Ducumentation for the second assignment

Chen Danqing EPRRZ8

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 championship. 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

At which championship has the most fish been caught?

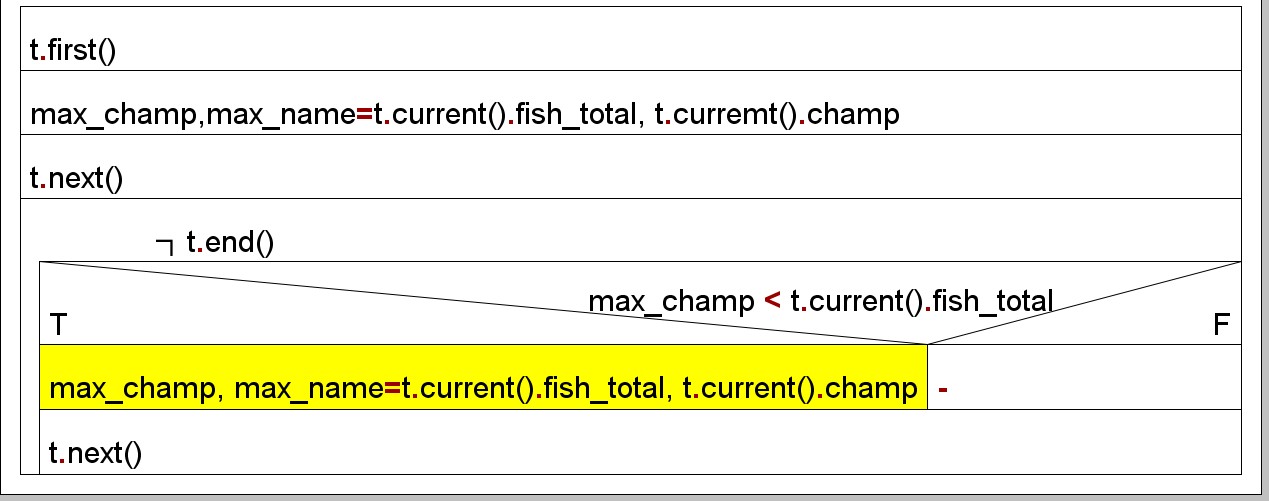
Plan of the main program:

A=(t:Enor(Champ), max\_name:String)

Champ=rec(champ:String, fish\_total=N)

Pre=(t=t` ∧ |t|>=0)

Post=max\_name=MAX eꞒt` e.champ



Enumerator of Champs:

|  |  |
| --- | --- |
| enor(Champ) | first(), next(0, current(), end() |
| \_tt:enor(Contest)  Act:Champ | first() ~\_tt.first(), \_.next()  next() ~ see below  current() ~act  end() ~end |

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

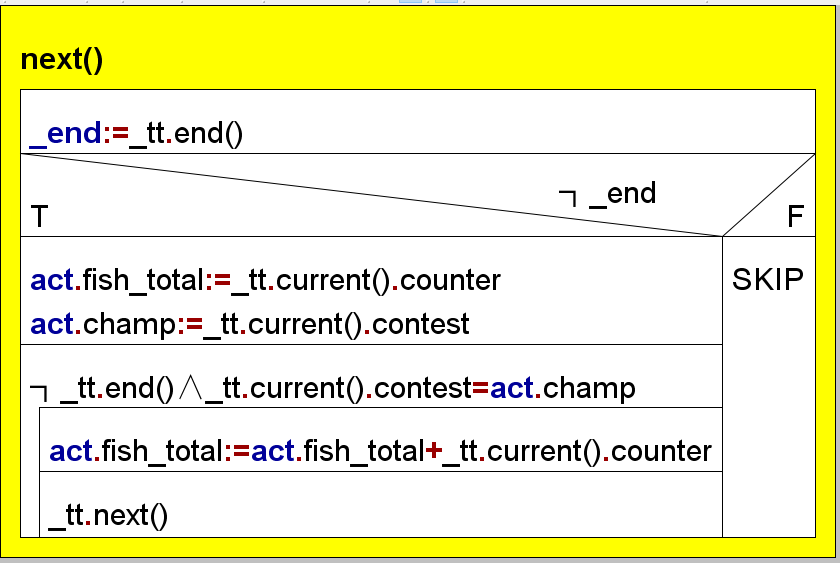
Get the next championship(champ) of which it has to sum up the total fish caught in that championship. Hence, the champs have to be enumerated with championship(contest) results (on which championship in total how many fishes are caught). It results in a Contest=rec(angler:String, contest:String, counter:N) data structure. The first championship of the actual contest is already stored in \_tt.current(), neither \_tt.first(), nor \_tt.next() is needed. The enumeration lasts as long as the same Championships are read by operation \_tt.next().

Anext=(\_tt:enor(Contest),\_end:L,act:Champ)

Prenext=(\_tt=\_tt1)

Post next=(\_end=\_tt.end()∧┐\_end -->

act.fish\_total+=(e.counter)eꞒ\_tt1)



Enumerator of Contest:

|  |  |
| --- | --- |
| enor(Contest) | first(), next(), current(0, end() |
| \_f:infile(line)  act:Contest  \_end: L | first() ~see below  next() ~see below  current() ~act  end() ~\_end |

Operation 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 championship and count the number of fishes.

Anext=(\_f:infile(line),\_end:L,act:Contest)

Prenext=(f=f1)

Postnext=(sf, df, f=read(f1)∧

\_end=(sf=abnorm)∧

┐\_end->act.angler=”first word of df”∧

act.contest=”second word of df”∧

act.counter=”number of fishes in df”)

In the implementation, the two classes of the two above enumerator object (t and \_tt) are placed into separate compilation units.

Testing plan:

TEST\_CASE 1: “no catch”

In this case we check the file with contest but in none of the contest did any anglers catch any fish

TEST\_CASE 2: “1 contest”

In this case there is only one contest in the test file.and we need to check if the most fish that is caught is indeed in the only championship in the file.

TEST\_CASE3: “more contests”

Here there are more than one contests and we have to check if the algorithm for max search indeed works correctly in the program.

TEST\_CASE4 : “check if the enumerator works correctly”

Here we use the enumerator and check if indeed with each enumerator, the fish caught in each championship is indeed the correct numbers.