11th April 2019

**2. assignment/10. task**

**Sadi Mamedov**

LHXOTB

sadimamedov7@gmail.com Group 2

# Task

The results of the National Angler's Championship is stored in a text file. Each line of the file contains the identifier of the participant and the championship (strings without whitespace), and the list of the caught fish, which are stored as pairs: (the kind of the fish, the size of the fish). The kind of the fish is a string without whitespace, its size is a natural number. The data in a line are separated by whitespace. The lines of the text file are sorted according to the identifier of the participant. You can assume that the text file is correct.

An example for a line of the text file:

James BigLakeChampionship Tuna 50 Salmon 20 Sardine 5 Tuna 100

Is it true that all anglers caught tuna at at least two championships?.

# Plan of the main program

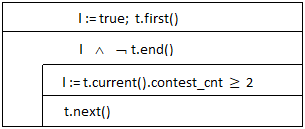
*A = (t : Enor(Angler), l :* 𝕃 *)*

*Angler =* ***rec****(name: String, contest\_cnt:* ℕ*)*

*Pre =( t = t’)*

*Post =( l =* ***∀*** *SEARCH e.contest\_cnt ≥ 2)*

*e∈t’*



# Enumerator of Anglers

|  |  |
| --- | --- |
| *enor(Angler)* | *first(), next(), current(), end()* |
| *tt* : *enor*(*Contest*)  *cur* : *Angler*  *end* : 𝕃 | *first() ~ tt.first(); tt.next() next() ~ see below current()* ~ *cur*  *end()* ~ *end* |

Operation *next()* of *enor(*Angler*)* has to solve the following problem:

Get the next angler of which it is have to be decided whether he caught a tuna on at least two of his competitions, therefore, for skillful angler it results an increment on counter by one for each competiton. For this purpose, the anglers have to be enumerated with number of successful competitons (which angler caught a tuna on how many competitons). It classifies as *Contest =* ***rec****(angler:String, championship:String, exist:𝕃)*) data structure. The first competition of the actual angler is already stored in *tt.current()*, neither *tt.first()*, nor *tt.next()* is needed. The enumeration lasts as long as the same angler’s competitions are read by operation *tt.next()*.

*Anext = (tt:enor(Contest), end:*𝕃 *, cur : Angler, contest\_cnt:* ℕ*)*

*Prenext =* ( *tt = tt1*)

*Postnext =* (*end = tt.end()*  *cur.contest\_cnt* = ∑ 𝒆.***exist*** 1)

𝒆∈ *tt1*

# next()

# 

***Enumerator of Competitions***

|  |  |
| --- | --- |
| *enor(Contest)* | *first(), next(), current(), end()* |
| *f* : *infile*(*Line*)  *cur* : *Contest*  *end* : 𝕃 | *first() ~ see below next() ~ see below current()* ~ *cur*  *end()* ~ *end* |

Operations *first()* and *next()* of *Enor(*Contest*)* are the same and they have to solve the following problem: Read the next line of the input file *f*. If there are no more lines, then variable *end* should be true. If there are more lines, then get the name of the competitor and decide whether exist the word „tuna” for the the corresponding competitor.

*Anext = (f: infile(Line), end:*𝕃*, cur:Contest) Line = seq(Word) Prenext =* ( *f = f1*)

*Postnext =* ( *sf, df, f = read(f1)*  *end=(sf=abnorm)* 

*end* → *cur.angler* = “first word of df”  *cur.championship* = “second word of df”  *cur.exist* = “if exists word ’tuna’ in df” )

In the implementation, the two classes of the two above enumerator objects (*t* and *tt*) are placed into separate compilation units.

# Testing plan

Three algorithmic patterns are used in the solution: optimist linear search, counting and linear search.

1. Test cases of deciding whether all contests are successful (optimistic linear search):

based on the **length of the interval**:

* 1. Empty file.
  2. One skillful angler, two contests.
  3. One unskillful angler, two contests.
  4. More anglers, more contests.

based on the **beginning and the end of the interval**:

* 1. Only first angler is skillful for two contests.
  2. Only last angler is skillful for two contest.

based on the **pattern**:

* 1. All anglers skillful, but only first.
  2. All anglers skillful, but only last.
  3. There is a skillful angler.

1. Test cases of counting successful contests (counting):

based on the **length of the interval**:

* 1. No successful contest.
  2. Multiple successful contests.

based on beginning **of the interval**

* 1. Enough successful contests for the first angler.
  2. Not enough successful contests for the first angler.

1. Test cases for deciding whether angler is skillful (linear search):

based on the **length of the interval**:

* 1. A line without catches.
  2. A line of one catch.
  3. A line of more catches.