Chapter 7

Exercise: Extracting Numbers from word sequences

7.1 Leaning Objectives

References to literature are provided in brackets:

- Test-first, [8, Ch. 1],
- Think, Red-Bar, Green-Bar, (Refactor), [8, Ch. 1],
- using classes and inner classes,
- using different STL containers, e.g. set or vector.

7.2 Exercises

In this exercise we will implement the first real functionality to extract the concepts from a controller utterance. We will extract numbers from a sequence of words.

see directory `06_ExeNumberExtraction` for solution code

7.3 Exercises at a glance

Exercise 7.1 Extract the next number (given as a word sequence) from a sequence of words.

Exercise 7.2 Extract all numbers (given as a word sequence) from a sequence of words.

7.4 Detailed Exercise Descriptions

Exercise 7.1:

Implement a class `NumberExtractor` with the following interface.

```
class NumberExtractor
{
public:
  class ExtractedNumber
  {
  public:
  /* ... this is an inner class, i.e. a class in a class ...*/
  };

public:
  //! the numbers are extracted from astr_WordSeq,
  //  starting after the word startIndexExtractionAfter, first word starts at index 0
  NumberExtractor(std::string astr_WordSeq, int startIndexExtractionAfter=-1);

  //! same as before, but the single are already extracted into a vector
  //! the numbers are extracted from ar_allWordsAsString,
  //  starting after the word startIndexExtractionAfter, first word starts at index in vector 0
  NumberExtractor (const std::vector<std::string>& ar_allWordsAsString,
             int startIndexExtractionAfter);

  //! Extracts the next complete number from sequence of words provide in constructor
  ExtractedNumber ExtractNextFullNumber();

/* ... */
};
```

The class `ExtractedNumber` is an inner `public` class of `NumberExtractor` with the following implementation:

```
class ExtractedNumber
{
public:
  ExtractedNumber(std::string astr_seqOfNumbers = "",
           int numberToStore = -1, int numAfterDeci = -1
  );
  int GetExtractedNumberAsInt() const {
    return mi_number;
  }
  double GetExtractedNumberAsDouble() const;
  bool IsExtractedNumberValid() const {
    return mi_number >= 0;
  }
  std::string GetNumberStringSeq() const {
    return mstr_numberSeq;
  }
  bool IsExtractedNumberAnInt() const {
    return (mi_numberAfterDecimal < 0) && IsExtractedNumberValid();
  }
  //! number of words in mstr_numberSeq
  int GetNumberOfStringsForNumber() const;

private:
  //! substring of mstr_numberSeq containing the words implementing the number
  std::string mstr_numberSeq;
  //! the extracted number
  int mi_number = -1;
  //! the extracted number after word "decimal" etc. if we have extracted a double
  int mi_numberAfterDecimal = -1;
};
```

Please use this interface (without adaptations), so that code exchange with me and other teams is possible.

The functionality of the main method `ExtractNextFullNumber` is best explained by an example. If we use the string `fly_niki six hundred zulu contact tower now one two three point eight servus` the first call of `ExtractNextFullNumber` should provide `600`[1] and the next one `123.8` and the next one is nothing, i.e. `IsExtractedNumberValid` should provide for its third call `false`, whereas first and second call would result in `true`

More detailed than this explanation via words is the code itself. The following test code might help:

```
/* We test whether we can extract two numbers from a string.
*/
{
  string utter = "fly_niki six hundred zulu contact tower now "
    "one two three point eight servus";

  NumberExtractor numEx(utter);
  NumberExtractor::ExtractedNumber exNum = numEx.ExtractNextFullNumber();
  if (exNum.GetExtractedNumberAsInt() != 600 ||
    exNum.GetNumberStringSeq() != "six hundred")
  {
    return false;
  }

  exNum = numEx.ExtractNextFullNumber();
  if ( (fabs(exNum.GetExtractedNumberAsDouble() - 123.8) > 0.0001)
    || (exNum.GetNumberStringSeq() != "one two three point eight"))
  {
    return false;
  }

  exNum = numEx.ExtractNextFullNumber();
  if ((exNum.IsExtractedNumberValid())
     || (exNum.GetNumberStringSeq() != ""))
  {
    return false;
  }
  return true;

} // testGetNxtIntNumber()
```

Exercise 7.2:

[1]Later we will extend the functionality so that numbers which are part of a callsign are ignored.

Now extend the class interface of `NumberExtractor` by the method `bool PerformFullExtraction()`. This method should extract all numbers from a given string, which is set by the constructor.

As `PerformFullExtraction` just returns `true` (no error has occurred) to the outside world, we need also some interfacing methods to get access to the status of an instance of `NumberExtractor`, after `PerformFullExtraction` is called.

```
class NumberExtractor
{
public:
  //! the numbers are extracted from astr_WordSeq,
  //   starting after the word startIndexExtractionAfter, first word is number 0
  NumberExtractor(std::string astr_WordSeq, int startIndexExtractionAfter=-1);
  //! the numbers are extracted from ar_allWordsAsString,
  //   starting after the word startIndexExtractionAfter, first word is number 0
  NumberExtractor (const std::vector<std::string>& ar_allWordsAsString, int startIndexExtractionAfter);

  //! Extracts all numbers from member mstr_wordSeq and
  //   inserts into member m_extractedNumberSeq;
  bool PerformFullExtraction();
  //! Extracts the next complete number from Stream ar_strstr
  ExtractedNumber ExtractNextFullNumber();

  //! how many numbers are extracted from string provided in constructor
  int GetExtractedNumbersCnt() const {
    return static_cast<int>(m_extractedNumberSeq.size());
  }
  //! return extracted number with index as an integer (throws exception if index invalid)
  int GetNumberAsInt(int index) const {
    return m_extractedNumberSeq.at(index).GetExtractedNumberAsInt();
  }
  //! return extracted number with index as a floating point (throws exception if index invalid)
  double GetNumberAsDouble(int index) const {
    return m_extractedNumberSeq.at(index).GetExtractedNumberAsDouble();
  }
  //! is extracted number with index a valid number (throws exception if index invalid)
  bool IsNumberValid(int index) const {
    return m_extractedNumberSeq.at(index).IsExtractedNumberValid();
  }
  //! is extracted number with index a valid integer? (throws exception if index invalid)
  bool IsNumberInt(int index) const {
    return IsNumberValid(index) &&
      m_extractedNumberSeq.at(index).IsExtractedNumberAnInt();
  }
  //! is extracted number with index a valid floating point number? (throws exception if index invalid)
  bool IsNumberDouble(int index) const {
    return IsNumberValid(index) &&
      ! m_extractedNumberSeq.at(index).IsExtractedNumberAnInt();
  }
/* ... */
};
```

For testing you can use the following code

```
bool testExtractNumberThousand()
/* The following test tests the functionality of PerformFullExtraction
  by calling it with different strings (containing controller
   utterances and testing whether we get the expected
   numbers */
{
   string wordSeq = "descend one zero thousand feet";
   vector<int> expNumbers = { 10000 };
   bool b_success = true;
   NumberExtractor numEx =
      testForIntNumbers(wordSeq, expNumbers, b_success, true);

   string wordSeq2 = "on qnh one thousand zero nine";
   vector<int> expNumbers2 = { 1009 };
   bool b_success2 = true;
   numEx = testForIntNumbers(wordSeq2, expNumbers2, b_success2, true);

   string wordSeq3 = "on qnh one thousand zero twenty bye bye";
   vector<int> expNumbers3 = { 1020 };
   bool b_success3 = true;
   numEx = testForIntNumbers(wordSeq3, expNumbers3, b_success3, true);

   string wordSeq4 = "on qnh one thousand twenty eight reduce one six zero knots";
   vector<int> expNumbers4 = { 1028, 160 };
   bool b_success4 = true;
   numEx = testForIntNumbers(wordSeq4, expNumbers4, b_success4, true);

   string wordSeq5 = "on qnh one thousand zero two nine";
   vector<int> expNumbers5 = { 1029 };
   bool b_success5 = true;
   numEx = testForIntNumbers(wordSeq5, expNumbers5, b_success5, true);


   return b_success && b_success2 && b_success3
      && b_success4 && b_success5;

} // testExtractNumberThousand()
```

You need to implement `testForIntNumbers` by yourself.

The following test should help to understand the challenges for floating point numbers:

```
bool testExtractDecimalNumbers()
{
   string wordSeq = "contact director one one nine decimal eight four goodbye";
   vector<double> expNumbers = { 119.84 };
   bool b_success = true;
   NumberExtractor numEx = testForDecimalNumbers(wordSeq, expNumbers, b_success, true);

   string wordSeq2 = "contact director one one nine point eight twenty one";
   vector<double> expNumbers2 = { 119.821 };
   bool b_success2 = true;
   numEx = testForDecimalNumbers(wordSeq2, expNumbers2, b_success2, true);

   return b_success && b_success2;

} // testExtractDecimalNumbers()
```

The following code fragment shows some controller utterances from real world traffic together with the expected values:

```
    "dobry den sky_travel five eight juliett ruzyne radar radar contact on present"
    " heading descend four thousand feet qnh one zero two two", // 58, 4000, 1022

    "oscar kilo victor india kilo roger descend three thousand five hundred feet "
    "squawk seven thousand", // 3500, 7000

    "snow cab two hundred one descend eight thousand feet", // 201 ,8000

    "austrian three nine two papa descend altitude one zero thousand "
    "qnh one zero zero three", // 10000 1003
    // dummy was understood instead of decimal, here we do only extraction, correction is done later
    "contact director one one nine dummy eight goodbye" // 119, 8

    "fly_niki six hundred zulu contact tower now "
      "one two three point eight servus", // 600, 123.8

    "negative sir temperature two three instead of two one "
      "and dew point one three instead of one two", // 23, 21, 13, 12
    // 119.825 is meant, however, extraction should be 119825
    "contact director one one nine eight two five goodbye"
```

Implement the internal test functions `testForIntNumbers` and `testForDecimalNumbers` as shown above, so that an easy test with other examples is possible.

7.5 Evaluation criteria

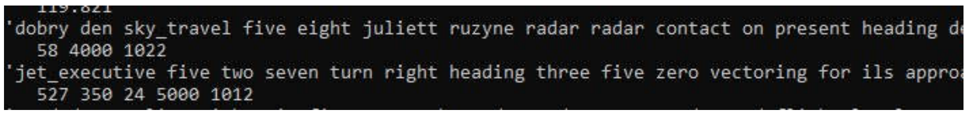
Your code should be in the folder `Aufgabe07/Code`, i.e. your screen dumps are expected in SVN in folder `Aufgabe07/Images`.

1. Upload a screen dump of your test code for the function `testForIntNumbers` into `exercise7-testForIntNumbers.jpg` .
2. Upload a screen dump of your test code for the function `testForDecimalNumbers` into `exercise7-testForDecimalNumbers.jpg` .
3. Implement a function, which calls your extraction code with the following word sequences:

```
  vector<string> wordSeq = {
    "dobry den sky_travel five eight juliett ruzyne radar radar contact"
    " on present heading descend four thousand feet qnh one zero two two",

    "jet_executive five two seven turn right heading three five zero vectoring for "
    "ils approach runway two four and descend five thousand feet qnh one zero one two",

    "good day vueling eight six five two praha radar radar contact "
    "descend flight level one zero zero current information is quebec and speed "
    "two seventy or less",

    "oscar kilo victor india kilo roger descend three thousand five hundred feet "
    "squawk seven thousand",

    "csa six six nine descend flight level one hundred turn left "
    " heading zero six five",

    "snow cab two hundred one descend eight thousand feet",
    "snow cab two hundred twenty one descend eight thousand six hundred feet",
    "snow cab two hundred twenty four "
    "descend eight thousand six hundred twenty one feet",

    "austrian three nine two papa descend altitude one zero thousand "
    "qnh one zero zero three",
    // dummy was understood instead of decimal, here we do only extraction, correction is done later
    "contact director one one nine dummy eight goodbye"
  };
```

Your implementation of `testForIntNumbers` and/or `testForDecimalNumbers` could be called. Your test code should print the input string and the extracted numbers to the screen.

Upload a screen dump `exercise7-wordSeqOutput-1.jpg` of your output for **all** the above word sequences. The following figure 7.1 shows the first lines of the expected output.



```
   119.821
'dobry den sky_travel five eight juliett ruzyne radar radar contact on present heading de
   58 4000 1022
'jet_executive five two seven turn right heading three five zero vectoring for ils approa
   527 350 24 5000 1012
```

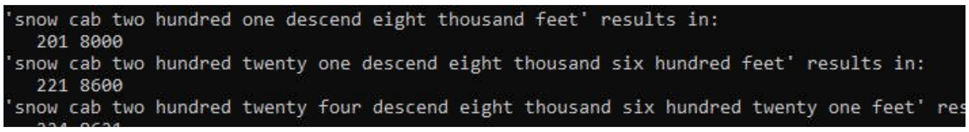
Figure 7.1: Word sequences and resulting numbers, part 1

After uploading your code and your results, you will get the full figure for self-evaluation.

4. Repeat for the following input utterances:

```
  vector<string> wordSeq = {
    "snow cab two hundred one descend eight thousand feet",
    "snow cab two hundred twenty one descend eight thousand six hundred feet",
    "snow cab two hundred twenty four descend eight thousand six hundred twenty one feet",
    "austrian three nine two papa descend altitude one zero thousand qnh one zero zero three",
    "contact director one one nine dummy eight goodbye"
  };
```

Upload a screen dump `exercise7-wordSeqOutput-2.jpg` of your output for **all** the above word sequences. The following figure 7.2 shows the first lines of the expected output.



```
'snow cab two hundred one descend eight thousand feet' results in:
   201 8000
'snow cab two hundred twenty one descend eight thousand six hundred feet' results in:
   221 8600
'snow cab two hundred twenty four descend eight thousand six hundred twenty one feet' res
```

Figure 7.2: Word sequences and resulting numbers, part 2

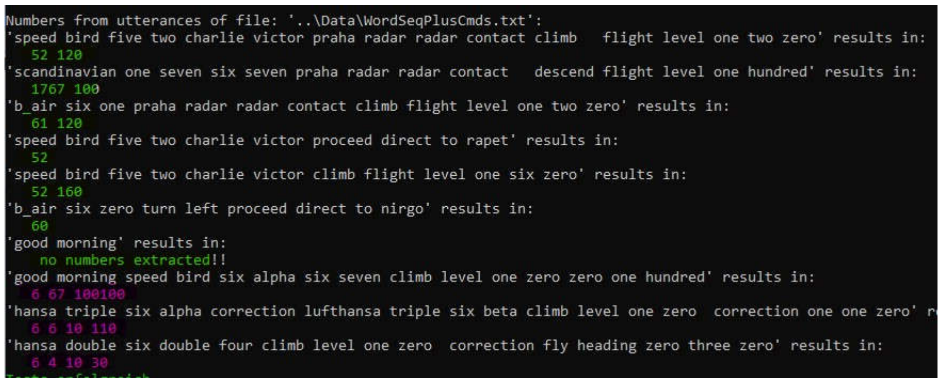
5. Implement a function `ReadUtterancesAndOutputNumbers`, which is able to read the files from the prevous exercises with filename, utterances and expected commands, i.e. it should be able to read:

```
2019-02-15__11-32-40-00:
  b_air six one praha radar radar contact climb flight level one two zero
  ABP61 INIT_RESPONSE
  ABP61 CLIMB 120 FL
2019-02-15__11-32-48-00:
  speedbird five two charlie victor proceed direct to rapet
  BAW52CV DIRECT_TO RAPET none
2019-02-15__11-33-26-00:
  speedbird five two charlie victor climb flight level one six zero
  BAW52CV CLIMB 160 FL
```

   In addition to previous exercise this function should also output the extracted numbers, as shown in the previous exercise.

   Upload a screen dump of your function `ReadUtterancesAndOutputNumbers` to `exercise7-ReadUtterancesAndOutputNumbers.jpg`.

   You will get after finishing your code and uploading the file `Fertig.txt` a new file from me. Use this file as input for `ReadUtterancesAndOutputNumbers` and upload the output as screen dump into the file `exercise7-ReadUtterOutputNumbers.jpg` to SVN into the images folder.[2]
   Attention: Before uploading mark the lines with numbers which have the same output than my ones with green colour. Mark the others with red (or something near red). See the following figure 7.3 as an example (in the example everything is correct, but just for demonstration, some lines are marked in red. In the next exercise it is expected that you also can handle correction, but not now).



```
Numbers from utterances of file: '..\Data\WordSeqPlusCmds.txt':
'speed bird five two charlie victor praha radar radar contact climb   flight level one two zero' results in:
   52 120
'scandinavian one seven six seven praha radar radar contact   descend flight level one hundred' results in:
   1767 100
'b_air six one praha radar radar contact climb flight level one two zero' results in:
   61 120
'speed bird five two charlie victor proceed direct to rapet' results in:
   52
'speed bird five two charlie victor climb flight level one six zero' results in:
   52 160
'b_air six zero turn left proceed direct to nirgo' results in:
   60
'good morning' results in:
    no numbers extracted!!
'good morning speed bird six alpha six seven climb level one zero zero one hundred' results in:
   6 67 100100
'hansa triple six alpha correction lufthansa triple six beta climb level one zero  correction one one zero' r
   6 6 10 110
'hansa double six double four climb level one zero  correction fly heading zero three zero' results in:
   6 4 10 30
```

Figure 7.3: Word sequences and resulting numbers in green and red colours

[2]Bachelor and Master student will get different files. Well you will get both two files, but you expected to use the correct one.