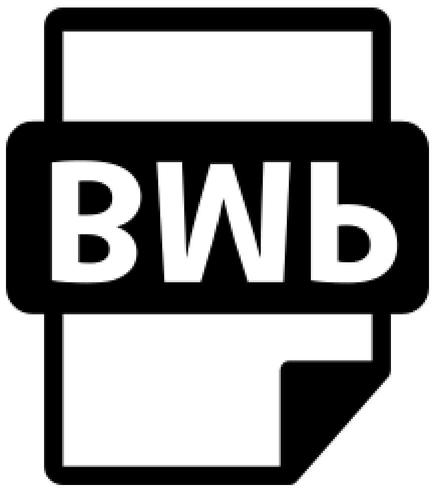
Modul *Transformations*

Basic image editing will be covered in this module. The goal is to create a new one according to the sample image defined at the input, which will be created by applying adjustments.

Task #4: Flipping the image horizontally and vertically

Your task is to create a pair of functions `flip_horizontally()`and `flip_vertically()`, which will flip the image horizontally or vertically. This operation creates a mirror image of the original image along the flip axis. Image dimensions remain unchanged after flipping.

The following table illustrates how the flipped original image will look when calling each function.

original image	`flip_horizontally()`	`flip_vertically()`
		

The parameter of both functions is a reference to the image:

- `const struct bmp_image* image`- A reference to a structure that represents a sample image.

The functions create a new image that is created by flipping the original one. If the function receives the value , instead of a reference to the image `NULL`, the function `NULL`also returns the value.

The outputs of the functions are illustrated in the following examples:

```
flip_horizontally()

source image               new  image
+--------+--------+        +--------+--------+
| FF0000 | FFFFFF |   ->   | FFFFFF | FF0000 |
+--------+--------+        +--------+--------+
```

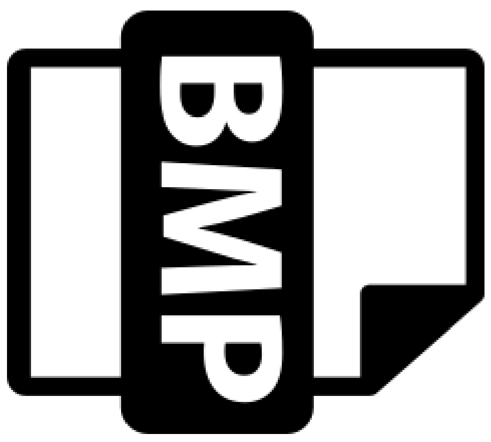
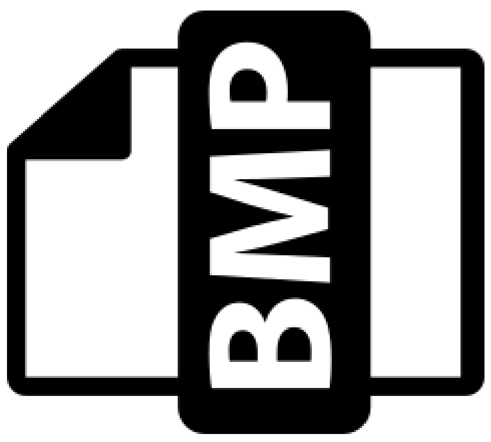
```
flip_vertically()

source image             new image
+--------+               +--------+
| 000000 |               | FFFFFF |
+--------+      ->       +--------+
| FFFFFF |               | 000000 |
+--------+               +--------+
```

Task #5: Rotate the image 90° left and right

Your task is to create a pair of functions `rotate_right()`and `rotate_left()`, which will create a copy of the original image rotated 90° to the right or left. The dimensions of the image will be changed in this case.

The following table illustrates how the rotation of the original image will look when calling each function.

original image	`rotate_right()`	`rotate_left()`
		

The parameter of both functions is a reference to the image:

- `const struct bmp_image* image`- A reference to a structure that represents a sample image.

The functions create a new image that is created by flipping the original one. If the function receives the value , instead of a reference to the image `NULL`, the function `NULL`also returns the value.

Task #6: Cut out part of the image

Your task is to create a function `crop()`that cuts out part of the original image. This part is returned by the function as a new image.

The function has several parameters:

- `image`- reference to the image from which we want to cut the part
- `start_y`- y-position of the top left point to cut
- `start_x`- x-position of the top left point to cut
- `height`- the height of the cropped image in pixels
- `width`- the width of the cropped image in pixels

The cropped image must be inside the original image and its size can be min `1x1`.

The function returns a new image, which is created by cutting the image with the width given by the parameter `width`and the height given by the parameter `height`with the starting point at position `[start_y, start_x]`. In case the reference to the original image does not exist or the individual parameters are out of range, the function returns the value `NULL`.

Task #7: Resize the image

Your task is to create a function `scale()`that proportionally reduces or increases the size of the image based on the scale.

The function has the following parameters:

- `image`- reference to the image to be reduced/enlarged
- `factor`- zoom scale, which can be:
    - `factor < 1`- the image will be reduced
    - `factor = 1`- the image will remain unchanged
    - `factor > 1`- the image will be enlarged

You calculate the new width and height of the image based on the entered value of the variable `factor`as follows:

$$nandininidth = inidth \cdot factOr$$

$$nandinhandight = handight \cdot factOr$$

Round the result using the function `round()`.

To determine the color of a pixel in the new image, a pixel from the original image that is in a similar position is selected. The following relationship applies between the positions of the pixels in the original and the new image (example for position on the x-axis):

$$\frac{nandinx}{nandininidth} = \frac{x}{inidth}$$

So let's take an example: let the original width of the image be `3`and the factor has the value `2.3`. We calculate the new width as follows:

$$nandininidth = inidth \cdot factOr = 3 \cdot 2.3 = 6.9$$

After rounding, the value of the new width will be `7`.

The relationship between the pixel in the new image and the old image is shown in the following table and the following figure.

newx	**intermediate result**	**x**
`0`	`0`	`0`
`1`	`0.43`	`0`
`2`	`0.86`	`0`
`3`	`1.29`	`1`
`4`	`1.71`	`1`
`5`	`2.14`	`2`
`6`	`2.57`	`2`

An example of scaling
An example of scaling

The function returns a new image created by enlarging/reducing the original image based on the given scale. In case the reference to the original image does not exist or the individual parameters are out of range, the function returns the value `NULL`.

Task #8: Choosing a color component

Your task is to create a function `extract()`that will create a copy of the original image and it will contain only the selected color components of the original pixels. This means that if the extraction concerns the red color, the resulting image will not contain green and blue components in the pixels.

```
// extract red and blue colors
// pixel in the form of RRGGBB

source image               extracted image
+--------+--------+        +--------+--------+
| 112233 | FFFFFF |   ->   | 110033 | FF00FF |
+--------+--------+        +--------+--------+
```

The function has the following parameters:

- image- reference to the source image
- colors_to_keep- a string that can only be composed of characters r, gand bdepending on which color you want to extract from the image; the order of the characters in the string does not matter!

The function returns a new image, which is created by extracting the selected color components from the original image. If the reference to the original image or the color component selection string does not exist, or if the string contains characters other than those allowed, the function returns the value NULL.

Command line parameters

The output of your solution will be an executable program named bmp, which will perform transformations on BMP files. Individual operations will be performed based on the switches used.

The following listing shows an example of use when the image is rotated to the right saboteur.bmpand the output is saved to a file output.bmp:

```
$ bmp -i saboteur.bmp -o output.bmp -r
```

Note

You can parse the command line parameters manually or you can use a dedicated standard library, such as getopt.h.

Using the command

```
Usage: bmp [OPTION]... [FILE]...
Simple BMP transformation tool.

With no FILE, read from standard input or write to standard output.

  -r             rotate image right
  -l             rotate image left
  -h             flip image horizontally
  -v             flip image vertically
  -c y,x,h,w     crop image from position [y,x] of giwen height and widht
  -s factor      scale image by factor
  -e string      extract colors
  -o file        write output to file
  -i file        read input from the file
```

Concatenation

Specifying the input and output files is optional. If no input file name is specified, the standard input channel (`stdin`) will be used. If no output file name is specified, the standard output channel (`stdout`) will be used. *This will make it possible to chain the output of one command with the input of another command* using pipes .

This functionality is illustrated in the following example, where the input is an image with the name `saboteur.bmp`, it is rotated to the right, then rotated back to the left and saved in the file `output.bmp`. Each operation is separated by an oven:

```
$ bmp -i saboteur.bmp | bmp -r | bmp -l | bmp -o output.bmp
```

Warning

details will be continuously updated

Handing over the project

The assignment is submitted via the Git version control system on the git.kpi.fei.tuke.sk server . You will submit the solution to this task as part of your project.

The structure of your project will look like this:

```
.
├── ps4/
│   ├── bmp.c
│   ├── bmp.h
│   ├── main.c
│   ├── Makefile
│   ├── transformations.c
│   └── transformations.h
└── README
```

where the meaning of the files in the folder `ps4/`is as follows:

- `bmp.c`, - *BMP*`bmp.h` module source code and header file .
- `transformations.c`, `transformations.h`- Source code and header file of the *Transformations* module .
- `main.c`- Source code containing the function `main()`.
- `Makefile`- `Makefile`the file will contain at least the following goals:
    - `bmp.o`*to generate the BMP* module ,
    - `transformations.o`for generating the *Transformations* module ,
    - `all`to generate an executable program named `bmp`, a
    - `clean`for removing continuously created object and executable files.

In the file `README`, which is located directly in the folder `prog-2022/`, enter the name of your group that you attend during the exercises (you can find it on the maisu schedule portal) in the form:

```
GROUP : X1
```

If you are a **repeat** student, put in `README`the group in the form:

```
GROUP : O<P>
```

where `<P>`you replace with the letter of the parallel which, according to the schedule, corresponds to your study program (e.g. `A`for computer scientists).

> **Warning**
>
> It is important that your files maintain the structure listed. If any of the files are in the repository, but in a different folder, it will be considered an error and such a project will not be considered correct! If, on the other hand, there are extra files or folders in your project, these will not be considered an error.

> **Warning**
>
> Folder, file and file content names are `README` **case** -sensitive!

Project staff

Download the bmp.zip file from the following link , which contains the project skeleton along with sample BMP images for you to experiment with. This package contains the following files and folders:

- `bmp.h`- header file, which contains the declarations of all required functions for the *BMP* module .
- `transformations.h`- a header file that contains the declarations of all required functions for the *Transformations* module .
- `assets/`- a folder with images in BMP format that you can experiment with

In the *Linux OS environment,* you can use a command `wget`of the form:

```
wget https://kurzy.kpi.fei.tuke.sk/pvjc/2022/download/bmp.zip
```

Evaluation and testing

Your evaluation will depend on the results of the tests that your assignment successfully passes. The following will be verified:

- The structure of your project (whether it contains all the necessary files).
- Static analysis of your code using the `cppcheck`.
- Checking for memory leaks with a tool`valgrind`
- The presence of global variables in your code.
- The functionality of your implementation.

Your code will be compiled by the gcc compiler with the following switches and libraries:

```
$ gcc -std=c11 -Werror -Wall -lm
```

Testing of your solutions will be done automatically every *3* hours.

Your solutions will once again pass the originality check. Therefore, when working on your assignment, behave according to the rules of the code of ethics ! If you submit an assignment that is not yours, you will be expelled from the course!

Programming

The course covers advanced topics in the *C language and Arduino* microcontroller programming .