Teréz A. Várkonyi
NEPTUNCODE
nick@inf.elte.hu
Group 0

2. assignment/0. task

11th February 2019

Task

At every competition of the National Angling Championship, the results of the competitors were recorded and put into a text file. Every line of the file contains the name of the angler the ID of the competition (string without spaces), and the species and the size of the caught fishes (pair of a string without spaces and a natural number). Data is separated by space or tab. The lines of the file are ordered by the name of the anglers. The file is assumed to be in the correct form. Sample line in the file:

PETER LAC0512 carp 45 carp 53 catfish 96

Give an angler who caught at least two catfishes on every competition he participated in.

Plan of the main program

A = (t : Enor(Angler), l : $\mathbb{L}$*, ID : String)* *Angler =* ***rec****(ID:String, skillful:*$\mathbb{L}$*)*
Pre =(t = t')
Post =($l, ID = \underset{e\in t'}{\boldsymbol{SEARCH}}(e.skillful)$*)*

```
l:=false; t.first()
¬l ∧ ¬t.end()
    l, ID := t.current().skillful, t.current().ID
    t.next()
```

Enumerator of Anglers

enor(Angler)	*first(), next(), current(), end()*

tt : *enor*(*Competition*)
act : *Angler*
end : $\mathbb{L}$
	first() ~ *tt.first(); tt.next()*
	next() ~ *see below*
	current() ~ *act*
	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 2 catfishes on all of his competitions. For this purpose, the anglers have to be enumerated with competition-results (on which competition which angler how many catfish he caught). It results in a *Competition* = ***rec****(angler:String, competition:String, count:*$\mathbb{N}$*)*) 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()*.

A^{next} = (tt:enor(Competition), end:$\mathbb{L}$, akt:Angler)
Pre^{next} = (tt = tt^1)
$Post^{next}$ = (end = tt.end() $\wedge$ ¬end → $\boldsymbol{act.skillful} = \bigwedge_{\boldsymbol{e\in t'}} (\boldsymbol{e.count} \geq \boldsymbol{2})$))

next()

```
end:= tt.end()
if ¬end
    act.ID:=tt.current().angler
    akt.skillful:=tt.current().count ≥ 2
    while ¬tt.end() ∧ tt.current().angler=act.ID
        akt.skillful:= akt.skillful ∧ tt.current().count ≥ 2
        tt.next()
else
    SKIP
```

Enumerator of Competitions

enor(Competition)	*first(), next(), current(), end()*
f : *infile*(*Line*) *act* : *Competition* *end* : $\mathbb{L}$	*first()* ~ *see below* *next()* ~ *see below* *current()* ~ *act* *end()* ~ *end*

Operations *first()* and *next()* of *Enor(*Competition*)* 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 the ID of the competition and count the word „catfish”.

A^{next} = (f: infile(Line), end:$\mathbb{L}$, act:Competition) Line = seq(Word)
Pre^{next} = (f = f^1)
$Post^{next}$ = (sf, df, f = read(f^1) $\wedge$ end=(sf=abnorm) $\wedge$
¬end → act.angler = "first word of df" $\wedge$
act.competition = "second word of df" $\wedge$
act.count = "number of word 'catfish' 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: linear search, optimist linear search, and counting.

A. Test cases for searching the skillful angler (linear search):

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

1. Empty file.
2. One angler.
3. More anglers.

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

4. First angler is skillful.
5. Only the last angler is skillful.

based on the **pattern**:

1. There is a skillful angler.
2. There is no skillful angler.
3. There are more skillful anglers.

B. Test cases of deciding if an angler is skillful (optimist linear search):

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

1. No catch.
2. One competition of one angler.
3. More competitions of one angler.

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

4. An angler did not catch two catfishes on his first competition, but he did catch on the rest.
5. An angler did not catch two catfishes on his last competition, but he did catch on the rest.

based on the **pattern**:

4. He did not catch two catfishes on any competition.
5. He caught two catfishes on one of the competitions.
6. He caught two catfishes on some of the competitions.
7. He caught two catfishes on every competition.

C. Test cases of counting the catfishes on the competition of one angler (counting):

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

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

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

4. A line the first catch of which is a catfish.
5. A line the last catch of which is a catfish.

based on the **pattern**:

7. A line without caught catfish.
8. A line containing one caught catfish.
9. A line containing more caught catfishes.