

Chapter 4

Exercise: Reading from files into structs/classes

4.1 Learning Objectives

References to literature are provided in brackets:

- using struct or class, [8, Ch. 2],
- using arrays or vector

4.2 Exercises

In this exercise we will implement further functionality to reach the final aim of automatic annotation of given ATCo transcriptions.

see directory ExeReadInfoStructures for the solution code

4.3 Exercises at a glance

Exercise 4.1 Read the transcriptions and annotations from files, store the contents in appropriate arrays and structs

Exercise 4.2 Split the day and time into year, month, day etc.

Exercise 4.3 Read the transcriptions from a given file, store the contents in appropriate variables and count the number of occurrences of the different words. Print to screen the TOP TEN occurrences, but only those which are specified in a file of the expected command types (as already done in exercise 3.2).

Exercise 4.4 Read the command annotations from a given file and print out to screen the command type, which occurs most often.

4.4 Detailed Exercise Descriptions

Exercise 4.1: You get a file with the example format, you know already from the last exercise:

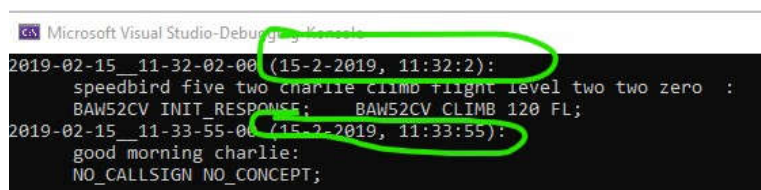
```
2019-02-15__11-32-02-00:
    speedbird five two charlie victor praha radar radar contact climb
        flight level one two zero
    BAW52CV INIT_RESPONSE
    BAW52CV CLIMB 120 FL
2019-02-15__11-32-24-00:
    scandinavian one seven six seven praha radar radar contact
        descend flight level one hundred
    SAS1767 INIT_RESPONSE
    SAS1767 DESCEND 100 FL
```

Read the contents of the file into an array (C-Array with [] of type `AtcoCommand`. `AtcoCommand` is a struct or if you prefer a class, which stores the filename (i.e. the time and date) (as string), the word sequence (as string) and the corresponding extracted commands (as array of string).

Currently you do not know, how to implement dynamic arrays. Therefore, assume that an utterance never consists of more than six commands and that a file never consists of more than 1'000 utterances.

How your solution is evaluated?

Create before final check in a screen dump of your running program, which shows on the two example files `WordSeqPlusCmds1.txt` and `WordSeqPlusCmds.txt` from my homepage the output of your program which shows that you can read the two files. Below you see an example output of the shorter file. Checkin into SVN into folder Images of this exercise. The file name should be `exercise-4-1-XX.jpg` (for jpg files). For the first image XX is 01, for the second it is 02... You need at least two images, because you have two input files. If the output does not fit into one image, create more.



```
Microsoft Visual Studio-Debugger Console
2019-02-15__11-32-02-00 (15-2-2019, 11:32:2):
    speedbird five two charlie climb flight level two two zero :
    BAW52CV INIT_RESPONSE;    BAW52CV CLIMB 120 FL;
2019-02-15__11-33-55-00 (15-2-2019, 11:33:55):
    good morning charlie:
    NO_CALLSIGN NO_CONCEPT;
```

Figure 4.1: Showing output for exercise-4.1

Exercise 4.2: Extend your type `AtcoCommand` so that it contains the date and time also not just as a string, but additionally it should contain a new type `DateTime`, which stores year, month, day, hour, minute, second and millisecond (as int). Write suitable tests to verify your splitting. How your solution is evaluated?

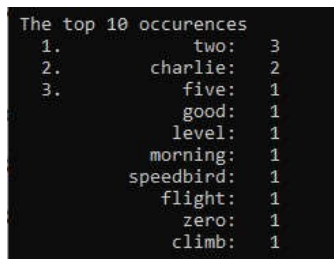
No further steps needed, if you integrate the output of the filename in day, month etc. into the previous exercise, see the green parts in Figure 4.1.

Exercise 4.3: How your solution is evaluated?

Do you have suitable Unit-Tests. I run your main, which should call or your tests and should output, that all tests are successful. I will change some code lines in your code (not in the tests). Now at least one test must fail or the whole program should crash. This shows the error in your code, i.e. the desired behaviour.

You must have at least one test, which reads a file with transcriptions and annotations and which checks the top ten occurrence. I will change your file with the transcriptions and commands (by a file from me) and I will check, if I get now the expected TOP 10 occurrences.

Create before final check in a screen dump of your running program, which shows on the two example files `WordSeqPlusCmds1.txt` and `WordSeqPlusCmds.txt` from my homepage the output of your program which shows that you can read the two files. Below you see an example output of the shorter file. Checkin into SVN into folder Images of this exercise. The file name should be `exercise-4-3-XX.jpg` (for jpg files). For the first image XX is 01, for the second it is 02... You need at least two images, because you have two input files. If the output does not fit into one image, create more.



```
The top 10 occurrences
1.      two: 3
2.      charlie: 2
3.      five: 1
        good: 1
        level: 1
        morning: 1
        speedbird: 1
        flight: 1
        zero: 1
        climb: 1
```

Figure 4.2: Showing output for exercise-4.3

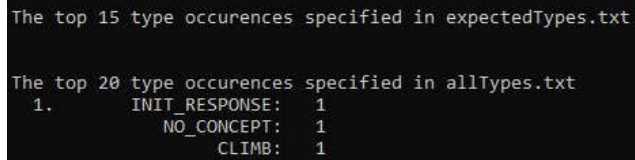
After you have finally checkin and after the deadline you get other input files, for which also create appropriate output images (increasing the image number) and check the output in to SVN. In case you need to adapt your program, to create the correct images, please document also

Exercise 4.4: How your solution is evaluated?

You must have at least one test, which reads a file with transcriptions and annotations and which checks, whether the expected command type is the most often occurring command type. I will change your file with the transcriptions and com-

mands (by a file from me) and I will check, if I get now the expected TOP 1 command type.

As before create two screen dumps. Below you see one example:



```
The top 15 type occurrences specified in expectedTypes.txt

The top 20 type occurrences specified in allTypes.txt
1.      INIT_RESPONSE: 1
        NO_CONCEPT: 1
        CLIMB: 1
```

Figure 4.3: Showing output for exercise-4.4

Consider in your image that different files with the expected commands should be shown as you see it in the example figure.