***6. táblás gyakorlat feladatai***

1. Adott egy egész számokat tartalmazó szekvenciális inputfájl.
2. Hány páros szám követi az első negatív számot?

*Specifikáció*:

*A* = ( x:infile(ℤ), db:ℕ )
*Ef* = ( x=x0 )
*Uf* *1* = ( (l, elem, (st’,e’,x’)) = **SEARCH**e∊x0(e<0) ∧
 ∧ db = $\sum\_{e\in (e^{'},x^{'})}^{}1$)
 e páros// másképp:

*Uf 2* = ( (e’, (st’,e’,x’)) =
 **SELECT**st,e∊x0(st=abnorm ∨ e<0) ∧
 ∧ db = $\sum\_{e\in x^{'}}^{}1$)
 e páros

*Ez utóbbi Uf szerinti megoldás:*

*Kiválasztás*

t:enor(E) ~ x:infile(ℤ) (st,e,x:read)

felt(e) ~ st=abnorm∨ e<0

c ~ db

*Számlálás*

t:enor(E) ~ x:infile(ℤ) „folytatása”

felt(e) ~ e páros

c ~ db

*Algoritmus*:

|  |  |
| --- | --- |
| st, e, x:read | e:ℤ  |
| st=norm ∧ e≥0 | st:Status |
|  | st, e, x:read |  |
| db := 0 st, e, x:read |
| st=norm |
|  | e páros |
|  | db := db+1 | ― |
|  | st, e, x:read |

*Megjegyzések*:

* Az első Uf szerinti megoldás egy lineáris keresés, majd egy számlálás szekvenciája. Lineáris keresés esetén az „elem” változóban lesz az első negatív szám (ha van), e’-ben a ciklusban már előreolvasott következő szám, és x’ a fájl maradék része. A számlálás tételnek ezt az előre olvasott e’-vel jelölt adatot, és a fájl maradék részét kell feldolgoznia. Ha nincs a fájlban negatív szám, akkor az olvasás eléri a fájl végét, st’=abnorm miatt a számlálás ciklusa nem hajtódik végre. Mivel a számlálás tételnek nem előfeltétele, hogy a felsoroló nem lehet üres, hibát nem okoz.
* A második megoldásban kiválasztást és számlálást alkalmazunk szekvenciában. Mivel nem garantálható, hogy a fájl tartalmaz negatív számot, a kiválasztás az st státusz változó értékét is figyeli. A kiválasztásnak két eredménye van: a keresett elem, amelyet a specifikáció most e’-vel jelöl (ennek tárolására nincs külön elem változó, mert erre az eredményre nincs szükség, és az e változó úgyis tárolja); valamint a szekvenciális inputfájl felsorolásának aktuális állapota, amelyet az st, e, és x változók pillanatnyi értékei (ezeket jelöli az st’, e’, x’) együttesen jellemez.
* A specifikáció képes a majdani megoldó program három különböző állapotát is megmutatni: például az x kezdőértékét az indulásra kész x0 jelzi; erre alkalmazzuk a kiválasztást (SELECT), amely következtében az x változó (közbeeső) értéke x’ lesz; erre alkalmazzuk a számlálást, amely után az x végértéke az üres sorozat lesz (ezt explicit módon nem írja az utófeltétel).
1. Hány páros szám van az első negatív számot megelőzően, és hány azt követően?

*Specifikáció*:

*A* = ( x:infile(ℤ), dbe, dbu:ℕ )
*Ef* = ( x=x0 )

*Uf* = ( (dbe, (st’,e’,x’)) = $\sum\_{e\in x\_{0}}^{e\geq 0}1 $
 e páros

 ∧ dbu = $\sum\_{e\in x^{'}}^{}1$)
 e páros

*Számlálás, feltétel fennállásáig*

t:enor(E) ~ x:infile(ℤ) (st,e,x:read)

 amíg: e≥0

felt(e) ~ e páros

c ~ dbe

*Számlálás*

t:enor(E) ~ x:infile(ℤ) (st,e,x:read)

 first() helyett next()

felt(e) ~ e páros

c ~ dbu

*Algoritmus*:

|  |  |
| --- | --- |
| dbe := 0 st, e, x:read | e:ℤ st:Status |
| st=norm ∧ e≥0 |  |
|  | e páros |  |
|  | dbe := dbe +1 | ― |  |
|  | st, e, x:read |  |
| dbu := 0 st, e, x:read |
| st=norm |
|  | e páros |
|  | dbu := dbu+1 | ― |
|  | st, e, x:read |

*Megjegyzés*: A második számlálás algoritmusának inicializáló részében szereplő read nem a first(), hanem a next() művelet (csak szekvenciális inputfájloknál ezek ugyanazok), hiszen itt folytatni kell a szekvenciális inputfájl már folyamatban levő felsorolását.

1. Hány páros szám van az első negatív számot megelőzően, és hány azt követően azzal együtt?

*Specifikáció*:

*A* = ( x:infile(ℤ), dbe, dbu:ℕ )
*Ef* = ( x=x0 )

*Uf* = ( (dbe, (st’,e’,x’)) = $\sum\_{e\in x\_{0}}^{e\geq 0}1 $
 e páros

 ∧ dbu = $\sum\_{e\in (e',x^{'})}^{}1$)
 e páros

*Számlálás, feltétel fennállásáig*

t:enor(E) ~ x:infile(ℤ) (st,e,x:read)

 amíg: e≥0

felt(e) ~ e páros

c ~ dbe

*Számlálás*

t:enor(E) ~ x:infile(ℤ) (st,e,x:read)

 first() nélkül

felt(e) ~ e páros

c ~ dbu

*Algoritmus*:

|  |  |
| --- | --- |
| dbe := 0 st, e, x:read | e:ℤ st:Status |
| st=norm ∧ e≥0 |  |
|  | e páros |  |
|  | dbe := dbe +1 | ― |  |
|  | st, e, x:read |  |
| dbu := 0 |
| st=norm |
|  | e páros |
|  | dbu := dbu+1 | ― |
|  | st, e, x:read |

*Megjegyzés*: A második számlálás algoritmusának inicializáló részében nem szerepel read (sem first(), sem next() nincsen), hiszen nincs szükség előre olvasására ahhoz, hogy az első negatív számot elérjük a fájlból, hiszen azt már előzőleg kiolvastuk.

1. Egy szekvenciális inputfájlban napi átlaghőmérsékleteket tárolunk. Mennyi az első fagypont alatti értéket megelőző napok hőmérsékleteinek átlaga, és mennyi a többi nap (első fagypont alatti értékkel együtt vett) átlaga?

*Specifikáció*:

*A* = ( x:infile(ℝ), a1, a2:ℝ )
*Ef* = ( x=x0  ∧ ∣x∣≥2 ∧ x[1]≥0 ∧ ∃i∊[2 ..∣x∣]: x[i]<0)
*Uf* = ( (s1, (st’,e’,x’)) = $\sum\_{e\in x\_{0}}^{e\geq 0}e$ ∧
 (db1, (st’,e’,x’)) = $\sum\_{e\in x\_{0}}^{e\geq 0}1 $∧
 ∧ s2 = $\sum\_{e\in (e^{'},x^{'})}^{}e$ ∧ db2 = $\sum\_{e\in (e^{'},x^{'})}^{}1$ ∧
 ∧ a1 = s1/db1 ∧ a2 = s2/db2 )

*Két összegzés, feltétel fennállásáig tartanak*

t:enor(E) ~ x:infile(ℝ) (st,e,x:read)

 amíg: e≥0

f(e) ~ e, 1

s ~ s1, db1

 H, +, 0 ~ (ℝ +, 0.0), (ℕ +, 0)

*Két összegzés*

t:enor(E) ~ x:infile(ℝ) (st,e,x:read)

 first() nélkül

f(e) ~ e, 1

s ~ s2, db2

H, +, 0 ~ (ℝ +, 0.0), (ℕ +, 0)

*Algoritmus*:

|  |  |
| --- | --- |
| s1, db1 := 0.0, 0st,e,x:read | e, s1:ℝ, c1:ℕ st:Status |
| st=norm ∧ e≥0 |  |
|  | s1, db1 := s1+e, db1 +1 |  |
|  | st,e,x:read |  |
| s2, db2 := 0.0, 0 | s2:ℝ, c2:ℕ |
| st=norm |  |
|  | s2, db2 := s2+e, db2+1 |  |
|  | st,e,x:read |  |
| a1, a2 := s1/db1, s2/db2 |  |

*Megjegyzés*: A specifikáció két utolsó összegzésének jelölésében az e∊(e’,x’) felsorolás arra utal, hogy nincs szükség a first() műveletre, mert eleve rendelkezünk már az első elemmel.

1. Egy szekvenciális inputfájlban tároljuk a Föld felszínének egy vonalán – adott távolságokként – mért tengerszint feletti magasságértékeket. Milyen magas a legmagasabb horpadás?

*Specifikáció*:

*A* = ( x:infile(ℝ), l:𝕃, max:ℝ )
*Ef* = ( x=x0)

*Új specifikáció*:

*A* = ( t:enor(ℝ⨯ℝ⨯ℝ), l:𝕃, max:ℝ )
*Ef* = ( t=t0)
*Uf* = ( (l, max)= **MAX**(e,a,k)∊t0 a )
 e>a∧k>a

*Ötlet:*

Kellene egy olyan felsoroló, amelyik mindig egymás után mért három értéket mutatja.

*Felt. max. ker.*

t:enor(E) ~ t:enor(ℝ⨯ℝ⨯ℝ),
 ahol (e, a, k) := t.current()

f(e) ~ a

felt(e) ~ e>a ∧ k>a

H, > ~ ℤ, >

*Algoritmus*:

|  |
| --- |
| l := hamist.first() |
|  ¬t.end() |
|  | (e, a, k) := t.current() | e, a, k: ℝ |
| ¬(e>a∧k>a) | e>a∧k>a ∧ l | e>a∧k>a ∧ ¬l |  |
| ― | a > max | l, max := igaz, a |  |
| max := a | ― |
| t.next() |

*Felsoroló*

t:enor(ℝ⨯ℝ⨯ℝ)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (ℝ⨯ℝ⨯ℝ)\* | first() | next() | current() : ℝ⨯ℝ⨯ℝ | end() : 𝕃 |
| x:infile(ℝ)e, a, k: ℝst : Status | st, e, x:readst, a, x:readst, k, x:read | e := aa := kst, k, x:read | **return** (e, a, k) | **return** st=abnorm |

1. Gyűjtsük ki egy szekvenciális inputfájlban rendezve tárolt egész számok közül azt, hogy melyik számból hány darab található!

*Specifikáció*:

*A* = ( x:infile(ℤ), y:outfile(Össz) )
 Össz = rec(szám:ℤ, db:ℕ)
*Ef* = ( x = x0 ∧ x↗ )
 (x↗ azt jelzi, hogy az
 x növekedően rendezett)

*Új Specifikáció*:

*A* = ( t:enor(Össz), y:outfile(Össz) )

*Ef* = ( t = t0 )

*Uf* = ( y = t0 ) = ( y = ⊕e∊t0 <e> )

*Összegzés (másolás)*

f(e) ~ <e>

s ~ y

H, +, 0 ~ Össz\*, ⊕, <>

*Ötlet:*

Ha lenne egy olyan felsorolónk, amelyik az eredményt, az összesítéseket tartalmazó rekordokat tudná felsorolni, akkor elég lenne ezeket az output fájlba átmásolni.

*Algoritmus*:

|  |
| --- |
| y := <>t.first() |
| ¬t.end() |
|  | y : write(t.current()) |
|  | t.next() |

*Felsoroló:*

t:enor(Össz) Össz = rec(szám:ℤ, db:ℕ)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Össz\* | first() | next() | current() : Össz | end() :𝕃 |
| x : infile(ℤ)dx : ℤsx : Statusakt : Összvége : 𝕃 | sx,dx,x:readnext() | lásd külön | **return** akt | **return** vége |

*next() művelet*

*A* = (x:infile(ℤ), dx:ℤ, sx:Status, akt:Össz, vége:𝕃)

*Ef* = (x = x’ ∧ x↗ ∧ dx = dx’ ∧ sx = sx’ ) dx = akt.szám

*Uf* = (vége = (sx’=abnorm) ∧ ( ¬vége → akt.szám=dx’ ∧ (akt.db, (sx,dx,x)) = **∑** dx∊(dx’,x’) 1 ) )

*Megjegyzés*: Az összegzésnek két eredménye van: a darabszám (akt.db); és a felsoroló aktuális állapota, amelyet az sx,dx,x változók értékei írnak le a next() művelet végén.

*Összegzés*

t:enor(E) ~ x:infile(ℤ) (sx,dx,x:read)
 first() nélkül, amíg: dx=akt.szám

f(e) ~ 1

s ~ akt.db

H, +, 0 ~ ℕ, +, 0

|  |
| --- |
| vége := sx=abnorm |
| ¬vége |
| akt.szám, akt.db := dx, 0 |  |
| sx=norm ∧ dx=akt.szám | ‒ |
|  | akt.db := akt.db + 1 |  |
|  | sx,dx,x:read |  |

1. Számoljuk meg egy karakterekből álló szekvenciális inputfájlban a szavakat úgy, hogy a *12* betűnél hosszabb szavakat duplán vesszük figyelembe! (Egy szót szóközök vagy a fájl vége határol.)

*Specifikáció*:

*A* = ( f:infile(𝕂), c:ℕ )
*Ef* = ( f = f0 )

*Új specifikáció*:

*A* = ( t:enor(ℕ), c:ℕ )

*Ef* = ( t = t0 )

*Uf* = ( c = **∑** e∊t0 {2, ha e>12; 1, ha e≤12} )

*Összegzés*

f(e) ~ {2, ha e>12; 1, ha e≤12}

s ~ c

H, +, 0 ~ ℕ, +, 0

*Felsoroló:Ötlet:*

Soroljuk fel csak a szavak hosszait, és ebből az eredményt már könnyen kiszámolhatjuk.

*Algoritmus*:

|  |
| --- |
| c := 0t.first() |
| ¬t.end() |
|  | t.current()>12 |
|  | c := c + 2 | c := c + 1 |
|  | t.next() |

t:enor(ℕ)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ℕ\* | first() | next() | current() : ℕ | end() : 𝕃 |
| f : infile(𝕂)ch : 𝕂st : Statusakt : ℕvége : 𝕃 | st,ch,f:readnext() | lásd külön | **return** akt | **return** vége |

*next() művelet*

*A* = (f:infile(𝕂), ch:𝕂, st:Status, akt:ℕ, vége:𝕃 )

*Ef* = (f = f’ ∧ ch = ch’ ∧ st = st’ )

*Uf* = ( (ch”,(st”,ch”,f”))=SELECTch∊(ch’,f’) (st=abnorm ∨ ch≠’ ’) ∧ ch ≠ ’ ’

 vége=(st”=abnorm) ∧ ( ¬vége → (akt, (st,ch,f)) = **∑** ch∊(ch”,f”) 1 ) )

*Kiválasztás*

t:enor(E) ~ f:infile(𝕂) (st,ch,f:read)

 first() nélkül

felt(e) ~ st=abnorm ∨ ch≠’ ’

*Összegzés (megszámolás)*

t:enor(E) ~ f:infile(𝕂) (st,ch,f:read)
 first() nélkül, amíg: ch≠’ ’

f(e) ~ 1

s ~ akt

H, +, 0 ~ ℕ, +, 0 *Algoritmus*:

|  |
| --- |
| st=norm ∧ ch=’ ’ |
|  | st,ch,f:read |
| vége := st=abnorm |
| ¬vége |
| akt := 0 |  |
| st=norm ∧ ch≠’ ’ | ‒ |
|  | akt := akt+1 |  |
|  | st,ch,f:read |  |

*Megjegyzés*: Konkrét megvalósításban nemcsak szóközök, hanem mindenféle szeparátor (sorvége-jel, tabulátor-jel, stb.) is lehet a szavak között.