OOP ASSIGNMENT 2

Jamala Khaligova A0JCKR

APRIL 8, 2020

Jamala Khaligova 2nd assignment/6th task 8th April 2020

A0JCKR

cemale_2000@mail.ru

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 name of the participants. 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

Was there an angler who caught a sardine at all the championships where he/she participated?

Main program:

A = (t : Enor(Participant), l : $\mathbb{L}$, ID : String) Participant = rec(ID:String, sardineCaughter:$\mathbb{L}$)

Pre =(t = t')

Post =(l,ID = $\boldsymbol{SEARCH}_{e\in t'}$ $(e.\text{sardineCaughter})$)

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

Enumerator of Participants

enor(Participant)	first(), next(), current(), end()

tt : enor(Competition)
current : Participant
end : $\mathbb{L}$
	first() ~ tt.first(); tt.next()
	next() ~ see below
	current() ~ current
	end() ~ end

Operation next() of Enor(Participant) has to solve the following problem:

Get the next participant of which it is have to be decided whether he caught any sardine on all of his competitions. For this purpose, the participants have to be enumerated with competition-results (on which competition which participant caught sardine). It results in a Competition = rec(participant:String, competition:String, sardineCaught : $\mathbb{L}$)) data structure. The first competition of the actual participant is already stored in tt.current(), neither tt.first(), nor tt.next() is needed. The enumeration lasts as long as the same participant's competitions are read by operation tt.next().

A^{next} = (tt:enor(Competition), end:$\mathbb{L}$, current:Participant)

Pre^{next} = (tt = tt^1)

$Post^{next}$ = (end = tt.end() $\wedge$ $\neg$end $\rightarrow$ current. sardineCaughter = $\bigwedge_{e \in t'}$ ($\boldsymbol{e}$. sardineCaught))

```
end := tt.end()
if ¬end
    current.ID:=tt.current().participant
    current.sardineCaughter:=tt.current().sardineCaught
    while ¬tt.end() ∧ tt.current().participant=current.ID
        current.sardineCaughter:=
        current.sardineCaughter ∧ tt.current().sardineCaught
        tt.next()
else
    SKIP
```

Enumerator of Competitions

Line =seq(Word)

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

Pre^{next} = (f = f^1)

$Post^{next}$ = (sf, df, f = read(f^1) $\wedge$ end=(sf=abnorm) $\wedge$ $\neg$end $\rightarrow$ current.participant = "first word of df" $\wedge$ current.competition = "second word of df" $\wedge$ current.sardineCaughter = "true if the word 'sardine' exits 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 linear search.

A. Test cases for searching the sardine caughter (linear search): based on the length of the interval:

1. Empty file.

2. One participant.

3. More participants. based on the beginning and the end of the interval:

4. First participant is sardine caughter.

5. Only the last participant is sardine caughter.

based on the pattern:

1. There is a sardine caughter.

2. There is no sardine caughter.

3. There are more sardine caughters.

B. Test cases of deciding if an participant is sardine caughter (optimist linear search): based on the length of the interval:

1. No catch.

2. One competition of one participant.

3. More competitions of one participant based on the beginning and the end of the interval:

4. A participant didn't catch any sardine on his first competition, but he caught on the rest.

5. A participant didn't catch any sardine on his last competition, but he caught on the rest.

based on the pattern:

1. He didn't catch any sardine on any competition.

2. He caught sardines on one of the competitions.

3. He caught sardines on some of the competitions.

4. He caught sardines on every competition.

C. Test cases of existence of sardine on the competition of one participant (decision/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. based on the beginning and the end of the interval:

4. A line the first catch of which is a sardine.

5. A line the last catch of which is a sardine