Exercises:

1. Install Visual Studio Code and the C++ build tools to create C++ applications. Document the required steps from the installation until the first running C++ program on a Confluence page.
2. Compile, link and run a C++ program on the console using Microsoft's build tools. Please document the required steps accordingly.
3. Find three examples of concepts in the reality that could be expressed as UDTs.
   a) Write three structs, representing these concepts!
   b) Document your ideas and the UDTs!
4. The UDT *Fraction*:
   a) Create the UDT *Fraction*.
   b) Create following API for *Fraction*:
      1. A free function that allows adding two *Fraction*s.
      2. A free function that allows subtracting two *Fraction*s.
      3. A free function that allows multiplying two *Fraction*s.
      4. A free function that allows dividing two *Fraction*s.
      5. A free function that allows outputting a *Fraction* object to the console.
      6. A free function that prompts the user to input numerator and denominator of a passed *Fraction* instance on the console.
   c) Comment the API, so that users can understand and use *Fraction* and its API without inspecting the code.
   d) Create tests for this API! Mind error cases!
   e) Create an application to let the user play with *Fraction* (incl. a menu).
5. An array of *Student* (i.e. the UDT shown in the presentation):
   a) Show how we can overwrite parts of an adjacent item in an array of *Student*. Document clearly, what is going on in the memory. – Use screenshots, if required.

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!**
- For the time being, only use structs for your UDTs.
- Do not put struct definitions into separate files (we have not yet discussed separated compilation of UDTs).
- The results of the programming exercises need to be runnable applications! All programs have to be implemented as console programs.
- 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.