OEP

6. táblás gyakorlat

felsorolós programozási tételek II.

egyedi felsoroló készítése

Tartalom

- Azonos felsorolóra támaszkodótételek szekvenciában
- Párhuzamosan futó tételek egy ciklusba vonása
- Egyszerű egyedi felsoroló
- Tételt használó egyedi felsoroló

Azonos felsorolót használó tételek szekvenciába fűzése

- Adott egy egészekből álló szekvenciális input fájl.
- (b) Hány páros szám követi az első negatív számot?
- ⟨ 1 3 **4** -1 2 -2 1 4 3⟩
- ⟨ 1 3 5 -1 3 1 5⟩
- ⟨ -1 ⟩
- ⟨ 1 3 **4 2** 7 **2**⟩
- ⟨ -1⟩
- ⟨ ⟩

- Meg kell találni a fájlban az első negatív számot, azt követően indulhat a számlálás, a fájl végéig.
- Lehet, hogy nem lesz negatív szám a fájlban.
- Milyen tétellel dolgozzuk fel a fájl első részét?
  - Lineáris keresés
  - Kiválasztás

Megoldás lineáris keresést használva

- *Specifikáció*:
- A = (x:infile($\mathbb{Z}$), db:$\mathbb{N}$)
- Ef = ($x=x_0$)
- Uf = ((l, elem, (st',e',x')) = **SEARCH**$_{e \in x_0}(e<0) \wedge$
  $\wedge$ db = $\sum_{e \in (e', x')}$ 1)
  e páros

A tétel ciklusa eggyel tovább olvas!

l := hamis; t.first()
$\neg l \wedge \neg t.end()$
elem := t.current()
l := felt(elem)
t.next()

„elem”-ben lesz az első negatív, ha találtunk

A folytatásnak fel kell dolgoznia az előre olvasott elemet, és a fájl maradék részét!

⟨ 1 3 4 -1 2 -2 1 4 3 ⟩

l=true; elem=-1

e'=2

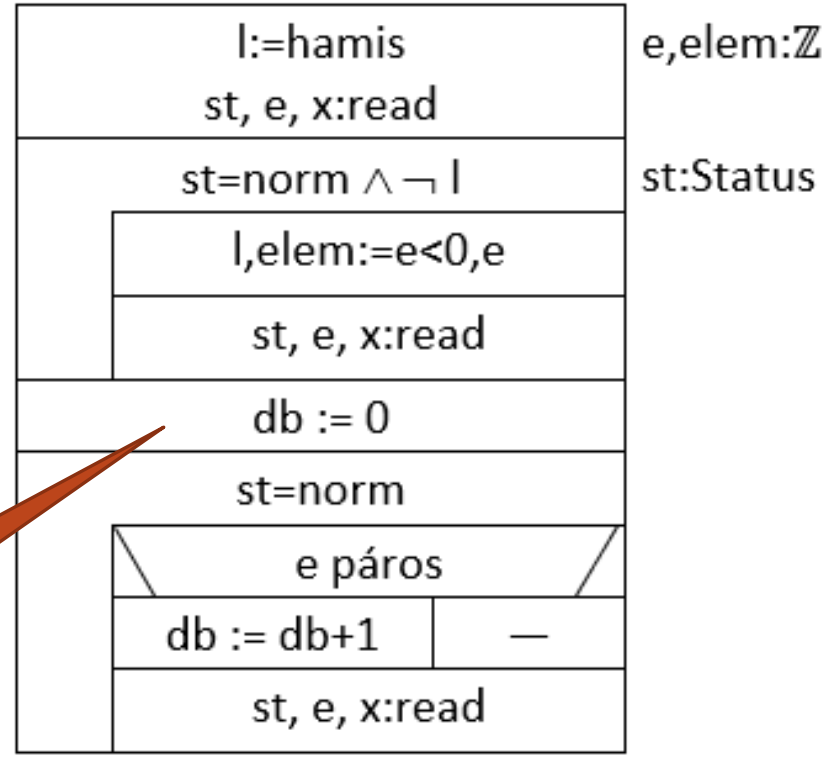
x' – a fájl hátralévő része

- *Specifikáció*:
- A = (x:infile($\mathbb{Z}$), db:$\mathbb{N}$)
- Ef = ($x=x_0$)
- Uf = ((l, elem, (st',e',x')) = **SEARCH**$_{e \in x_0}(e<0) \wedge$
  $\wedge$ db = $\sum_{e \in (e', x')} 1$)
  e páros

- *Visszavezetés:*
- *Lineáris keresés*
- t:enor(E) ~ x:infile($\mathbb{Z}$) (st,e,x:read)
- felt(e) ~ e<0
- *Számlálás*
- t:enor(E) ~ e, x:infile($\mathbb{Z}$) „folytatása”
- felt(e) ~ e páros
- c ~ db

Fontos!
Számlálásban nincs előreolvasás!
first() elmarad

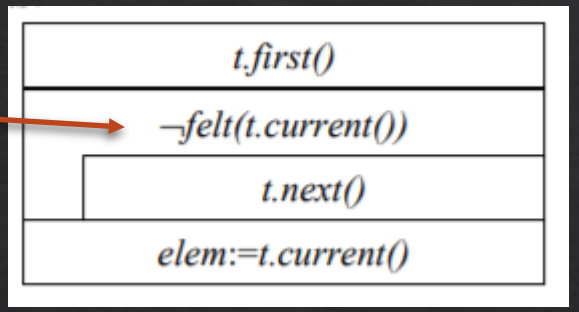
Algoritmus:



Megoldás kiválasztás tételt használva

- *Specifikáció*:
- A = (x:infile($\mathbb{Z}$), db:$\mathbb{N}$)
- Ef = ($x=x_0$)
- Uf = ((e', (st',e',x')) =
  SELECT$_{st,e \in x_0}$(st=abnorm $\vee$ e<0) $\wedge$
  $\wedge$ db = $\sum_{\substack{e \in x' \\ \text{e páros}}} 1$)

A tétel ciklusa leáll, amikor negatívat olvasott



„e"-ben lesz az első negatív, ha találtunk

A folytatás feldolgozza a fájl maradék részét.

⟨ 1 3 4 -1 2 -2 1 4 3 ⟩

e=-1

x' – a fájl hátralévő része

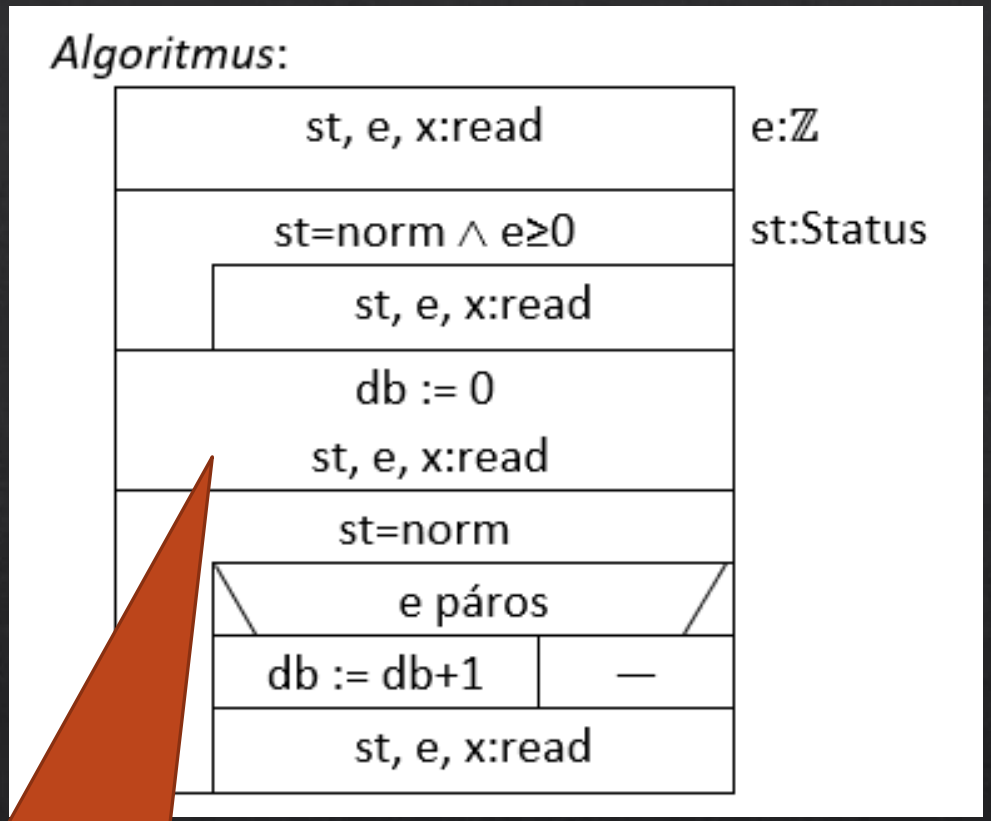
- *Specifikáció*:
- A = (x:infile($\mathbb{Z}$), db:$\mathbb{N}$)
- *Ef* = ($x=x_0$)
- *Uf* = ((e', (st',e',x')) =

  $\mathbf{SELECT}_{st,e \in x_0}(st=abnorm \vee e<0) \wedge$

  $\wedge \quad db = \sum_{\substack{e \in x' \\ e \text{ páros}}} 1)$

- *Visszavezetés:*
- *Kiválasztás*
- t:enor(E) ~ x:infile($\mathbb{Z}$) (st,e,x:read)
- felt(e) ~ st=abnorm $\vee$ e<0
- *Számlálás*
- t:enor(E) ~ x:infile($\mathbb{Z}$) „folytatása” first() helyett next()
- felt(e) ~ e páros
- c ~ db

Algoritmus:



Fontos!
Tovább kell olvasni, hogy az első negatív utáni számmal kezdhessen a számlálás.

Azonos felsorolót használó tételek szekvenciába fűzése

- Adott egy egészekből álló szekvenciális input fájl.
- (c) Hány páros szám van az első negatív számot megelőzően, és hány azt követően?
- ⟨ 1 3 **4** -1 2 -2 1 4 3⟩
- ⟨ 1 3 5 -1 3 1 5⟩
- ⟨ -1 ⟩
- ⟨ 1 3 **4 2** 7 **2**⟩
- ⟨ -1⟩
- ⟨ ⟩

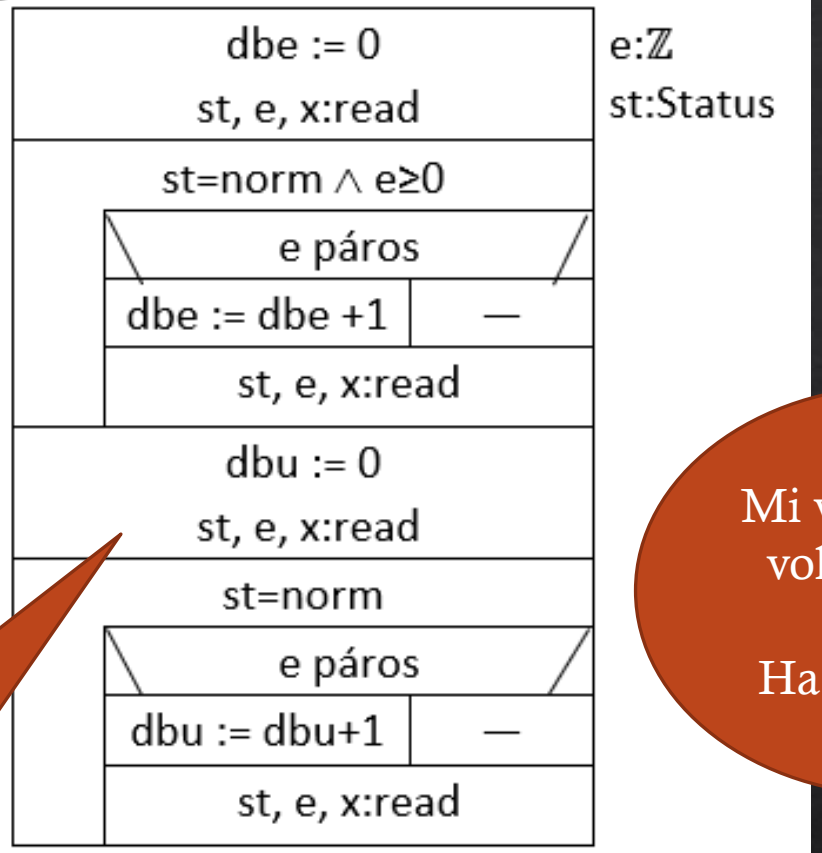
- A fájlban lévő első negatív elem a „vízválasztó”.
- Előtte lévő és utána következő fájl részeket kell tétellel feldolgozni.
- Első lépés: feltétel fennállásáig tartó számlálás.
- Negatív elemet „átlépjük”.
- Utána a fájl végéig egy újabb számlálás.

- *Specifikáció*:
- A = (x:infile($\mathbb{Z}$), dbe, dbu:$\mathbb{N}$)
  Ef = ($x=x_0$)
- **KVÍZ /1**
- Uf = ((dbe, (st',e',x')) = $\sum_{\substack{e\geq 0 \\ e\in x_0 \\ e \text{ páros}}} 1$

  $\wedge$ dbu = $\sum_{\substack{e\in x' \\ e \text{ páros}}} 1$)

- *Számlálás, feltétel fennállásáig*
- t:enor(E) ~ x:infile($\mathbb{Z}$) (st,e,x:read)
  amíg: e≥0
- felt(e) ~ e páros
- c ~ dbe
- *Számlálás*
- t:enor(E) ~ x:infile($\mathbb{Z}$) (st,e,x:read)
  first() helyett next()
- felt(e) ~ e páros
- c ~ dbu

Algoritmus:



Mi van, ha nem volt negatív a fájlban?
Ha üres a fájl?

Tételek közötti átmenet: e-ben az első negatív van (ha van negatív szám a fájlban), így tovább kell olvasni.

Azonos felsorolót használó tételek szekvenciába fűzése

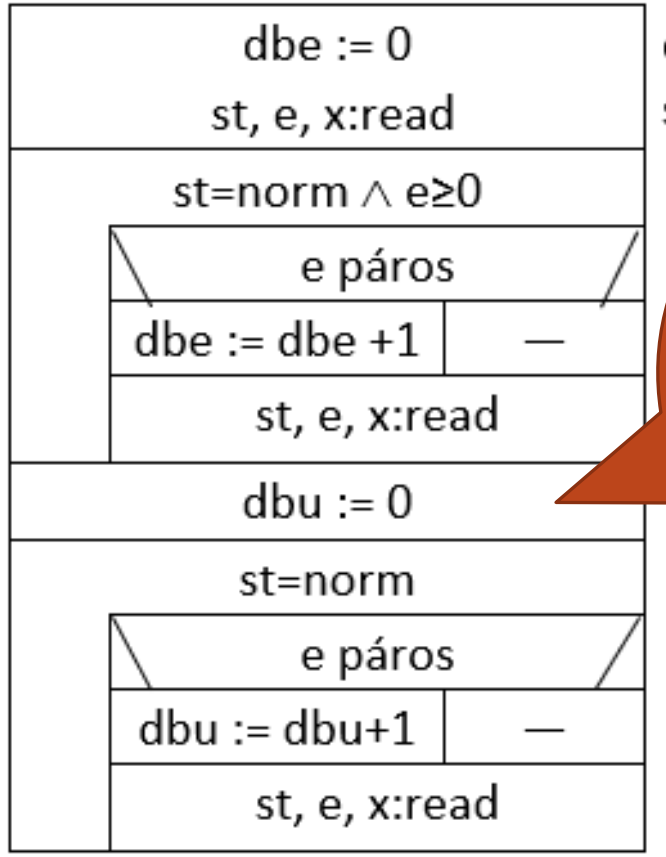
- Adott egy egészekből álló szekvenciális input fájl.
- (d) Hány páros szám van az első negatív számot megelőzően, és hány azt követően, azzal együtt?
- ⟨ 1 3 **4** -2 2 -2 1 4 3⟩
- ⟨ 1 3 5 -2 3 1 5⟩
- ⟨ -2 ⟩
- ⟨ 1 3 **4 2** 7 **2**⟩
- ⟨ -2⟩
- ⟨ ⟩

- A fájlban lévő első negatív elem előttiek tartoznak az első számláláshoz.
- Első lépés: feltétel fennállásáig tartó számlálás.
- Utána a fájl végéig egy újabb számlálás.
- Negatív elemet a második számlálás fel kell dolgozza!

- *Specifikáció*:
- A = (x:infile($\mathbb{Z}$), dbe, dbu:$\mathbb{N}$)
  Ef = ($x=x_0$)
- Uf = ((dbe, (st',e',x')) = $\sum_{\substack{e\in x_0 \\ \text{e páros}}}^{e\geq 0} 1$

  $\wedge$ dbu = $\sum_{\substack{e\in(e',x') \\ \text{e páros}}} 1$)
- *Számlálás, feltétel fennállásáig*
- t:enor(E) ~ x:infile($\mathbb{Z}$) (st,e,x:read)
  amíg: e≥0
- felt(e) ~ e páros
- c ~ dbe
- *Számlálás*
- t:enor(E) ~ x:infile($\mathbb{Z}$) (st,e,x:read)
  first() nélkül
- felt(e) ~ e páros
- c ~ dbu

Algoritmus:



Nincs first() / next(), mivel az „előre olvasott” elemet is fel kell dolgozni!

Nincs negatív/üres fájl?

Több tétel egy ciklusba vonása

- 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?
- Átlag esetén vigyázni kell a nullával való osztásra! Mik a helyes bemenetek?
- Ne kezdődjön negatív értékkel a fájl.
- Legyen benne negatív érték.
- ⟨ 1 3 **4** **2** 7 -1 2 -2 ….⟩
- ⟨ 5 -1 2⟩
- ⟨ 5 -1 ⟩

- *Specifikáció*:
- A = (x:infile($\mathbb{R}$), a1, a2:$\mathbb{R}$)
  Ef = ($x=x_0 \wedge |x|\geq 2 \wedge x[1]\geq 0 \wedge \exists i\in[2..|x|]: x[i]<0$)
  Uf = ((s1, (st',e',x')) = $\sum_{e\in x_0}^{e\geq 0} e \wedge$
  (db1, (st',e',x')) = $\sum_{e\in x_0}^{e\geq 0} 1 \wedge$
  $\wedge$ s2 = $\sum_{e\in(e',x')} e \wedge$ db2 = $\sum_{e\in(e',x')} 1 \wedge$
  $\wedge$ a1 = s1/db1 $\wedge$ a2 = s2/db2)

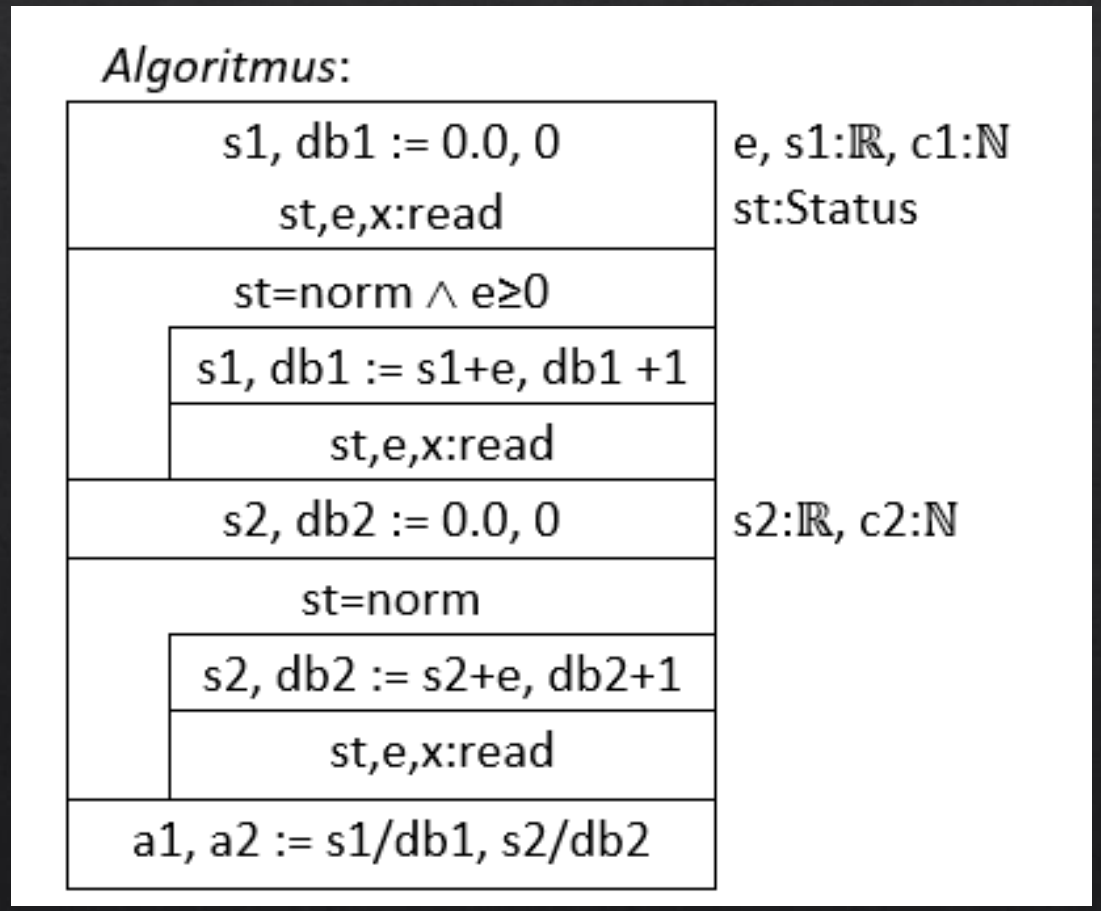
- *Két összegzés, feltétel fennállásáig tartanak*
- t:enor(E) ~ x:infile($\mathbb{R}$) (st,e,x:read)
- amíg: e≥0
- f(e) ~ e, 1
- s ~ s1, db1
- H, +, 0 ~ ($\mathbb{R}$ +, 0.0), ($\mathbb{N}$ +, 0)
-
- *Két összegzés*
- t:enor(E) ~ x:infile($\mathbb{R}$) (st,e,x:read)
  first() nélkül
- f(e) ~ e, 1
- s ~ s2, db2
- H, +, 0 ~ ($\mathbb{R}$ +, 0.0), ($\mathbb{N}$ +, 0)

Algoritmus:

s1, db1 := 0.0, 0 st,e,x:read	e, s1:$\mathbb{R}$, c1:$\mathbb{N}$ st:Status
st=norm $\wedge$ e≥0 s1, db1 := s1+e, db1 +1 st,e,x:read	
s2, db2 := 0.0, 0	s2:$\mathbb{R}$, c2:$\mathbb{N}$
st=norm s2, db2 := s2+e, db2+1 st,e,x:read	
a1, a2 := s1/db1, s2/db2	

Kvíz/2

Megjegyzések



- Azonos felsorolón futó feldolgozások összevonására általában jól használható az összegzés tétel.
- Lásd az első gyakorlatot: összegzéssel végzett eldöntések. („összevagyolás”/ „összeéselés”)
- Bizonyos esetekben a maximum kiválasztás is lehet összegzés (ha van extremális értékünk, az lehet a bal oldali null eleme a műveletnek).
- Ez a megoldás is ezen az ötleten alapszik, összegzést és számlálást von egybe egy speciális összegzés:
  a „+” művelet R × N elemekhez ad hozzá R × N elemet.

Egyedi felsoroló 1. feladat

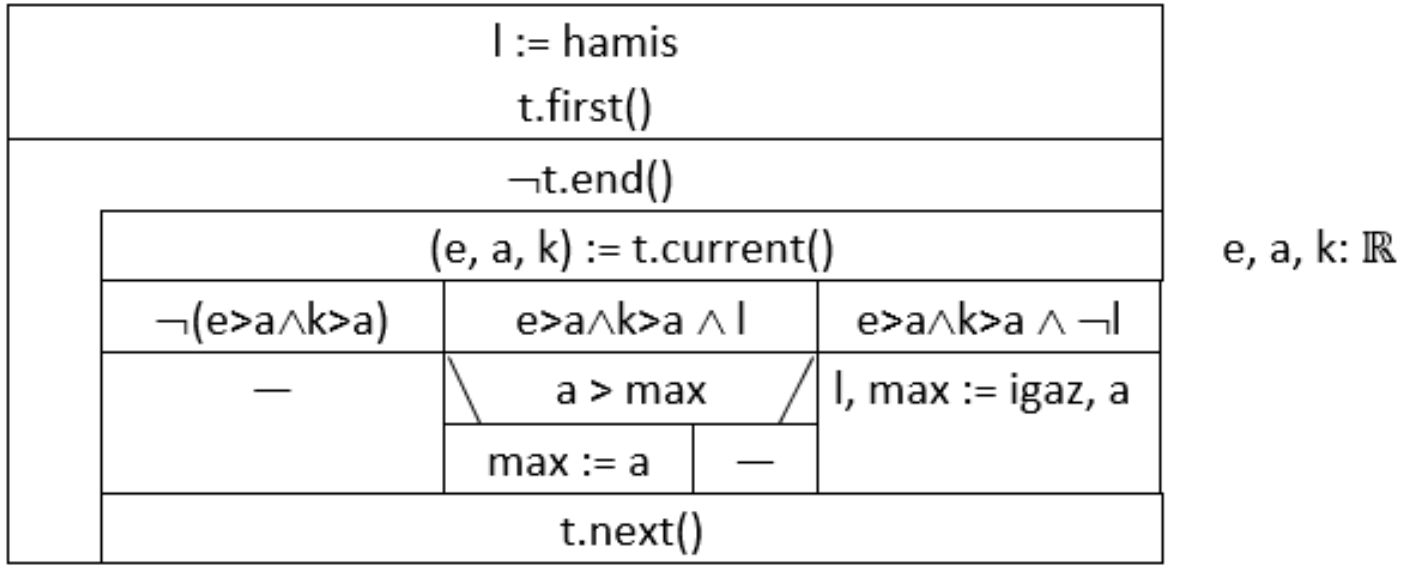
- Egy szekvenciális inputfájlban tároljuk a Föld felszínének egy vonalán – adott távolságokként – mért tengerszint feletti magasságértékeket. Milyen magas a legmagasabb horpadás?
- (e,a,k) három egymás utáni adat a fájlból „horpadás”, ha e>a és k>a.
- Bemenet: x:infile(R)
- A fájl nevezetes felsorolója (st,e,x) egy adatot ad a feldolgozáshoz, így ezen az állapottéren nem tudjuk tétellel megoldani a feladatot.
- Be kell vezetni egy olyan felsorolót, mely a fájl három egymás utáni adatát adja át $\Rightarrow$ egyedi felsorolót kell készíteni a feladathoz.
- Ha van egy ilyen felsorolónk, akkor a feltételes maximum keresés tétellel tudjuk megoldani a feladatot.

- Tegyük fel, hogy van egy ilyen felsorolónk, oldjuk meg ezzel a felsorolóval a kitűzött feladatot!

- A = (t:enor($\mathbb{R}\times\mathbb{R}\times\mathbb{R}$), l:$\mathbb{L}$, max:$\mathbb{R}$)
- Ef = ($t=t_0$)
- Uf = ((l, max) = $\mathbf{MAX}_{(e,a,k)\in t0}$ a)
  $e>a\wedge k>a$

- *Felt. max. ker.*
- t:enor(E) ~ t:enor($\mathbb{R}\times\mathbb{R}\times\mathbb{R}$),
  ahol (e, a, k) := t.current()
- f(e) ~ a
- felt(e) ~ $e>a \wedge k>a$
- H, > ~ $\mathbb{R}$, >

Algoritmus:



Készítsük el a felsorolót!

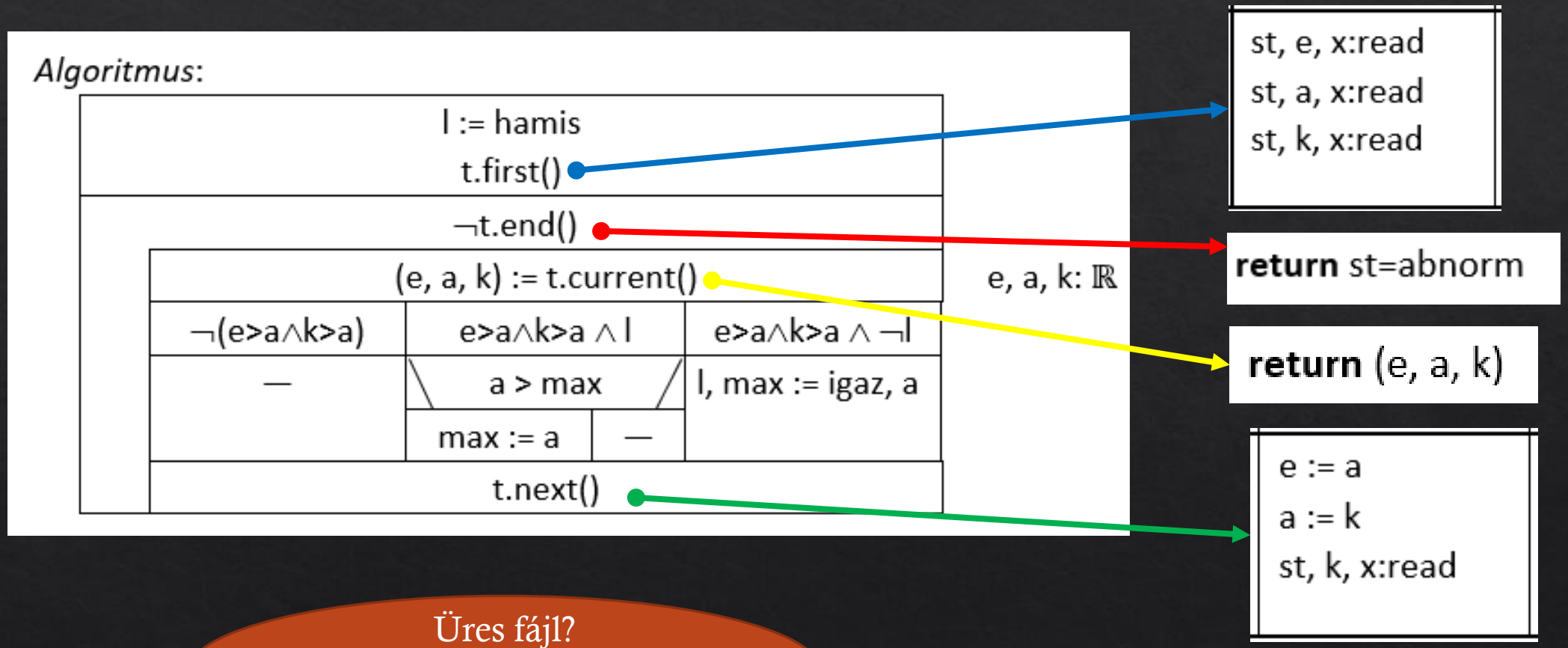
- A felsoroló fogja olvasni az input fájlt.

Felsoroló

t:enor($\mathbb{R}\times\mathbb{R}\times\mathbb{R}$)

$(\mathbb{R}\times\mathbb{R}\times\mathbb{R})^*$	first()	next()	current() : $\mathbb{R}\times\mathbb{R}\times\mathbb{R}$	end() : $\mathbb{L}$
x:infile($\mathbb{R}$) e, a, k: $\mathbb{R}$ st : Status	st, e, x:read st, a, x:read st, k, x:read	e := a a := k st, k, x:read	**return** (e, a, k)	**return** st=abnorm

Hogyan fog együttműködni a tétel és a felsoroló?



Üres fájl?
1 vagy 2 adatot tartalmazó fájl?

Egyedi felsoroló 2. feladat

- Gyűjtsük ki egy szekvenciális inputfájlban rendezve tárolt egész számokról azt, hogy melyik számból hány darab található.

Bemenet:

```
2 2 2
2
3 4 5 5
5 5
...
```

Kimenet:

Egész szám	Mennyi van belőle
2	4
3	1
4	1
5	4

Specifikáció készítés

- Próbáljuk meg specifikálni:

A = (x:infile($\mathbb{Z}$), y:outfile(Össz))
Össz = rec(szám:$\mathbb{Z}$, db:$\mathbb{N}$)
Ef = ($x = x' \wedge x\nearrow$)
(x↗ azt jelzi, hogy az
x növekedően rendezett)

UF ???

- Mi a probléma?
  Ezen az állapottéren nem tudjuk programozási tételre visszavezetni a feladatot.

Ötlet: Soroljuk fel az eredménybe szánt rekordokat, és másoljuk output fájlba őket.
(Sejthető: összegzés tételre kell majd visszavezetni.)

- Megoldás: egy olyan felsoroló készítése, amelyik előállítja a bemeneti input fájlból a < szám, db > párokat, mert akkor ezek kiírása már az összegzés tétellel megoldható.

- Próbáljunk készíteni egy ilyen felsorolót $\Rightarrow$ egyedi felsorolós a feladatunk.

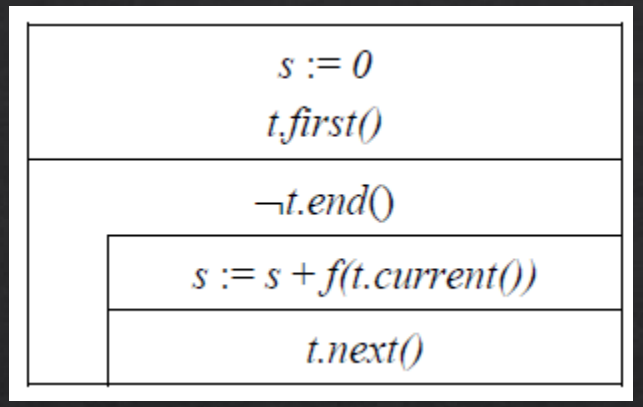
- Fussunk neki újra a specifikációnak, képzeljük el, hogy van egy megfelelő felsorolónk.

Feladat megoldása az egyedi felsorolón

- Új specifikáció:

A = (t:enor(Össz), y:outfile(Össz))
Össz = rec(szám:$\mathbb{Z}$, db:$\mathbb{N}$)
Ef = (t = t')
Uf = (y = t') = (y = $\bigoplus_{e \in t'}$ <e>)

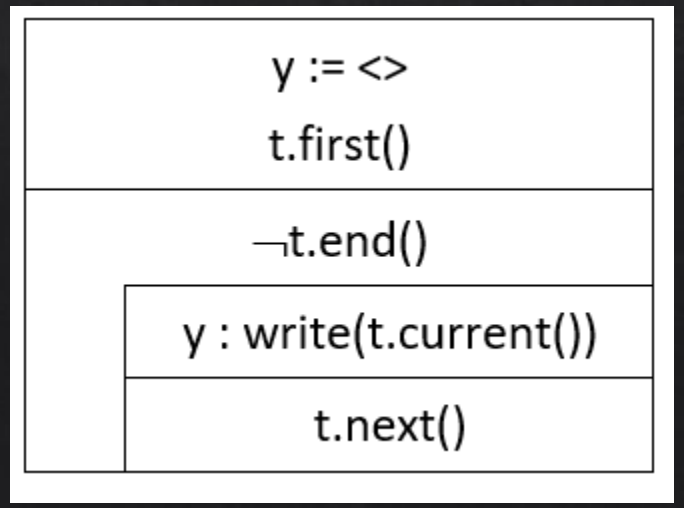
- Összegzés tétel sablonja



- Folytassuk visszavezetéssel:

Összegzés (másolás)
felsoroló ~ egyedi felsoroló
f(e) ~ <e>
s ~ y
H, +, 0 ~ Össz*, $\bigoplus$, <>

- A kapott algoritmus:



Következő lépés: tervezzük meg a felsorolót!

t:enor(Össz) Össz = rec(num:ℤ, count:ℕ)

- Mi kell a tervhez:
- Állapottér, és a felsoroló tanult műveletei:

Össz*	first()	next()	current() : Össz	end() :𝕃
x : infile(ℤ) dx : ℤ sx : Status akt : Össz vége : 𝕃	sx,dx,x:read next()	lásd külön	**return** akt	**return** vége

next() művelet megtervezése

- Mi lesz a művelet feladata?
  Induláskor lesz egy beolvasott adat, addig kell beolvasnia, amíg a fájl el nem fogy, és ugyanazt a számot olvassa folyamatosan.
- Mi történik a fájl végén?
  Olvasás közben elérkezik a fájl vége, az addig elvégzett számlálás eredményét még át kell adnia a felsorolót használó algoritmusnak, hogy feldolgozhassa, csak a következő next() hívásnál jelezheti, hogy vége a felsorolásnak.

Bemeneti fájl egy részlete:

1				⇒ < 1 , 1 >
2	2			
2	2	2		⇒ < 2 , 5 >
3	3	3	3	*Amíg hármasokat olvasunk, számoljuk az elemek számát.*
3	3	3		*Amikor az első négyeshez érünk, a next() leáll,*
4	4	4		*current() = < 3, 7 >*
...				
8	8			
9				
9	9			*Elérkeztünk a fájl végéhez, de még van current() elem:*
9	9	9		*current() = < 9 , 6 >*

next() művelet

- Specifikáció:

A = (x:infile($\mathbb{Z}$), dx:$\mathbb{Z}$, sx:Status, akt:Össz, vége:$\mathbb{L}$)

Ef = (x = x' $\wedge$ x↗ $\wedge$ dx = dx' $\wedge$ sx = sx')

Uf = (vége = (sx'=abnorm) $\wedge$ ($\neg$vége $\rightarrow$ akt.szám=dx' $\wedge$ (akt.db, (sx,dx,x))=$\sum_{dx \in (dx',x')} 1$))

dx = akt.szám

Fontos 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.

- Visszavezetés:

Összegzés

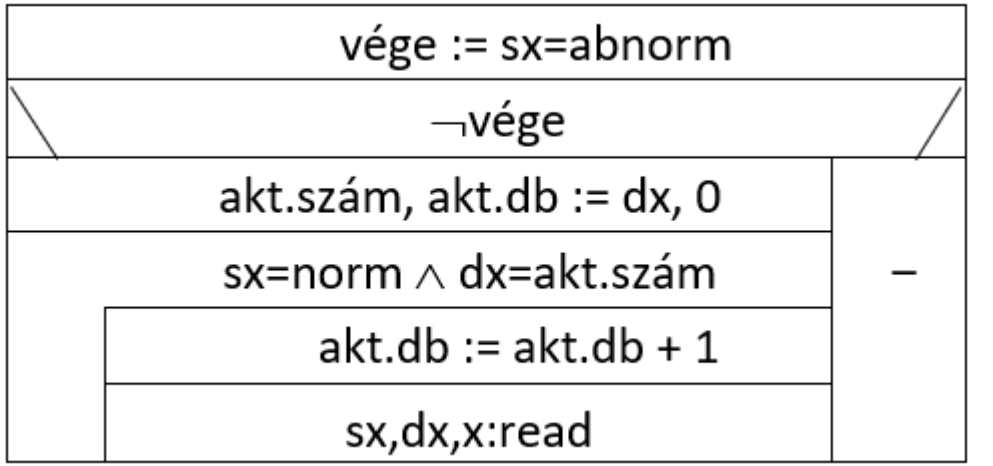
t:enor(E) ~ x:infile($\mathbb{Z}$) (sx,dx,x:read)
first() nélkül, amíg: dx=akt.szám

f(e) ~ 1

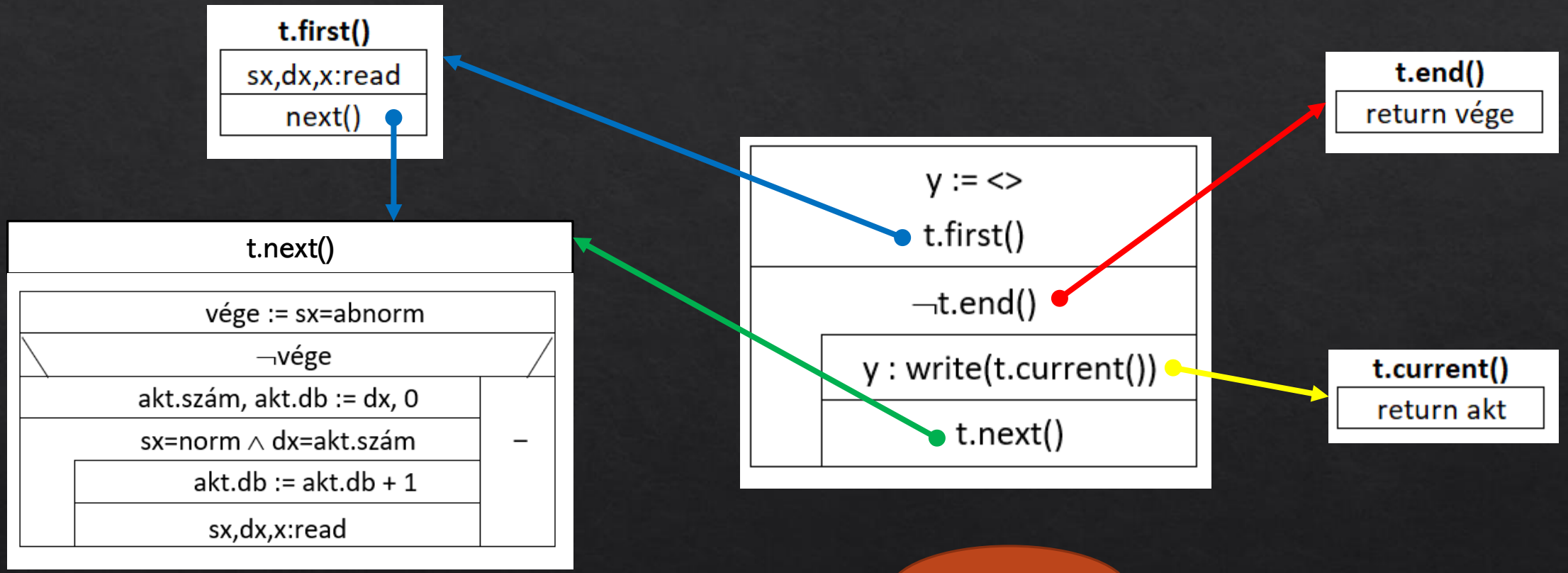
s ~ akt.db

H, +, 0 ~ $\mathbb{N}$, +, 0

- Algoritmus:



Első feladat algoritmusai és kapcsolatuk



Kvíz/3

Egyedi felsoroló 3. feladat

- 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.)

Bemenet:

A mássalhangzók hosszúságát
betűkettőzéssel jelöljük
Idegen betűkapcsolatokat
tartalmazó tulajdonnevek

Eredmény: **13**

Specifikáció készítés

- Próbáljuk meg specifikálni:

Specifikáció:

A = (f:infile($\mathbb{K}$), c:$\mathbb{N}$)

Ef = (f = f_0)

UF ???

- Mi a probléma?
  Ezen az állapottéren nem tudjuk pogramozási tételre visszavezetni a feladatot.

Tervezzünk különböző, alkalmas felsorolót a feladathoz: Kvíz/4

- Megoldás: egy olyan felsoroló kell, amelyik előállítja a bemeneti input fájlból a a szavak hosszait amelyen a feladat az összegzés tétellel megoldható.

- Próbáljunk készíteni egy ilyen felsorolót $\Rightarrow$ egyedi felsorolós a feladatunk.

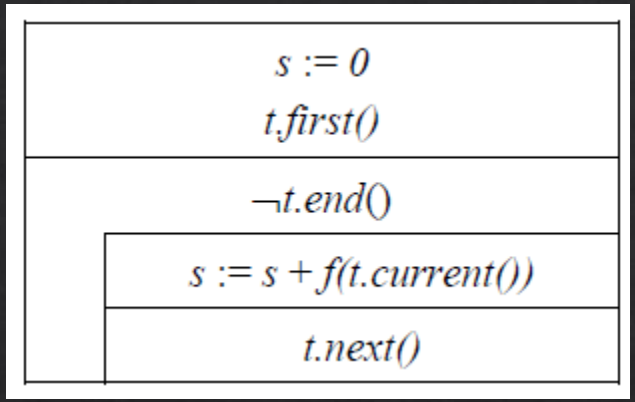
- Fussunk neki újra a specifikációnak, képzeljük el, hogy van egy megfelelő felsorolónk.

Feladat megoldása az egyedi felsorolón

- Új specifikáció:

A = (t:enor($\mathbb{N}$), c:$\mathbb{N}$)
Ef = (t = t_0)
Uf = (c = $\sum_{e \in t_0}$ {2, ha e>12; 1, ha e≤12})

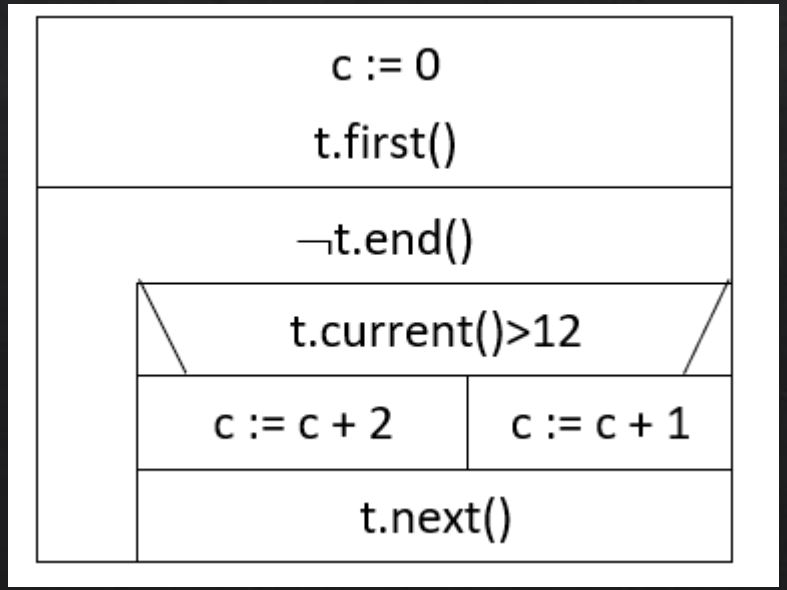
- Összegzés tétel sablonja



- Folytassuk visszavezetéssel:

Összegzés
felsoroló ~ egyedi felsoroló
f(e) ~ {2, ha e>12; 1, ha e≤12}
s ~ c
H, +, 0 ~ $\mathbb{N}$, +, 0

- A kapott algoritmus:



Következő lépés: tervezzük meg a felsorolót!

- Mi kell a tervhez:
- Állapottér, és a felsoroló tanult műveletei:

Kvíz/5

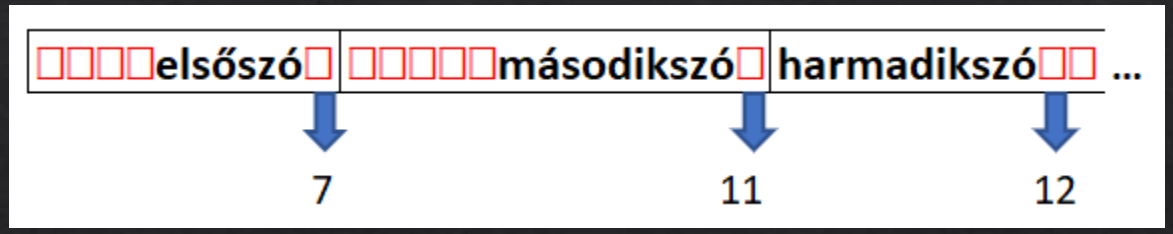
Felsoroló:

t:enor(ℕ)

ℕ*	first()	next()	current() : ℕ	end() : 𝕃
f : infile(𝕂) ch : 𝕂 st : Status akt : ℕ vége : 𝕃	st,ch,f:read next()	lásd külön	**return** akt	**return** vége

next() művelet megtervezése

- Mi lesz a művelet feladata?
  Induláskor lesz egy előre olvasott karakter, ami lehet akár egy szóköz! Tehát elsőként a szóközöket át kell olvasnia, amíg elérkezik az első nem-szóköz karakterhez. Onnan addig kell tovább olvasnia, amíg nem-szóköz karakterek jönnek, és közben meg kell számolnia a beolvasott karaktereket. A második hívástól kezdve, amíg nem érünk a fájl végére, már biztos, hogy az előre olvasott karakter az egy szóköz lesz!



- Mi történik a fájl végén?
  Olvasás közben elérkezik a fájl vége, az addig elvégzett összegzés eredményét még át kell adnia a felsorolót használó algoritmusnak, hogy feldolgozhassa, csak a következő next() hívásnál jelezheti, hogy vége a felsorolásnak. (Előfordulhat, hogy az utolsó szó után közvetlenül a fájl vége jel áll, de állhatnak szóközök is az utolsó szót követően.)

next() művelet

- Most egy kicsit bonyolultabb a specifikáció, hiszen két tétel is rejtőzik a next() algoritmusában:
  elsőként egy kiválasztás: kiválasztjuk az első nem szóköz karaktert,
  majd egy feltételtől függő összegzés, addig számláljuk a karaktereket, amíg újra egy szóközzel, vagy esetleg a fájl végével találkozunk.
- A specifikációban a két tétel közti átmenetet is meg kell jelenítenünk!

next() művelet

$A = (\ f{:}infile(\mathbb{K}), ch{:}\mathbb{K}, st{:}Status, akt{:}\mathbb{N}, vége{:}\mathbb{L}\ )$

$Ef = (\ f = f' \wedge ch = ch' \wedge st = st'\ )$

$Uf = (\ (ch'',(st'',ch'',f''))=SELECT_{ch\in(ch',f')}\ (st=abnorm \vee ch\neq' '\ ) \wedge \quad _{ch\neq' '}$

$vége=(st''=abnorm) \ \wedge \ (\ \neg vége \rightarrow (akt, (st,ch,f)) = \sum_{ch\in(ch'',f'')} 1\ )\ )$

- *Megjegyzés: a kiválasztásnak két eredménye van: a keresett szóköz (ch”), amelyhez nem vezetünk be külön output változót; és a felsoroló aktuális állapota, amelyet az (st”,ch”,f”) hármas ír le. (Nem okozna félreértést, ha csak az (st”,ch”,f”)-t tüntetnénk fel a kiválasztás eredményeként.)*

next() művelet folytatása

- Visszavezetések:

Kiválasztás

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

felt(e) ~ st=abnorm ∨ ch≠' '

Összegzés (megszámolás)

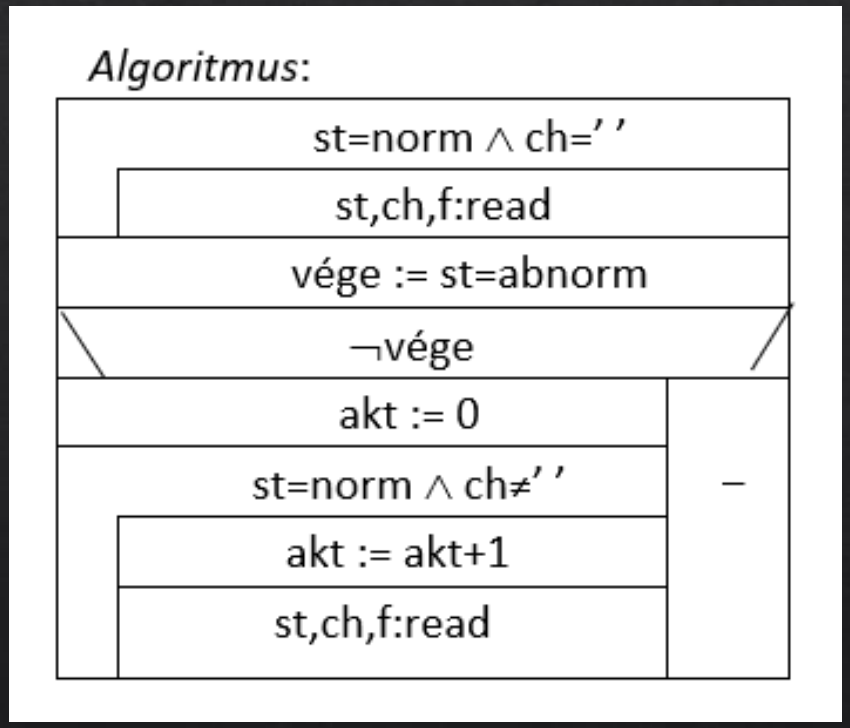
t:enor(E) ~ f:infile(𝕂) (st,ch,f:read)
first() nélkül, amíg: ch≠' '

f(e) ~ 1

s ~ akt

H, +, 0 ~ ℕ, +, 0

- Algoritmus:



Harmadik feladat algoritmusai és kapcsolatuk.