Mandatory tasks

1. Declare and define a variable outside the main function. Change its value and write it out!
2. Create a function, declare and define a variable in it. Can you access the variable outside the function?
3. Write a branch to the function written in the previous problem after the variable definition. Examine whether you reach the variable in the if condition as well as in the true block, and optionally in the else and else if branches.
4. Declare and define a variable in any branch of an if, or in the core of an arbitrary loop. Examine whether you can access the variable outside the block.
5. Create nested scopes (blocks between "{}" pairs), introduce a variable in each. Examine which variables from which scope you reach! When can you use scopes? What is a stack and how does it connect to scope management?
6. Declare a two-variable function using the same variable name for the two parameters. What's happening?
7. Create a function, declare and define a static variable in it! Increase the value of the variable and write it out. Call the function in the main several times! What's happening?
8. Write a swap function that swaps the values of two int variables.
9. Write a function that expects two int pointers as parameters and returns the pointer pointing to a larger value.
10. The following task can only be solved with a GCC compiler: modify the swap task by declaring and defining the function within the main! When can this functionality be useful?
11. Create a file my_utils.h and my_utils.c! Place the declarations of the functions created in Tasks 8 and 9 in the header file and the definitions in the c file. Include the old header in a main.c file and then call the functions in main! Compile and run your code!
12. Create additional functions in the my_utils.c file! Functions should refer to each other! Can you also call these functions in the main.c file? Why? Why not?

Optional tasks

1. Declare and define a variable at the core of a loop, then embed a second cycle and see if you reach the variable declared in the outer cycle within it!

2. Declare a function in advance and then define it using other parameter names as parameters. What's happening? When can this be useful?

3. Static variable lets you create a factorial a function that counts the number of times it was called with a parameter less than 1!

4. Write a function that expects two ints as parameters and returns a pointer to the result. What happens when you print the result in main (dereferenced)?