Bodó József 2.beadandó/6.feladat 2020.április.9.

I7P4UQ

[jozsi2000@gmail.com](mailto:jozsi2000@gmail.com)

3.csoport

**Feladat**

Az Országos Horgász Bajnokság mindegyik versenyén feljegyezték a versenyzők eredményeit, és egy szöveges állományban rögzítették az adatokat. Ennek minden sora tartalmazza a versenyző horgász és a verseny azonosítóját (szóközök nélküli sztring), majd halfajta-súly párok (szóközök nélküli sztring és valósszám párok) formájában az adott versenyen elért fogásokat. Az adatok szóközökkel vagy tabulátorjelekkel vannak egy soron belül elválasztva. A szöveges állomány sorait a horgászok szerint rendezték. Feltehetjük, hogy a szöveges állomány helyesen van kitöltve. Példa az állomány egy sorára:

JANIBÁ Kiliti0512 Ponty 4.5 Ponty 5.3 Harcsa 9.6

(1) Hány olyan sora van a fájlnak, melyben a fogott halak között a harcsa szerepel?

(2) Volt-e olyan horgász, aki mindegyik versenyén fogott harcsát?

**(1) Részfeladat megoldása:**

**Főprogram terve:**

*A = (fishE : enor(Fisherman), count:* ℕ*)*

*Fisherman =* ***rec****(name:String, contest:String, hasCaughtCatfish:* 𝕃*)*

*Ef =(fishE = t’)*

*Uf =(* count =**Σ**𝑒∈𝑡′(*𝑒.hasCaughtCatfish == true*)*)*

|  |  |  |
| --- | --- | --- |
| c:=0; fishE.first() | | |
|  | ¬fishE.end() | |
|  | fishE.current().hasCaughtCatfish | |
|  | c:=c+1 | - |
|  | fishE.next() | |

**Fisherman felsorolója:**

|  |  |
| --- | --- |
| enor(Fisherman) | first(), next(), current(), end() |
| f: infile(Sor)  curr: Fisherman  sf : Status | first() *~ next()*  *next() ~ ld. külön*  current() *~ curr*  *end() ~ sf==abnorm* |

*Az sf egy Status felsorolási típusnak az értékét adja vissza, ami vagy norm vagy abnorm.*

Az *enor(Fisherman) first()* és *next()* műveletei megegyeznek, és az alábbi feladatot kell megoldaniuk: *olvassuk be a szöveges állomány (az f szekvenciális inputfájl) soron következő sorát. Ha ilyen nincs, akkor az sf változó értéke legyen abnorm, így az end() hamisat ad. Ha van, akkor ezt a sort sztringek sorozatának tekintve kiszedhetjük belőle a horgász és a verseny azonosítóját, majd az ezeket követő részben eldönthetjük, hogy található-e a sorban „Harcsa” szó.*

**next() specifikációja:**

*Anext = (f: infile(Sor), end:*𝕃*, cur:Verseny) Sor = String\**

*Efnext =* ( *f = f’)*

*Ufnext =* ( *sf, df, f = read(f’)* ∧ *end=(sf=abnorm)* ∧ ¬*end* → *curr.name* = *df*1 ∧ *curr.contest* = *df*2 ∧ *curr.hasCaughtCatfish* = **SEARCHi**∈[𝟑..∣𝒅𝒇∣]df[i] ==”Harcsa”)

|  |  |  |
| --- | --- | --- |
| sf,df,f:read | | |
| end := sf==abnorm | | |
| curr.name, curr.contest := df1, df2 | | |
| curr.hasCaughtCatfish := false | | |
| i:= 3 | | |
|  | i < |df| ∧ ¬ curr.hasCaughtCatfish | |
|  | df[i] ==” Harcsa” | |
|  | curr.hasCaughtCatfish := true | - |

**(2) Részfeladat megoldása:**

**Főprogram terve:**

*A = (fishCE : enor(FishermanContest), wasThere:* 𝕃, fishermanName: String*)*

*FishermanContest =* ***rec****(name:String, allContest:* 𝕃*)*

*Ef =( fishCE = t’)*

*Uf =( (*wasThere,fishermanName) =**SEARCH**𝑒∈𝑡′(*𝑒.allContest == true*)*)*

|  |  |
| --- | --- |
| wasThere = false; fishCE.first(); | |
|  | ¬wasThere ∧ ¬fishCE.end() |
|  | wasThere := fishCE.current().allContest |
|  | fishermanName := fishCE.current().name |
|  | fishCE.next() |

**FishermanContest felsorolója:**

|  |  |
| --- | --- |
| enor(FishermanContest) | first(), next(), current(), end() |
| actFM: enor(Fisherman)  curr: enor(FishermanContest)  end: 𝕃 | first() *~ actFM.First(), next()*  *next() ~ ld. külön*  current() *~ curr*  *end() ~ end* |

Az *enor(FishermanContest) next()* műveletnek az alábbi feladatot kell megoldania:

Állítsuk elő a soron következő horgászt, akiről el kell dönteni, hogy vajon minden versenyén fogott-e harcsát. Ehhez fel tudjuk használni az első részfeladat megoldásához készített felsorolót (*enor(Fisherman)*), mely az input fájl egy sorát értelmezi, és megadja, hogy az adott versenyen az adott horgász fogott-e harcsát. A *next()* művelet végrehajtásakor a *Fisherman* felsoroló (ha még nem ért a felsorolás végére) már tartalmazza azt a horgászt, és annak első versenyét, amelyet a *next()* feldolgoz majd, így, a *actFM.current()* értékét már ismerjük, ehhez nem kell sem a *actFM.first()*, sem a *actFM.next()*. A felsorolás csak addig tart, amíg ugyanazon horgász versenyeit „olvassuk” a *actFM.next()* művelettel.

**next() specifikációja:**

*Anext = (actFM:enor(Fisherman), end:*𝕃*, curr:FishermanContest)*

*Efnext =* ( actFM *= tt’*)

*Ufnext = (end = tt’.end()* ∧ ¬*end* → (*curr.name := tt’.current().name* ∧ curr.allContest =**∀**SEARCHe∈tt’ (e.hasCaughtCatfish == true)))

|  |  |  |  |
| --- | --- | --- | --- |
| end := actFM.end() | | | |
| ¬*end* | | | |
| curr.name := actFM.current().name  curr.allContest := true | | | SKIP |
|  | ¬actFM.*end* ∧ actFM.current().name == curr.name | |
|  | actFM.current().hasCaughtCatfish == false | |
|  | curr.allContest := false | - |
|  | actFM.next() | |

**Tesztelési terv:**

A megoldásban három programozási tételt alkalmaztunk: lineáris keresés, optimista lineáris keresés, számlálás.

1. Első feladat számlálásának tesztesetei:

-Hány olyan horgászat történt, ahol fogtak ki harcsát? -

* 1. üres fájl
  2. olyan fájl, aminek csak egy sora van, és nem fogtak harcsát
  3. olyan fájl, aminek csak egy sora van, és fogtak harcsát
  4. olyan fájl, aminek több sora van, és nem fogtak harcsát
  5. olyan fájl, aminek több sora van, és csak a legelső sorban fogtak harcsát
  6. olyan fájl, aminek több sora van, és mindenki fogott harcsát
  7. olyan fájl, aminek több sora van, és csak a legutolsó sorban fogtak harcsát

1. Egy horgász fogott e harcsát egy versenyen (lineáris keresés):
   1. üres fájl
   2. fogott harcsát
   3. nem fogott harcsát
2. Második feladat lineáris keresésének tesztesetei:

-Olyan horgász keresése, aki minden versenyén fogott harcsát-

* 1. üres fájl
  2. olyan fájl, aminek egy sor van, egy versenyző, és nem fogott harcsát
  3. olyan fájl, aminek egy sor van, egy versenyző, és fogott harcsát
  4. olyan fájl, aminek több sora van, több versenyző, egy verseny, de senki sem fogott harcsát
  5. olyan fájl, aminek több sora van, több versenyző, egy verseny, az első fogott harcsát
  6. olyan fájl, aminek több sora van, több versenyző, egy verseny, az utolsó fogott harcsát
  7. olyan fájl, aminek több sora van, több versenyző, több verseny, senki nem fogott harcsát
  8. olyan fájl, aminek több sora van, több versenyző, több verseny, az első fogott minden versenyén harcsát
  9. olyan fájl, aminek több sora van, több versenyző, több verseny, az utolsó fogott minden versenyén harcsát
  10. olyan fájl, aminek több sora van, több versenyző, több verseny, többen fogtak minden versenyükön harcsát

1. Egy horgász minden versenyén fogott-e harcsát (optimista lin. keresés):
   1. Nincs verseny
   2. 1 horgász, 1 verseny, nem fogott
   3. 1 horgász, 1 verseny, fogott
   4. 1 horgász, több verseny, egyiken sem fogott
   5. 1 horgász, több verseny, nem mindenhol fogott
   6. 1 horgász, több verseny, mindenhol fogott