Exercises:

1) Learn, how your IDE can display the contents of the general purpose registers in debug mode. Make screenshots of the registers while debugging an application.
2) Show how the stack frames of functions being called, differ from the stack frames of the callee functions in their addresses. To which conclusions can you come concerning the addresses of caller/callee stack frames? Document your findings with screenshots if required.
3) By default, arguments are:
   1. passed by value to functions and
   2. objects are getting returned by value from functions.

   Write a program that proves both statements. Document clearly how this works with the stack.
4) Write a program that proves that uninitialized global variables and uninitialized local static variables are automatically 0-initialized.
5) Create a program containing a function that uses only one local int and it prints this int to the console. Each time this function is called the int should be incremented. I.e. the 10th call of that function should increment the local int to 10 and print a 10 to the console. Document and explain how this works.
6) Try to get one of the presented stack overflow examples working on your compiler system. Be aware that you'll need to modify some compiler settings to get rid of stack protection stuff.
7) What is happening in the following code? How can it be fixed?

```
int main() {
        double data = 42;
        char buffer[3];
        std::sprintf(buffer, "%f", data);
}
```

Remarks:

- Everything that was left unspecified can be solved as you prefer.
- In order to solve the exercises, only use known constructs, esp. the stuff you have learned in the lectures!
- **Please obey these rules for the time being:**
  - **The usage of goto, C++11 extensions, as well as #pragmas is not allowed.**
  - **The usage of global variables is not allowed.**
  - **You mustn't use the STL, esp. *std::string*, because we did not yet understood how it works!**
  - **But *std::cout*, *std::cin* and belonging to manipulators can be used.**
  - **You mustn't use new and delete!**
  - **You are not allowed to use C++ references instead of pointers.**
- Avoid magic numbers and use constants where possible.
- The results of the programming exercises need to be runnable applications! All programs have to be implemented as console programs.
- The programs mustn't have any memory leaks!
- The programs need to be robust, i.e. they should cope with erroneous input from the user.
- You should be able to describe your programs after implementation. Comments are mandatory.

- In documentations as well as in comments, strings or user interfaces make correct use of language (spelling and grammar)!
- Don't send binary files (e.g. the contents of debug/release folders) with your solutions! Do only send source and project files.
- Don't panic: In programming multiple solutions are possible.
- If you have problems use the Visual Studio help (F1) or the Xcode help, books and the internet primarily.
- Of course you can also ask colleagues; but it is of course always better, if you find a solution yourself.