Exercises:

1. Use the __*FILE*__ and the __*LINE*__ macro in an example.
2. Where do you see possible problems in following macros?

```
#define PI = 3.14
#define MAX(a,b) a>b?a:b
#define fac(a) (a)*fac((a)-1)
```

3. Correct the *SQUARE()* macro from the presentation.
4. Optimize the macro call *MAX(++n, ++m)* from the presentation.
5. Write a program that uses the *assert()* macro with a failing condition.
   a) How does the failed assertion show up in debug mode in the IDE and on the console?
   b) How does the failed assertion show up in release mode in the console?
   c) Document your findings!
6. Prove that a class definition is a definition and not a declaration.
7. Learn how to define macros, when starting the compiler on the command line. – Create an example, that shows how it works with conditional compilation.
8. Show a usage of *assert()* and its behavior, when asserts were activated at compile time and when asserts were deactivated at compile time.
9. Learn how you can generate a translation unit (tu) of an implementation file with your compiler system in the IDE and on the command line. Write some sentences to explain how this works.
   a) How does the *assert()* macro appear in the tu?
10. Create a project that has a link time error. Write some sentences to explain how this error could emerge.
11. How does the *assert()* macro appear in an o-file?
12. Where do the C/C++ standard h-files reside for the compiler you use?
13. Write a program that defines the UDTs and hierarchy *Engine*, *Tyre*, *Car*, *VintageCar* and *Bus*. Spread the definitions into multiple files! Use the types *Car*, *VintageCar* and *Bus* in the *main()* function.
    a) Put each of that types into the namespace *MyNamespace*.
    b) Answer this question: Why are C++-namespaces said to be "open"?
14. Spread the types of the Shape-Triangle-Circle-Rectangle-Square-example of a former lecture into multiple files as well.
15. Show an example provoking an unresolved external symbol and another example provoking duplicate/multiple ambiguous external symbols.
    a) Please write some words how either comes to happen.

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, because we did not yet understood how it works!**
  - **But *std::string*, *std::cout*, *std::cin* and belonging to manipulators can be used.**
- Only use classes for your UDTs. The usage of public fields is not allowed! The definition of inline member functions is only allowed, if mandatory!
- Your types should apply const-ness as far as possible. They should be const-

correct. Minimize the usage of non-const&!

- 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.