Exercises:

1. Write a new program. The program should ask the user for an integer number and this integer number will be echoed to the console. If the user enters a negative number, the program will be stopped.
2. Write a new program. The program should print all letters from a-z to stdout, incl. their ASCII code in dec, hex and oct notation. Use a tabular format.
3. Write a program, that shows an iteration-construct only with forced branching, i.e. without any loops!
4. Write a new program, which outputs the numbers in [1, 100], obeying following rules:
   a) If the number is a multiple of 3 output "foo" instead of the number.
   b) If the number is a multiple of 5 output "bar" instead of the number.
   c) If the number is a multiple of 3 and a multiple of 5 output "foobar" instead of the number.
5. Write a new program, which prints following output to the console:

```
Terminal
PC:src hers$      00
PC:src hers$     0000
PC:src hers$    000000
PC:src hers$   00000000
PC:src hers$  0000000000
PC:src hers$ 000000000000
PC:src hers$    000000
PC:src hers$    000000
```

6. Write a new program asking the user for five integers (Don't use arrays!). The program then outputs the sum of squares.
   a) Learn how you can start a C++ program from the Windows/macOS console, i.e. without the IDE (i.e. without Visual Studio or Xcode)! -- Document clearly how it works! – Your knowledge about using the console should help you.
   b) A menu should be implemented with following options: [input numbers], [calculate sum of squares] and [quit]. All options (except [quit]) return to the menu after their action has taken place. Erroneous user inputs (e.g. text instead of an integer) should be checked by the program and require new input from the user.

Remarks:

- If exercises ask to document something, a Word document with explanatory text, maybe incl. snippets and screenshots is awaited as companion artifact in the repository or sent as attachment to the solution of the exercise!
- 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!**
- 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 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.