# OOP Khongorzul Khenchbish: Documentation, 2nd assignment 1.

**Khongorzul Khenchbish** 5th April 2021
R8HP78
r8hp78@inf.elte.hu
Group 1

## 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 contest ID. The file is assumed to be in the correct form. Sample line in the file:

Peter LAC0512 carp 45 carp 53 catfish 96

(1) Is there an angler who has caught only and at least three catfishes on one of his contests? Give the contest ID, too.
(2) How many contests there are where at least one angler has caught only and at least three catfishes?

### *(1) First part*

### *Plan of the main program:*

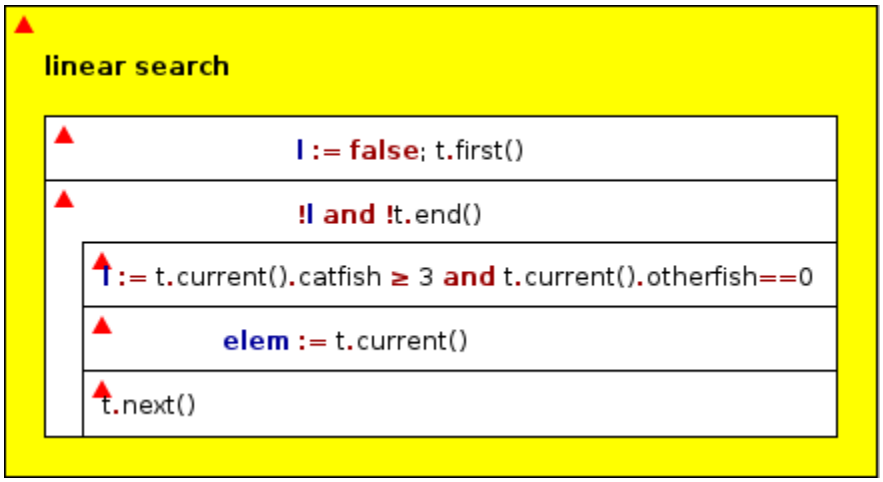
A = ( f : infile(Line), l : $\mathbb{L}$, elem: Contest )
Line = rec (anglerName : String, contestId: String, catches : Catch*)
Catch = rec (species : String, size : $\mathbb{N}$)
Contest = rec (anglerName : String, contestId: String, catfish : $\mathbb{N}$, otherfish : $\mathbb{N}$)

New state space:

A = ( t : enor(Contest), l : $\mathbb{L}$, elem : Contest )

Pre =( t = t' )

Post =( $l$,$elem$ =$\boldsymbol{SEARCH}_{e \in t'}$ ($e$. $c$atfish≥ 3 ∧ $e.otherfish == 0$))




Analogy: linear search, custom enumerator

E ~ Contest

cond(e) ~ e.catfish ≥ 3 ∧ $e.otherfish = 0$

## $Enumerator\ of\ contests$[1]

| *enor(Contest)* | *first(), next(), current(), end()* |
|---|---|
| *f : infile(Line)*<br>*cur : Contest*<br>*end : $\mathbb{L}$* | *first()* ~ *next()*<br>*next()* ~ *see below*<br>*current()* ~ **return** *cur*<br>*end()* ~ **return** *end* |

*Status = { norm, abnorm }*

In enor(Contest), operations first() and next() are the same. They have to solve the following task: read the next line of the textfile (f sequential input file). If there is no

more, then variable $end$ gets true. If there is any, the current angler's name and the contest ID can be extracted. Then, the "catfish" species can be counted in the catches.

$A^{next} = (f: infile(Line),\ cur: Contest,\ end: L)$

$Pre^{next} = (f = f')$

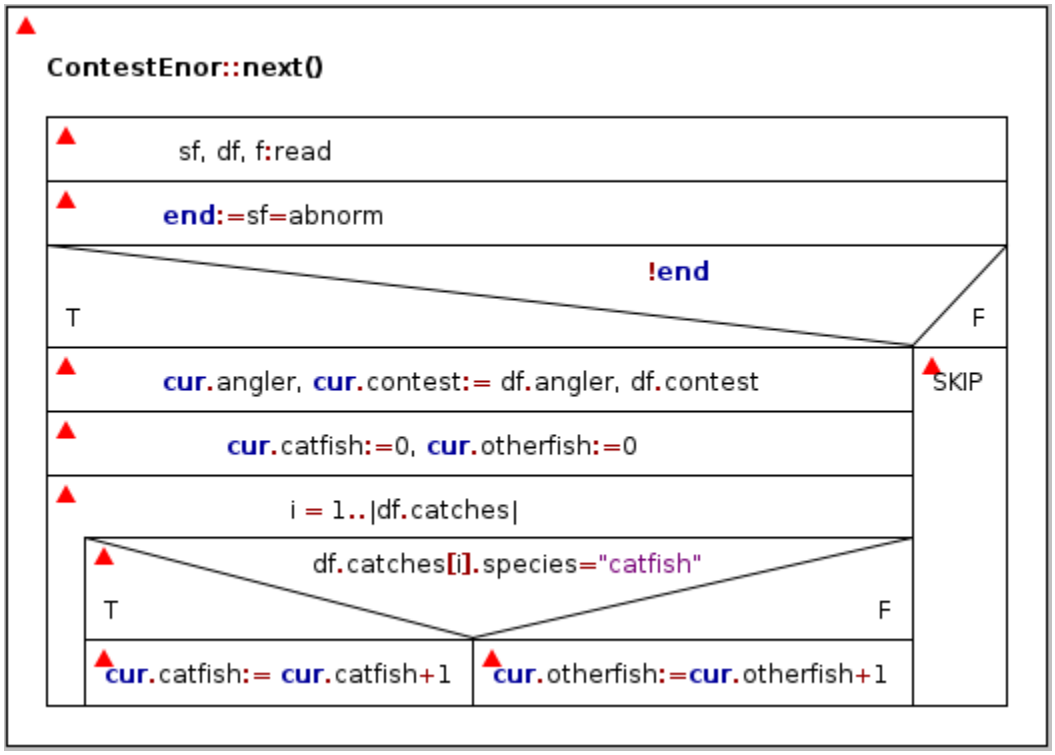
$Post^{next} = (sf,\ df,\ f = read(f')\ \wedge end = (sf = abnorm)$

$\wedge \neg end \rightarrow cur.anglerName = df.anglerName \wedge cur.contestId = df.contestId$

$\wedge\ cur.catfish = \sum_{i \in [1..|df.catches|]} g(i)\ ) \wedge cur.otherfish = \sum_{i \in [1..|df.catches|]} t(i)\ )$

$g(i) = \{1\ \ if\ df.catches[i].species = "catfish",\ \ otherwise\ 0\}$

$t(i) = \{1\ \ if\ df.catches[i].species = "otherfish",\ \ otherwise\ 0\}$




Analogy: counting, on interval enumerator

t: [m .. n ] ~ i : [1 .. |df.catches|]

cond(i) ~ df.catches[i].species="catfish"

c ~ cur.catfish, cur.otherfish

## *(2) Second part*

## *Plan of the main program:*

A = ( f : infile(Line), contestCounter : $\mathbb{N}$)

Line = rec (anglerName : String, contestId: String, catches : Catch*)

Catch = rec (species : String, size : $\mathbb{N}$)

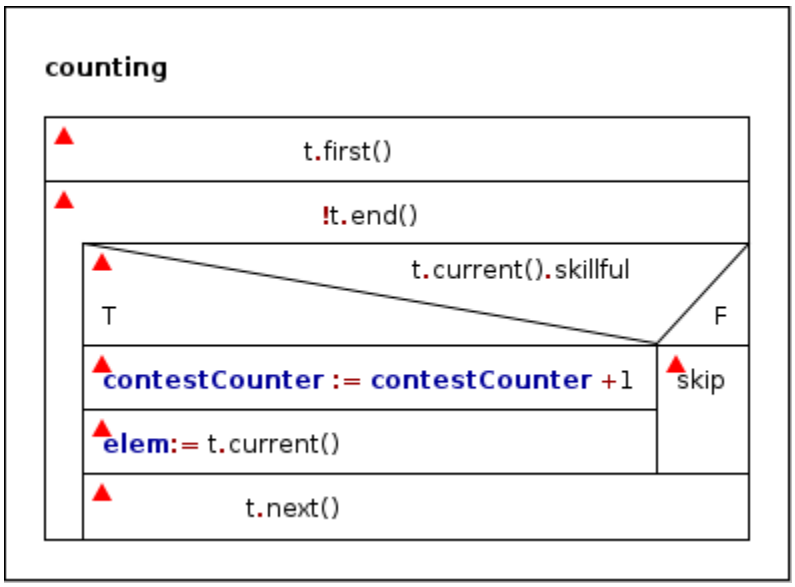
New state space:

A = ( t : enor(Angler), contestCounter : $\mathbb{N}$,

Angler = rec(contestId: String, skillful : $\mathbb{L}$)

Pre =( t = t' )

Post =( contestCounter =$\sum_{e \in t'}^{e.skillful} 1$)




Analogy: counting, on interval enumerator

E: Angler

cond(e) ~ e.skillful

$Enumerator\ of\ Anglers^2$

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

*Contest = rec (angler : String, contest : String, counter : ℕ)*

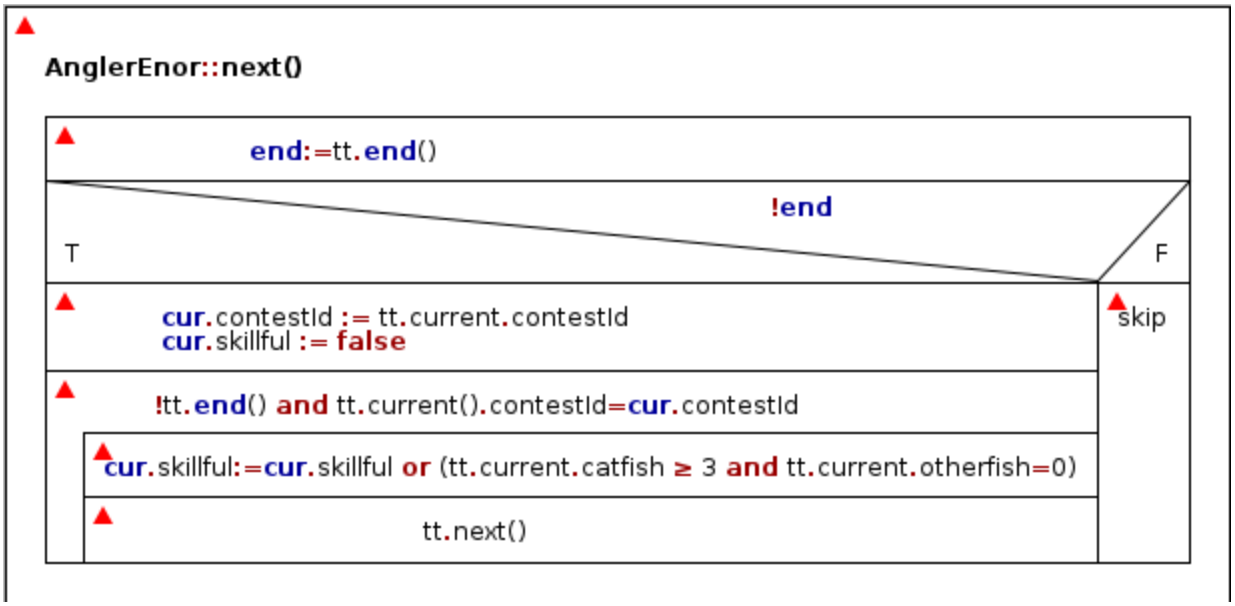
In enor(Angler), operation next() has to solve the following task: Give the next angler and decide if he has caught only and more than 3 catfishes on one of his contest. To do the decision, the enumerator of the first part can be used (enor(Contest)) which processes one line of the input file and calculates how many catfishes the angler has caught on one contest. At the beginning of operation next(), the current item of the Contest enumerator is the first contest of the angler to be processed, so for getting tt.current(), tt.first() and tt.next() are not needed. The enumeration runs as long as the same angler's contests are "read" with tt.next().

$$A^{next} = (tt\colon enor(Contest),\ cur\colon Angler,\ end\colon L)$$

$$Pre^{next} = (tt = tt')$$

$$Post^{next} = (end = tt'.end() \wedge \neg end \rightarrow (contestCounter = \sum_{e \in tt'}^{e.species="catfish"} 1 \wedge \sum_{e \in tt'}^{e.species="otherfish"} 1 \wedge$$

$$(cur.skillful,\ tt) = \vee_{\substack{e.contestId=cur.contestId \\ e \in (tt'.current(),\ tt')}} (e.catfish \geq 3 \wedge e.otherfish = 0))$$




Analogy: counting (linear search), custom enumerator

t:enor(E) ~ tt:enor(Contest) without first as long as e.contestId = cur.contestId

eϵt' ~ e ϵ (tt'.current(), tt')

s ~ e.skillful

(H,+,0) ~ (𝕃,∨, false)

## *Testing*

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

A. **Linear search** in the first part:

– Searching for one angler who has caught only and more than 3 catfishes.

**length-based:**

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

**first and last-based:**

4. The first angler meets the requirement.
5. The last angler meets the requirement.

**pattern-based:**

1. There doesn't exist such angler that meet the requirement.

2. There exist an angler who meet the requirement.
3. There are more anglers who meet the requirement.

B. **Counting** in the first part

- Number of the caught catfishes | otherfishes on one contest (counting)

**length-based:**

1. Line without catches.
2. Line with one catch.
3. More catches.

**first and last-based:**

4. Line with catches, the first contest has catfish.
5. Line with catches, the last contest has catfish.

**pattern-based:**

7. Line without catfish (could have otherfish).
8. Line with 2 catfishes and 1 otherfish ~ wrong pattern.
9. Line with only and more catfishes.

C. **Counting** on the second part:

– Count the number of contest where angler meet criteria.

**length-based:**

1. Line without contest.
2. One contest.
3. More contests.

**first and last-based:**

4. The first contest meet requirement.
5. The last contest meet requirement.

**pattern-based:**

6. There is no such contest.
7. There is only one contest.
8. There are more contest.

D. **Linear search** in the second part:

Deciding if the angler is skillful (linear search):

**length-based:**

1. No angler.
2. One contest with one angler.

3. One contest with more angler.

**first and last-based:**
4. when the first angler of the contest is skillful.
5. when the last angler of the contest is skillful.

**pattern-based:**
6. There exist no skillful angler.
7. There exist an angler that meet requirement.
8. There exist more angler that meet requirement.