CP2406 Prac-8

By solving the following exercises, you can practice the material discussed in the relevant chapter. Solutions to all exercises are available in the "solutions"-subfolder. However, if you are stuck on an exercise, first reread parts of the chapter to try to find an answer yourself before looking at the solutions.

NOTE! VSCode is getting confused very often. Tested fixes:
* Close and open it.
* Delete ".vscode" folder(s).
* How to configure and compile VSCode for multiple .cpp files: see prac05

Please note that std-17 solutions may not be provided but you need to submit this prac solution in std-17.

prac08_task1 for Chapter-14-"Errors"

1) Complete the textbook's Exercise 14-1: find and correct the errors in the starter code "prac08_task1_ex14p1/test.cpp".

2) Hint: Review the chapter-14 pdf

3) Modify main() to **trigger** every implemented exception.

4) Check your solution against std-20 "solutions_textbook" folder.

prac08_task2

5) Compete the textbook's Exercise 14-2 (see starter code in "prac08_task2_ex14p2"):

Exercise 14-2: Take the code from the bidirectional I/O example from Chapter 13. You can find this in the `c13_code\19_Bidirectional` folder in the downloadable source code archive. The example implements a `changeNumberForID()` function. Retrofit the code to use exceptions on all places you deem appropriate. Once your code is using exceptions, do you see a possible change you can make to the `changeNumberForID()` function header?

6) Hint: You need to replace returned bool with exceptions. Review the chapter-14 pdf.

7) Modify main() to **trigger** every implemented exception. To trigger some exceptions, you may need to run your code in debugger, while in the debugger mode, modify/delete the text file.

8) Check your solution against std-20 "solutions_textbook" folder.

prac08_task3

9) Complete the textbook's Exercises 14-3 (see starter code in "prac08_task3_ex14p3"):

Exercise 14-3: Add proper error handling using exceptions to your person database solution of Exercise 13-3.

10) Hint: Look for error reporting code and replace it with exceptions. Review the chapter-14 pdf.

11) Modify main() to **trigger** every implemented exception. To trigger some exceptions, you may need to run your code in debugger, while in the debugger mode, modify/delete the database files.

prac08_task4 for Chapter-15-Overloading-C++-ops

12) Convert and test the std-20 solution for the textbook's Exercise 15-1 to std-17.

Exercise 15-1: Implement an `AssociativeArray` class. The class should store a number of elements in a `vector`, where each element consists of a key and a value. The key is always a `string`, while the type of the value can be specified using a template type parameter. Provide overloaded subscripting operators so that elements can be retrieved based on their key. Test your implementation in your `main()` function. Note: this exercise is just to practice implementing subscripting operators using non-integral indices. In practice, you should just use the `std::map` class template provided by the Standard Library and discussed in Chapter 18 for such an associative array.

13) Use the std-20 starter code from prac08_task4_ex15p1.

========== THE END ==========