Technische Universität München

Informatics I for MSE

MSE IN8011
Prof. Bjoern Menze, Giles Tetteh
Philipp Kaeß

WS 19/20
Homework 6

Due Date: 15.12.19, 23:59

Release Date: 06.12.19

Homework 6: Swapping with Pointers

Topics: Pointer, Pointer with `Struct`, Swapping, `Struct` Initialization

Introduction

One very common and apparently trivial problem in Informatics is the task of swapping two values. But as we will notice it isn't always as simple as it may seem.

As we want to define an external function to swap e.g. two `int`s from the main, we will have to make use of pointers. As you already learned in the tutorials, a simple **pass by value** (just as we did it until now) will not be sufficient. When calling the swap-function with two plain `int`-arguments, two local copies of that values will be created and all changes to these local copies won't affect the two original `int`s from the main.

Ancient Wise Saying from Middle Earth, colorized

To solve this, we make use of pointers and **pass the two values by reference**. This enables us to directly work at the respective pieces of memory and swap them. In this homework you will create three functions. In addition to swapping `int`s, we also want to initialize and swap a more complex structure, namely variables of type `struct rectangle`. This type is defined in the header file ***swap.h***.

Specifications

After a run of the function `swap_ints(int *a, int *b)` with the input parameters

- `a` , the pointer to the first int
- `b` , the pointer to the second int

the two integer values at the respective addresses should be swapped.

After a run of the function `initialize_rectangle(struct rectangle *r, char color[20], int length, int width)` with the input parameters

- `r` , the pointer to the rectangle to be initialized
- `color` , the char array containing the color (0-terminated)
- `length` , the length as int
- `width` , the width as int

the `struct rectangle` at the address `r` should be correctly initialized with the values `color`, `length` and `width`.

After a run of the function `swap_rectangle(struct rectangle *r1, struct rectangle *r2)` with the input parameters

- `r1` , the pointer to the first rectangle
- `r2` , the pointer to the second rectangle

the `struct rectangle`s at the two addresses `r1` and `r2` should be swapped.

Tasks

You are provided with the template file ***swap.c*** on Moodle. Complete the functions `swap_ints()`, `initialize_rectangle()` and `swap_rectangle()`, so that the program works as specified above. Please note:

- In this template file we predefined the function `print_rectangle(struct rectangle *r)`. This function may help you to test your progress in the two rectangle functions. It receives a pointer to a `struct rectangle` and prints the stats. Please do only use this function in the process of creating your functions, and **do not call it in the final version** that you submit! Your functions should not print anything!

- You are only allowed to write code in the spots indicated in the functions. Do not change anything else. Also do not include or remove any libraries or headers.

- Do not print anything

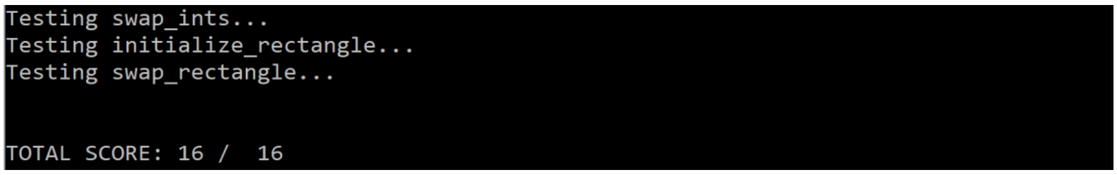
- You are provided with additional files:
  - ***main_swap.c*** is the main file we provide you to test your functions. Do not change anything in this file.
  - ***swap.h*** is the header to the template file ***swap.c***. It contains the definition of `struct rectangle` and the declaration of the functions from ***swap.c***. This header is included in ***swap.c*** as well as in ***main_swap.c***.

- To run your functions, compile the two C-Files together (`gcc -Wall -Werror main_swap.c swap.c`).

- When you are finished, put ***swap.c*** directly in a zip-File named "HW6_Mustermann_Max" (with your name of course) and upload this file to Moodle. Do not upload ***swap.h*** or ***main_swap.c***, and do not put any additional folders into the zip-File.

Example Output

After you compiled and executed your files together, your terminal should look like this:



```
Testing swap_ints...
Testing initialize_rectangle...
Testing swap_rectangle...


TOTAL SCORE: 16 /  16
```

Tool Box

The explanations below **may** help you to solve this problem:

POINTER

A pointer is nothing else than a variable that holds the address of a specific piece of memory. We call it pointer, as we can imagine it as pointing to the variable that is stored in this piece of memory, and quite often we visualize pointers as arrows. As it is important to know of what type the referenced variable is, the pointer itself "contains" this datatype. An `int *` (`int`-pointer) always points to a variable of type `int`, a `float *` points to a variable of type `float`, etc. We can declare a pointer just like any other variable:

```
int *my_ptr = 0;
```

This is now a NULL-pointer, which means it contains no valid address, it "points to NULL". We can assign the address of a variable to our pointer by using `&`:

```
int a = 5;
my_ptr = &a;
```

Here, `&a` returns the address of our variable `a`, which we then directly assign to our pointer. If we want to access the variable where `my_ptr` points to, we can dereference it by using `*`:

```
*my_ptr = 6;
```

Now the value of `a` evaluates to `6`. It is very important to clearly differentiate between the usage of a pointer with and without `*`. If we use it without `*`, we change/read the pointer itself, so for example we change where our pointer points to. If we use it with `*`, we don't change/read the pointer, but the variable it points to (here `a`).

STRUCT POINTERS

When using pointers with structs, it can become a little complicated. Imagine you have a `struct rectangle *r`. How can we access a member of that rectangle? We first have to dereference our pointer, and then access the member (e.g. `length`) with the dot operator. And as the order is of importance, we have to use parenthesis:

```
(*r).length = 10;
```

As this can get complicated (especially when we have pointers as struct members), there is a more convenient equivalent to the above statement which we encourage you to use instead:

```
r->length = 10;
```

Threshold
To pass this homework, your program has to fulfill the following conditions:

- Your file ***swap.c*** must be compilable and runnable together with the main file without any errors and with the flags -Wall -Werror enabled
  (`gcc -Wall -Werror -o test main_swap.c swap.c`)

- Your functions will be tested on a total of 50 cases. To pass this assignment, you have to pass at least 80% (=40) of the test cases.