***8. táblás gyakorlat feladatai***

Egy szekvenciális inputfájl vadászok eredményeit (vadász neve, vadászat dátuma, zsákmány fajtája, zsákmány súlya) tárolja a vadászok neve, azon belül a vadászat dátuma szerint rendezetten. Igaz-e, hogy minden vadász valamelyik vadászatán lőtt medvét, és hány olyan vadász van, aki minden vadászatán lőtt nyulat és a zsákmányainak összsúlya meghaladja a 250 kilogrammot?

Minta bemenet:

A képen asztal látható

Automatikusan generált leírás

*Specifikáció*:

*A* = ( x:infile(Trófea), l: 𝕃, c:ℕ )

Trófea = rec(név:𝕊, dátum:𝕊, fajta:𝕊, súly:ℕ)

*Ef* = ( x=x0 ∧ x↗(név,dátum))

*Ötlet:*

Soroljuk fel, hogy az egyes vadászok lőttek-e medvét, lőttek-e minden vadászatukon nyulat, és mennyi a zsákmányaiknak az összsúlya.

*Új specifikáció*:

*A* = ( t:enor(Vadász), l: 𝕃, c:ℕ )

Vadász = rec(medve:𝕃, nyúl:𝕃, összsúly:ℕ)

*Ef* = ( t=t0 )

*Uf* = ( l = e∊t0e.medve ∧ c = **∑** e∊t01 )  
 e.nyúl ∧ e.összsúly>250

*Összeéselés és Számlálás közös felsorolón*

t:enor(E) ~ t:enor(Vadász)

f(e) ~ e.medve

H, +, 0 ~ 𝕃, ∧, igaz

felt(e) ~ e.nyúl ∧ e.összsúly>250

*Algoritmus*:

|  |  |  |
| --- | --- | --- |
| l, c := igaz, 0  t.first() | | |
| ¬t.end() | | |
|  | l := l ∧ t.current().medve | |
|  | t.current().nyúl ∧ t.current().összsúly>250 | |
|  | c := c + 1 | ‒ |
|  | t.next() | |

*Felsoroló*:

*Ötlet:* A t:enor(Vadász) felsoroló megvalósításához szükség lesz egy olyan y:enor(Összesítés) felsorolóra, amely minden vadásznak az egyes vadászatokon elért összesített eredményeit sorolja fe a vadász neve és a vadászat dátuma szerint növekedően rendezetten: lőtt-e medvét, lőtt-e nyulat, és mennyi volt ott a zsákmányai súlyának összege.

t:enor(Vadász) Vadász = rec(medve:𝕃, nyúl:𝕃, összsúly:ℕ)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vadász\* | first() | next() | current() : Vadász | end() : 𝕃 |
| y : enor(Összesítés)  akt : Vadász  vége : 𝕃 | y.first()  next() | lásd külön | **return** akt | **return** vége |

Összesítés = rec(név: 𝕊, dátum:𝕊, medve:𝕃, nyúl:𝕃, össz:ℕ)

*next() művelet*

*A* = (y:enor(Összesítés), akt:Vadász, vége:𝕃 )

*Ef* = (y = y’ ∧ (y folyamatban van ∨ y befejeződött) ∧ y↗név)

*Uf* = ( vége = y’.end() ∧ (¬vége → (akt.medve, y) e.medve

∧ (akt.nyúl, y) e.nyúl

∧ (akt.összsúly, y) ) )

*Három összegzés (közös felsorolóval)*

t:enor(E) ~ y:enor(Összesítés)

first() nélkül

amíg: y.current().név=akt.név

f(e) ~ (y.current().medve,  
 y.current().nyúl,   
 y.current().össz )

s ~ akt

H, +, 0 ~ ( 𝕃×𝕃×ℕ), (∨,∧,+), (hamis,igaz,0)

|  |  |  |
| --- | --- | --- |
| vége := y.end() | | |
| ¬vége | | |
| név, akt.medve, akt.nyúl, akt.összsúly :=   y.current().név, hamis, igaz, 0 | | ‒ |
| ¬y.end() ∧ y.current().név=név | |
|  | akt.medve := akt.medve ∨ y.current().medve |
| akt akt.nyúl := akt.nyúl ∧ y.current().nyúl |
| akt.összsúly := akt.összsúly + y.current().össz |
| y.next() |

*Felsoroló*:

y:enor(Összesítés) Összesítés = rec(név: 𝕊, dátum:𝕊, medve:𝕃, nyúl:𝕃, össz:ℕ)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Összesítés\* | first() | next() | current() : Összesítés | end() : 𝕃 |
| x : infile(Trófea)  dx : Trófea  sx : Status  akt : Összesítés  vége : 𝕃 | sx,dx,x:read next() | lásd külön | **return** akt | **return** vége |

Trófea = rec(név:𝕊, dátum:𝕊, fajta:𝕊, súly:ℕ)

*next() művelet*

*A* = (x:infile(Trófea), dx:Trófea, sx:Status, akt:Összesítés, vége:𝕃 )

*Ef* = (x = x’ ∧ x↗(név,dátum) ∧ dx = dx’ ∧ sx = sx’)

*Uf* = ( vége = (sx’=abnorm) ∧ (¬vége → akt.név=dx’.név ∧ akt.dátum=dx’.dátum ∧

dx,név=akt.név ∧ dx,dátum=akt.dátum

∧ (akt.medve, (sx,dx,x)) = dx∊(dx’,x’) dx.fajta=”medve”∧

dx,név=akt.név ∧ dx,dátum=akt.dátum

∧ (akt.nyúl, (sx,dx,x)) = dx∊(dx’,x’) dx.fajta=”nyúl”∧

dx,név=akt.név ∧ dx,dátum=akt.dátum

∧ (akt.össz, (sx,dx,x)) = **∑** dx∊(dx’,x’) dx.súly ) )

*Három összegzés (két vagyolás és egy összeadás)*

*közös felsorolóval*

t:enor(E) ~ x:infile(Trófea) sx,dx,x:read,   
 first() nélkül,

amíg: dx.név=akt.név ∧ dx.dátum=akt.dátum

f(e) ~ (dx.fajta =”medve”,

dx.fajta =”nyúl”,

dx.súly)

s ~ akt.medve, akt.nyúl, akt.össz

H, +, 0 ~ (𝕃×𝕃×ℕ), (∨,∨,+), (hamis,hamis,0)

|  |  |  |  |
| --- | --- | --- | --- |
| vége := sx=abnorm | | | |
| ¬vége | | | |
| akt := (dx.név, dx.dátum, hamis, hamis, 0) | | ‒ |
| sx=norm ∧ dx.név=akt.név  ∧ dx.dátum=akt.dátum | |
|  | akt.akt.medve := akt.medve ∨ dx.fajta=”medve” |
| akt.nyúl := akt.nyúl ∨ dx.fajta=”nyúl” |
| akt.össz := akt.össz + dx.súly |
| sx, dx, x:read |

**Mátrixok feldolgozása:**

**Nevezetes bejárások (felsorolók)**

A mátrixok feldolgozhatók sor-, vagy oszlop folytonosan. Ezt a két bejárást tekinthetjük nevezetes felsorolásnak.

Legyen A egy n × m-es valós mátrix (n sora és m oszlopa van). Sorfolytonos felsorolója indexpárokat állít elő, amelyekkel a mátrix elemek elérhetők:

Sorfolytonos felsoroló:

t:enor(ℕ⨯ℕ) sorfolytonos felsorolása az indexpároknak

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (ℕ×ℕ)\* | first() | next() | current() : (ℕ×ℕ) | end() : 𝕃 |
| i: ℕ  j: ℕ  n,m : ℕ | i,j:=1,1 | **if** j<m **then**  j:=j+1  **else**  i:=i+1  j:=1  **endif** | **return** (i,j) | **return** (i>n) |

**Feladat:** Határozzuk a mátrix (egyik) legnagyobb elemét:

*A* =(X: ℝ nxm, ind,jnd: ℕ)

*Ef* =(X=X’ ∧ n>0 ∧ m>0)

*Uf* =(ind,jnd=)

Maximum kiválasztás

t:enor(E) ~ indexpárok sorfolytonos felsorolója

f(e) ~ X[i,j]

H,> ~ ℝ, >

*Algoritmus:*

|  |  |  |  |
| --- | --- | --- | --- |
| t.first() | | |  |
| (ind,jnd):=t.current() | | |  |
| t.next() | | |  |
| ¬t.end() | | |  |
|  | (i, j) := t.current() | | i, j: ℕ |
| X[i,j]>X[ind,jnd] | |  |
| (ind,jnd):=(i,j) | - |  |
| t.next() | |  |

*Szokásosabb formában:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (ind,jnd):=(1,1) | | | |  |
| i = 1 .. n | | | | i, j: ℕ |
|  |  | j = 1 .. n | |  |
| X[i,j]>X[ind,jnd] | |  |
| (ind,jnd):=(i,j) | ― |  |

**Feladat:** Mennyi egy négyzetes mátrix sakktábla szerinti fehér mezőire eső elemeinek az összege?

I. megoldás: nevezetes felsorolóval

*Specifikáció*:

*A* = ( a:ℝn×n, s:ℝ )  
*Ef* = ( a=a’)  
*Uf* = ( Ef ∧ s = )

2⏐i +j

*Összegzés*

t:enor(E) ~ indexpárok klasszikus felsorolója

f(e) ~ a[i,j] ha 2⏐i+j

H, +, 0 ~ ℝ, +, 0.0

*Algoritmus:*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| s := 0.0 | | | | |  | |
| i = 1 .. n | | | | | i, j: ℕ | |
|  |  | j = 1 .. n | |  | |
| 2⏐i + j | |  | |
| s := s + a[i,j] | ― |  | |

**Mátrix, egyedi felsoroló készítés**

*Specifikáció*:

*A* = ( a:ℝn×n, , t:enor(ℕ⨯ℕ), s:ℝ)  
*Ef* = ( a=a’ ∧ t=t’)  
*Uf* = (s = **∑**(i,j)∊t’ a’[i,j] )

*Összegzés*

t:enor(E) ~ indexpárok sakktáblás felsorolója

f(e) ~ a[i,j]

H, +, 0 ~ ℝ, +, 0.0

*Algoritmus*:

|  |  |  |
| --- | --- | --- |
| s := 0.0  t.first() | |  |
| ¬t.end() | |  |
|  | (i, j) := t.current() | i, j: ℕ |
| s := s + a[i,j] |  |
| t.next() |  |

*Felsoroló*

t:enor(ℕ⨯ℕ) sakktáblaszerű felsorolása az indexpároknak

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (ℕ⨯ℕ)\* | first() | next() | current() : ℕ⨯ℕ | end() : 𝕃 |
| n : ℕ  i, j : ℕ | i, j := 1, 1 | **if** j ≤ n-2 **then**  j:=j+2  **else**  i := i+1  **if** i páros  **then** j := 2  **else** j := 1  **endif**  **endif** | **return** (i, j) | **return** (i>n) |