Exercises:

1. Create an interactive program that provides all sorts of conversion between decimal, binary, hexadecimal and octal numbers. – The program should have a menu to let the user select the kind of conversion.
2. Use the memory view in the debugger once again.
   a) What byte order does your system use in memory? – Show this with screenshots if necessary and write some words about your findings.
3. Create a program that shows how an array can be iterated without the []-operator.
4. What do the following statements do? When is it an error, when is it ok?

   ```
   pi = &ival;
   pi = pi + 1024;
   ```
5. Create an interactive program, which expects a network IP4 address of a node in a format like "192.168.1.42" (dotted decimal notation) and a subnetwork mask in a format like "255.255.255.0". The program should print the address of the network and the address of the device as result. In its implementation, the program should mainly use cstring operations and bit-wise operations. Additionally the program should also work with the suffix notation.

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.