Extraction of Callsigns

Table of Contents

Learning Objectives

- Using STL
- Programming in a team, you may work in teams up to 5

Exercise at a glance

- Extract all callsigns from given utterances
- Calculate your extraction rate
- Consider also the word “*correction*”

What is a Callsign?

Extract the callsign part from a given utterance, e.g.,

- "`oscar echo india november kilo direct whisky whisky nine eight five`". The expected output would be “OEINK”.
- "`good morning lufthansa one two bravo descend eight zero`". The expected output would be “DLH12B”.
- “`gruess gott ryan_air seven seven delta kilo in radar contact`”. The expected output would be “RYR77DK”.
- `"standby":` No callsign is in this utterance. The expected output would be “NO_CALLSIGN”
- “`gruess gott lupus one one zero expect ils approach three four`”. The expected output would be “AYY110”.

These are the easy examples, which will already cover 80% to 90% of the utterances, you will get as test cases. More challenging is already

- "gruess gott austrian triple seven sierra identified climb flight level two three zero". The expected output would be "AUA777S".
- "descending eight thousand feet direct dexon speed bird twenty nine seventeen". The expected output would be "BAW2917".
- "speed bird twenty nine seventeen standby lufthansa four double alfa after air france taxi via november november 8 to delta four eight one". The expected output would be "BAW2917" and "DLH4AA". "air france" is not used as a callsign.
- The callsign is not always in the beginning: "climbing flight level two three zero austrian triple seven sierra".

You will see more and more "interesting", i.e. challenging examples.

Here you find the data structures, which could be used for mapping from a letter to the name in the NATO alphabet.
Attention: you will need the opposite map, which maps from NATO alphabet to the ASCII character. You will also find this code segments in SVN in folder
`..\AlleGruppen\VonJedemInSeineUmgebungZuKopieren\Aufgabe04\Hilfscode`.

```
static const std::unordered_map<char, std::string> lettersToNato =
{
{ 'A', "alfa" }, // it is alfa, not alpha
{ 'B', "bravo" },
{ 'C', "charlie" },
{ 'D', "delta" }, // there is also an airline 'delta', with the three letter code DAL
{ 'E', "echo" },
{ 'F', "foxtrot" }, // also fox is said
{ 'G', "golf" },
{ 'H', "hotel" },
{ 'I', "india" },
{ 'J', "juliett" },
{ 'K', "kilo" },
{ 'L', "lima" },
{ 'M', "mike" },
{ 'N', "november" },
{ 'O', "oscar" },
{ 'P', "papa" },
{ 'Q', "quebec" },
{ 'R', "romeo" },
{ 'S', "sierra" },
{ 'T', "tango" },
{ 'U', "uniform" },
{ 'V', "victor" },
{ 'W', "whiskey" },
{ 'X', "x-ray" },  // not xray
{ 'Y', "yankee" },
{ 'Z', "zulu" }  // not zoulou
};
```

The following code segments could also be helpful:

```
static const std::unordered_map<char, std::string> numbersToNato =
{
{ '1', "one" },
{ '2', "two" },
{ '3', "three" },
{ '4', "four" },
{ '5', "five" },
{ '6', "six" },
{ '7', "seven" },
{ '8', "eight" },
{ '9', "nine" },
{ '0', "zero" },
{ '.', "decimal" }
};
```

```
static const std::unordered_map<std::string, int> numbersToNatoMultipleDigits =
{
{ "ten", 10 },
{ "eleven", 11 },
{ "twelve", 12 },
{ "thirteen", 13 },
{ "fourteen", 14 },
{ "fifteen", 15 },
{ "sixteen", 16 },
{ "seventeen", 17 },
{ "eighteen", 18 },
{ "nineteen", 19 },

{ "twenty", 20 },
{ "thirty", 30 },
{ "fourty", 40 },
{ "fifty", 50 },
{ "sixty", 60 },
{ "seventy", 70 },
{ "eighty", 80 },
{ "ninety", 90 },

{ "hundred", 100 },
{ "thousand", 1000 }
};
```

In SVN in the file completeDesignators.json in folder `AlleGruppen/VonJedemInSeineUmgebungZuKopieren/10_EfficientExtraction/Hilfscode` you find a subset of the three letter airline designators[1] `DLH` stands for `hansa or lufthansa`, `GTW` for `united states of america`, `DAL` for `delta`. `BAW` for `speed_bird` or `speedbird` or `speed bird` with a blank etc.

[1] There are more than 7000 combinations, but we will use only some of them in our test data. Combinations not provided here, are not used in our test data -- I hope so. You should find

Just for understanding your task: You find here a subset in simple JSON format in SVN, e.g.:

```
{
"ABP": ["b_air"],
"ACA": ["canada"],
"AEG": ["airest", "eastern", "east air", "east"],
"AFR": ["air_france", "france"],
"BER": ["air_berlin", "berlin"],

"DLH": ["lufthansa", "hansa"],
"GEC": ["lufthansa cargo"],
"LCI": ["lufthansa india"],
"LHT": ["lufthansa technik"],

"DAL": ["delta"],
"DAT": ["deltair", "delta air"],

"MHV": ["snowcap", "snow cap"],
"NLY": ["flyniki", "fly niki", "fly_niki"]
}
```

Detailed Exercise Description

Exercise 4-0: Who is doing what in your team

Send a short description, who will do what in your team, if you are working in a team. If you are working alone this is not necessary.

If you are sending me this description until 2023-11-12 via email you have time for the whole exercise until end of the month November otherwise until 2023-11-24.

Exercise 4-1: Read Utterance and extract callsigns

Implement a function `ReadUtteranceCheckCallsign`, which reads a file in the format shown below and extracts from each utterance (word sequence) the callsign and compares, whether the extracted callsign is equal to the expected callsign (which you analyse by reading the utterance by yourself and set as desired value).

The following file with name `NumbersWithCallsignsEx1.txt` is an example:[2]

---

roughly 30 of them in the file "`completeDesignatorsShort.json`". Later you will get an additional file "`completeDesignatorsShort.json`" with a more complete list.

[2] The file structure is always:

- Line 1: one-word file name, which always starts with "20*. The line ends with a colon ":".
- Line 2: an utterance consisting of an arbitrary number of words, but at least one
- Line 3: Might contain the first word "Csgn" followed by a colon ":" and then an arbitrary number of strings separated by white spaces might follow. All in the same line. The line starting with "Csgn" might also be missing.
- Line 3, 4, … or Line 4, 5: The extracted commands. For you, only the callsign is interesting. Sometimes there might be more than one callsign, but this is the exception. These lines contain more than one word.
- And then it starts again with a one-word line with next file name.

```
2019-02-15__11-32-02-00:
  oscar echo india november kilo direct whisky whisky nine
eight five
  Csgn: DLH123 OEINK DLHABC
  OEINK DIRECT_TO WW985
2019-02-15__11-33-02-00:
  good morning lufthansa one two bravo descend eight zero
  DLH12B DESCEND 80 none
2019-02-15__11-34-02-00:
  gruess gott ryan_air seven seven delta kilo in radar contact
  RYR77DK INIT_RESPONSE
2019-02-15__11-35-02-00:
  standby
  Csgn: DLH123 OEINK DLHABC AFR257A
  NO_CALLSIGN  NO_CONCEPT
2019-02-15__11-37-02-00:
  gruess gott lupus one one zero expect ils approach three four
  AYY110 EXPECT ILS 34
2019-02-15__11-38-02-00:
  gruess gott austrian seven seven seven sierra identified
climb flight level two three zero
  AUA777S  INIT_RESPONSE
  AUA777S CLIMB 230 FL
2019-02-15__11-39-07-00:
speed bird twenty nine seventeen standby lufthansa four double
alfa after air france taxi via november november eight to
delta four eight one[3]
  BAW2917  CALL_YOU_BACK
  DLH4AA GIVE_WAY AFR none
  DLH4AA TAXI VIA N N8
  DLH4AA TAXI TO D481
2019-02-15__11-39-02-00:
  hi united states of america two fox one foxtrot identified
climb flight level two three zero
  AAL2F1F  INIT_RESPONSE
  AAL2F1F CLIMB 230 FL
```

First the keyword sequence `oscar echo india november kilo direct whisky whisky nine eight five` is read and the expected callsign `OEINK` is expected.
The function has at least three parameters:

- parameter 1 specifies the full filename of the file you want to read in from disk
- and parameter 2 is a boolean parameter. If set to `true`, the keyword sequence is printed to `cout`, then the expected callsign is printed followed by the extracted callsign. If the parameter is `false`, no output is printed to `cout`. If the function detects a deviation between extracted and expected callsign it outputs after the extracted callsign "####" and it returns `false`, otherwise `true` (i.e. if no #### is output at all).

[3] This is a long line split into multiple lines in this document. In the file it is just one line, which should make extraction for you much easier.

- The third output parameter (new class to be implemented by you) counts the number of read callsigns and also the number of correctly and wrongly extracted callsigns. [4] If parameter 2 is true, also output these values to cout.

Exercise 4-2: Unit Tests

Write **at least** three tests for the function `ReadUtteranceCheckCallsign`. In one test it should be tested, whether the expected output value `true` is returned and in one other test the expected value should be `false`. Test 3 tests e.g., whether the third parameter contains the correct and expected values. More tests are always better.

%%

Evaluation criteria for Exercise 4-1 and 4-2

- Upload at least three screen dumps of your test code for the function `ReadUtteranceCheckCallsign` with the filenames ReadUtteranceCheckCallsignTest01.jpg, ReadUtteranceCheckCallsignTest02.jpg etc.
- Run the function `ReadUtteranceCheckCallsign` also on the file NumbersWithCallsignsEx1.txt shown above.[5] Call the function with boolean parameter set to \ true. Make a screen dump of this screen output and upload it in file NumbersWithCallsignsEx1.jpg in the image folder.
- After you have uploaded your code, you will get after the deadline a new test file (currently not known to you). Run your function ReadUtteranceCheckCallsign (again with boolean parameter set to true) on it and upload the screen dump to callsignExtraction.jpg. It should also run on my computer by just changing the file.

Exercise 4-3: Consider "correction"

If the keyword sequence contains the word correction please also consider this, as described by the examples below:

We expect the following in callsign correction:

- `"oscar echo correction oscar delta india november kilo direct whisky whisky nine eight five"` --> ODINK
- `"good morning lufthansa correction speed bird one two bravo descend eight zero"` // --> BAW12B
- `"gruess gott ryan air correction lufthansa eight correction lupus seven  seven delta kilo in radar contact"` --> AYY77DK
- `"gruess gott ryan air correction lufthansa correction i call you back"` --> NO_CALLSIGN, because the callsign is not clear.

Evaluation criteria for Exercise 4-3

- Create an input file with the above four examples and run your function ReadUtteranceCheckCallsign on it with boolean parameter set to true. Upload the resulting screen dump to file `CallsignExtrWithCorrection.jpg` and check the results.

---

[4] Attention. It could happen that you expect only one callsign, but you extract three different ones and vice versa.

[5] This file does not contain the tag "Csgn:" You find the file also in data folder of "`..\AlleGruppen\VonJedemInSeineUmgebungZuKopieren\Aufgabe04`".

- Furthermore, you will get an unknown file and we might have a *competition* between me and the other teams and your team.