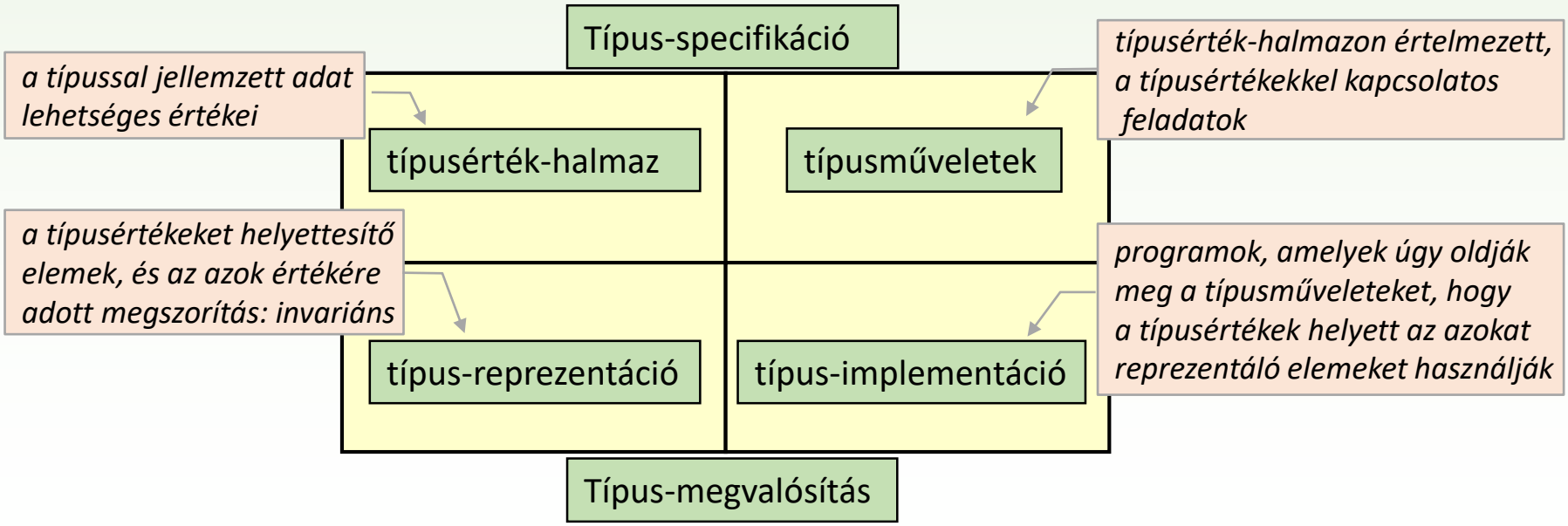
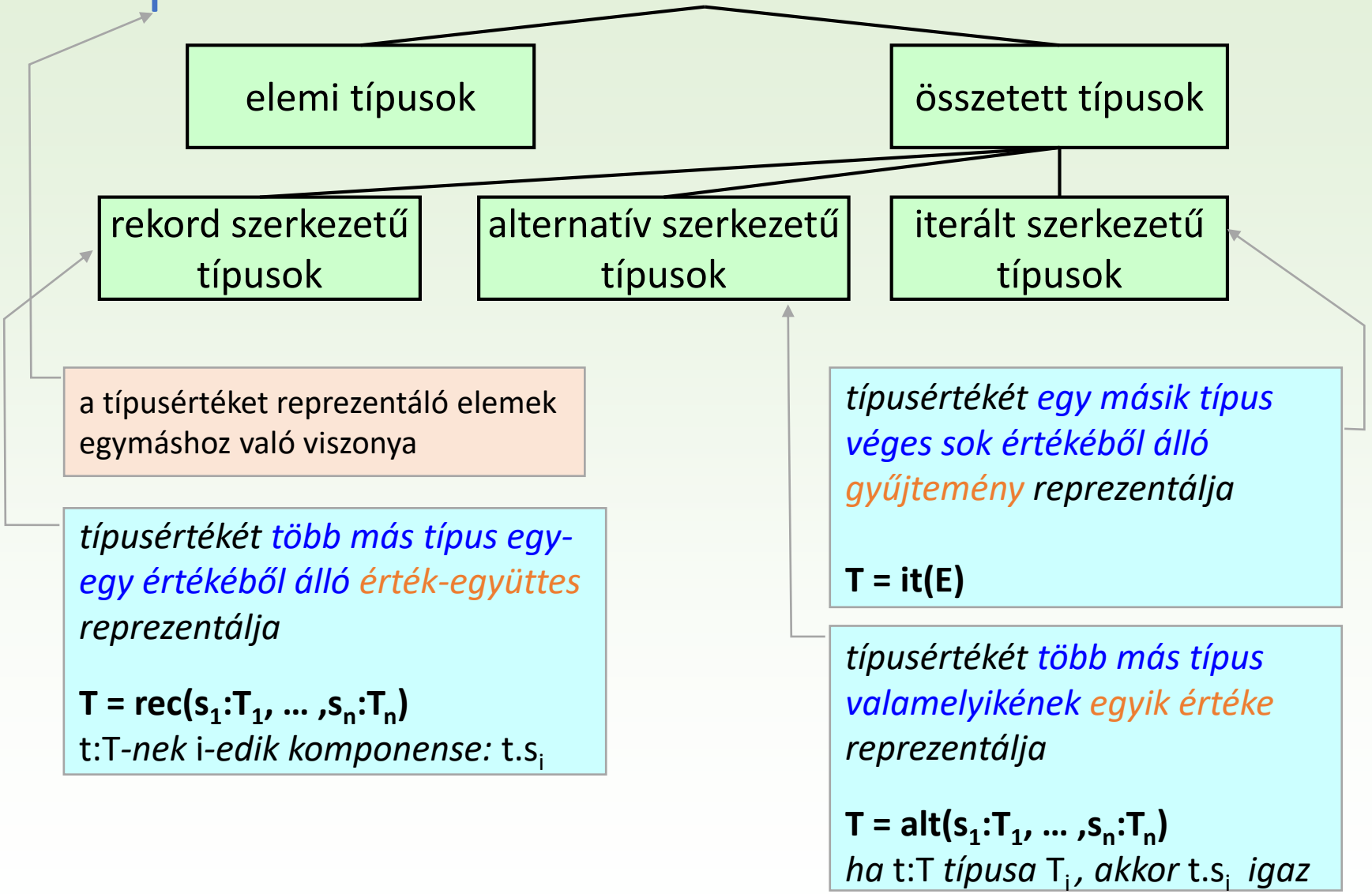
Gyűjtemények, felsorolók, programozási tételek

Adattípus fogalma

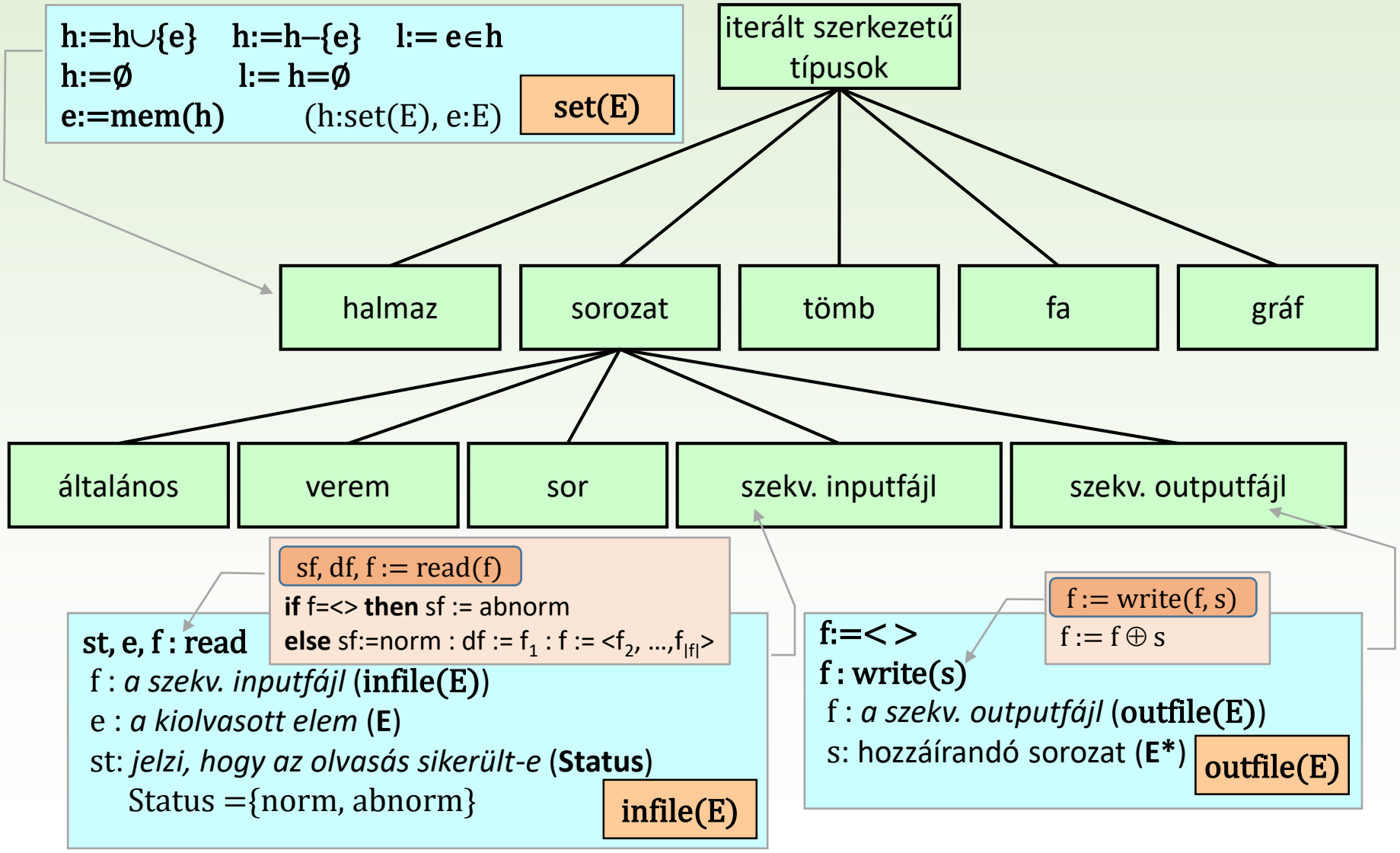
- Egy adat (speciálisan egy objektum) típusát az adat által felvehető típusértékek halmaza és az azokkal végezhető típusműveletek együttesen mutatják meg. Ez a típus specifikációja.
- A típus megvalósítása azt mutatja meg, hogyan szeretnénk a típusértékeket a számítógépen ábrázolni, azaz reprezentálni, és ennek ismeretében milyen programok oldják meg, azaz implementálják a típusműveletek feladatait.



Típus szerkezet



Nevezetes iterált szerkezetű típusok



Gyűjtemény feldolgozása

- A gyűjtemény (tároló, kollekció, iterált) egy olyan objektum, amely elemek tárolására alkalmas, és az eltároláshoz, valamint a visszakereséshez biztosít műveleteket.
  - Ilyenek az összetett szerkezetű, de különösen az iterált típusú objektumok: halmaz, sorozat (verem, sor, fájl),tömb, fa, gráf.
  - Vannak úgynevezett virtuális gyűjtemények is, amely elemeit nem kell explicit módon tárolni: pl. egész számok egy intervallumának elemei, vagy egy természetes szám prím-osztói.
- Egy gyűjtemény feldolgozásán a benne levő elemek feldolgozását értjük.
  - Keressük a halmaz legnagyobb elemét!
  - Hány negatív szám van egy számsorozatban?
  - Járjuk be az [*m* .. *n*] intervallum minden második elemét visszafelé!
  - Adjuk össze az *n* természetes szám prím-osztóit!

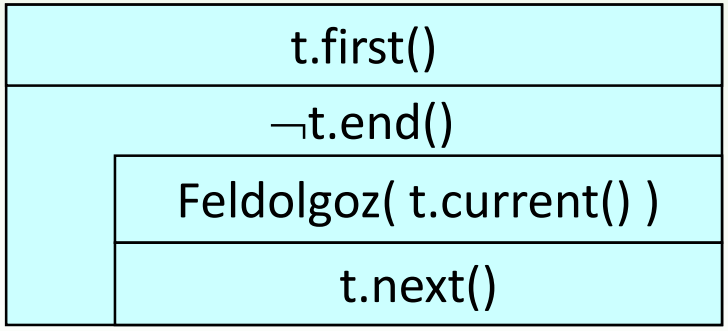
Felsorolás

- Egy gyűjtemény *E* típusú elemeinek felsorolását egy *E** halmazbeli sorozatként foghatjuk fel, amelynek bejárását az alábbi műveletek végzik:
  - *first()* : rááll a felsorolás első elemére, azaz elkezdi a felsorolást
  - *next()* : rááll az elkezdett felsorolás soron következő elemére
  - *l* := *end()* (*l*:𝕃) : mutatja, hogy a felsorolás végére értünk-e
  - *e*:= *current()* (*e*:*E*): visszaadja a felsorolás aktuális elemét

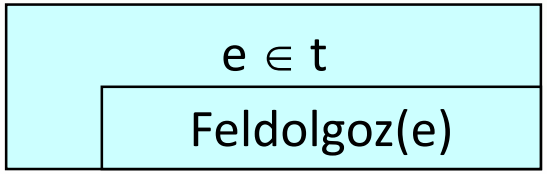
Felsorolás állapotai

- Egy felsorolásnak különböző állapotai vannak (*indulásra kész*, *folyamatban van*, *befejeződött*): műveletei csak bizonyos állapotokban értelmesek (máshol nem definiált a hatásuk).
- A feldolgozó algoritmus garantálja, hogy egy felsoroló műveletet mindig akkor (olyan állapotban) hajt végre, amikor az értelmes.

t : enor(E)




```
for(t.first(); !t.end(); t.next())
{
    feldolgoz(t.current());
}
```



foreach (forall) ciklus

```
for( auto e : t )
{
    feldolgoz(e);
}
```

Felsorolás objektummal

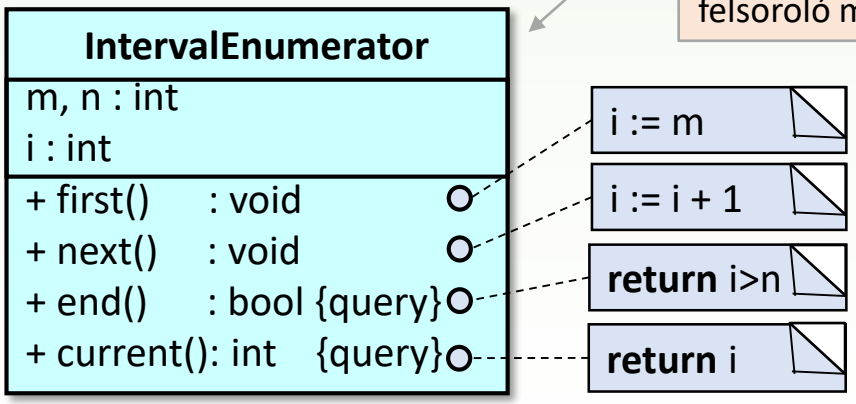
- A felsorolást a felsorolni kívánt gyűjteménytől elkülönülő objektum végzi, amelyből ugyanazon gyűjteményhez egyszerre több is lehet.
- Egy felsoroló objektum típusát az *enor*(*E*) jelöli.
- A felsoroló objektum típusának megvalósítása a bejárandó gyűjtemény típusától függ.
  - Mivel a felsoroló objektumnak ismernie kell az általa bejárt gyűjteményt, ezért a reprezentációja nyilvántart egy hivatkozást a gyűjteményre.
  - A felsoroló műveletek implementációja az esetek többségében segéd adatokat igényel.
- Előnyös, ha a felsorolót a felsorolni kíván gyűjtemény egy metódusa hozza létre: így a gyűjtemény biztosan értesül arról, hogy őt bejárják.

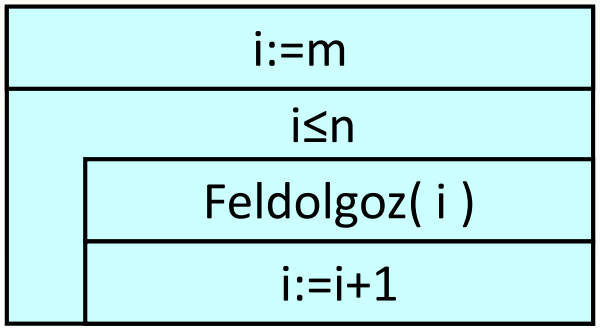
Intervallum klasszikus felsorolója

Egész számok intervallumába eső egész számok felsorolása növekedően.

enor(ℤ)				
ℤ*	first()	next()	l:= end()	e:= current()
m , n : ℤ i : ℤ	i:=m	i:=i+1	l:= i>n l:𝕃	e:=i e:ℤ

a megvalósításnál nem szoktuk ezt az osztályt definiálni, így a felsoroló maga az *i* változó lesz



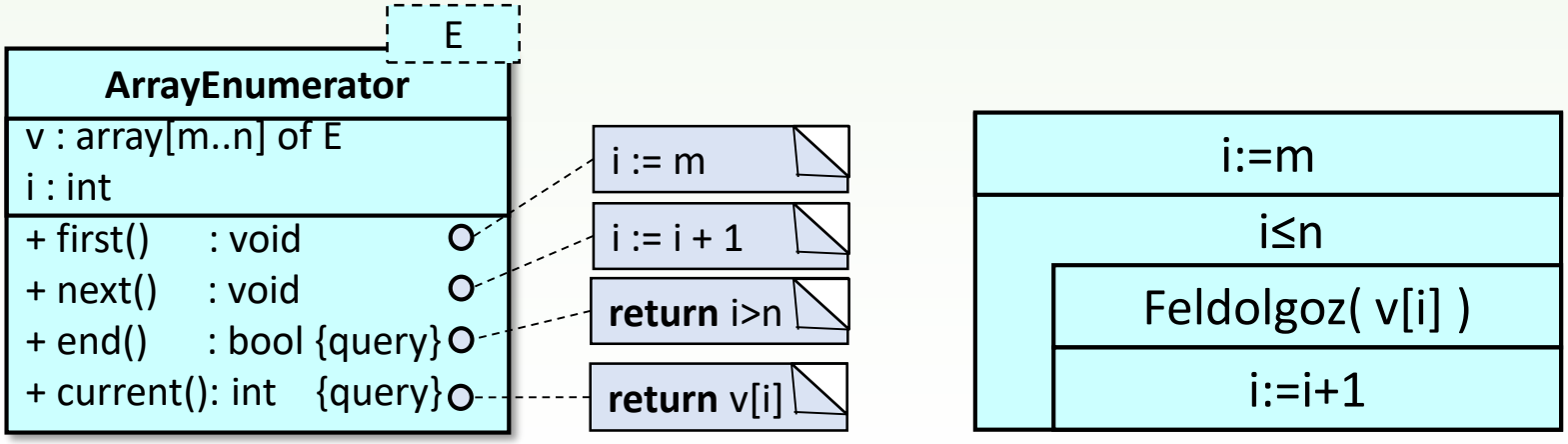


Vektor klasszikus felsorolója

E-beli értékekből álló vektor elemeinek felsorolása elejétől a végéig

enor(*E*)

E*	first()	next()	l:= end()	e:= current()
v : $E^{m..n}$ i : $\mathbb{Z}$	i:=m	i:=i+1	l:= i>n l:$\mathbb{L}$	e:=v[i] e:E

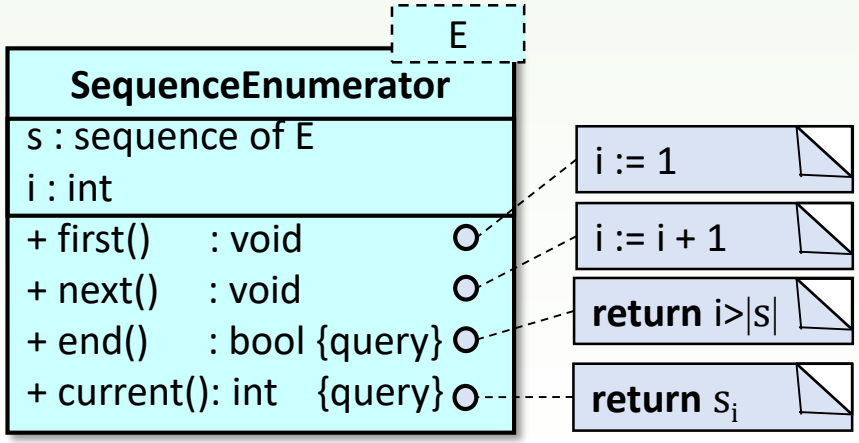


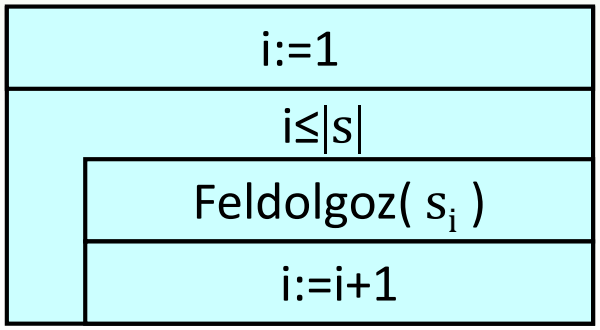
Sorozat klasszikus felsorolója

E-beli értékek véges sorozatának felsorolása elejétől a végéig

enor(*E*)

E*	first()	next()	l:= end()	e:= current()
s : E^* i : $\mathbb{Z}$	i:=1	i:=i+1	l:= i>|s| l:$\mathbb{L}$	e:= s_i e:E



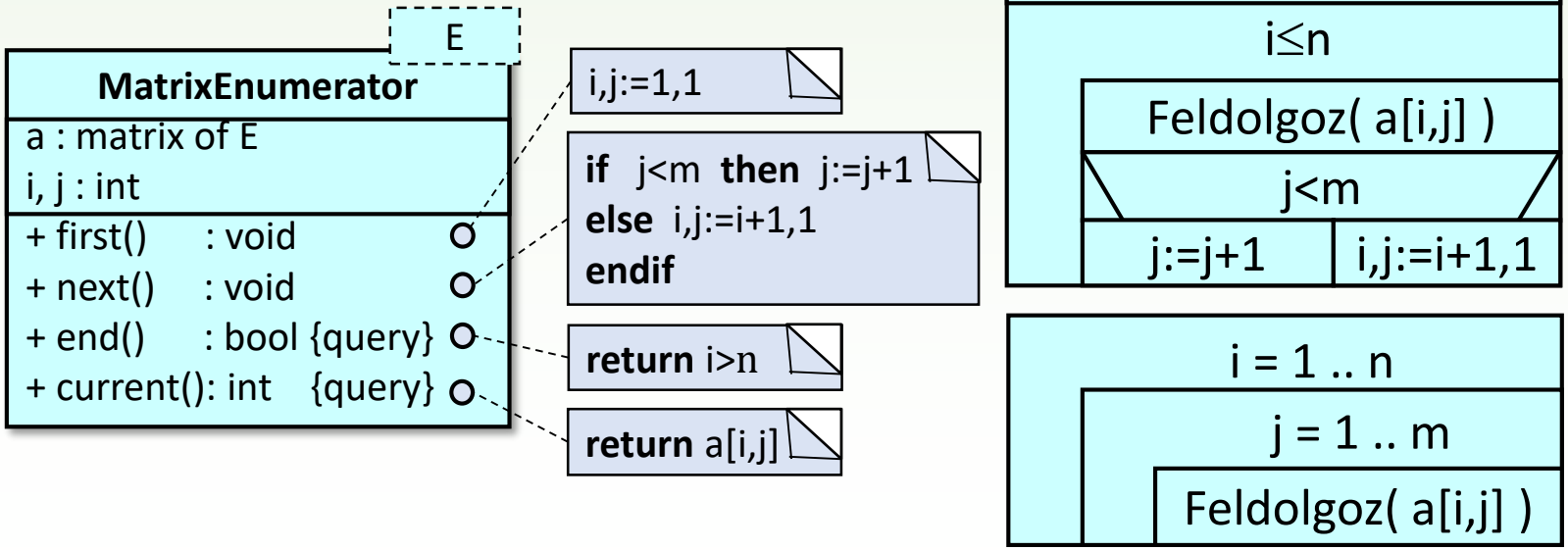


Mátrix sorfolytonos felsorolója

E-beli értékekből álló mátrix elemeinek felsorolása sorfolytonos sorrendben

enor(*E*)

E*	first()	next()	l:= end()	e:= current()
a : $E^{n \times m}$ i,j : $\mathbb{Z}$	i,j:=1,1	**if** j<m **then** j:=j+1 **else** i,j:=i+1,1	l:= i>n l:$\mathbb{L}$	e:= a[i,j] e:E

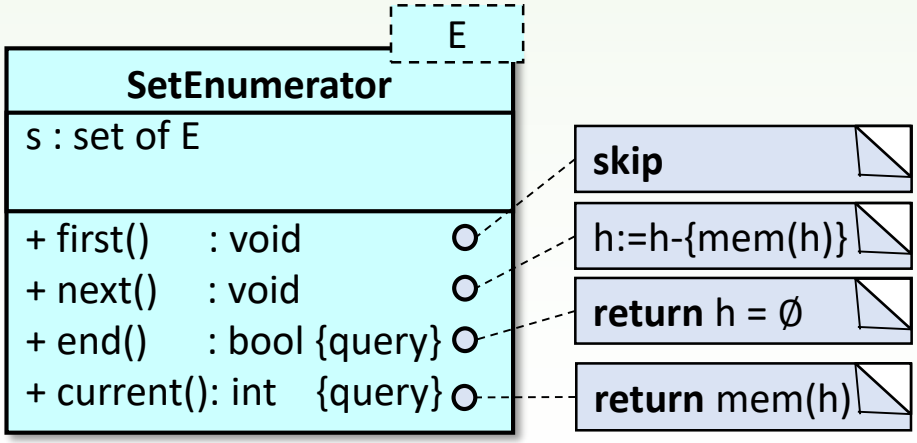


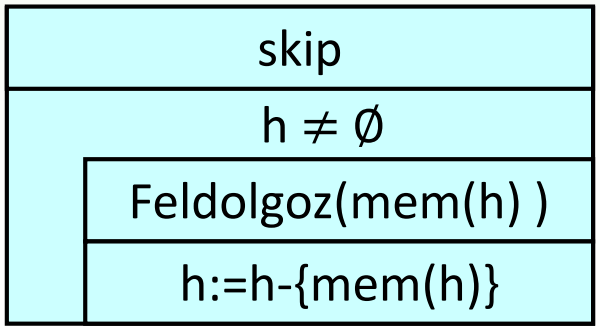
Halmaz klasszikus felsorolója

E-beli értékek véges halmazának felsorolása

ez a felsorolás elfogyasztja a felsorolt halmazt

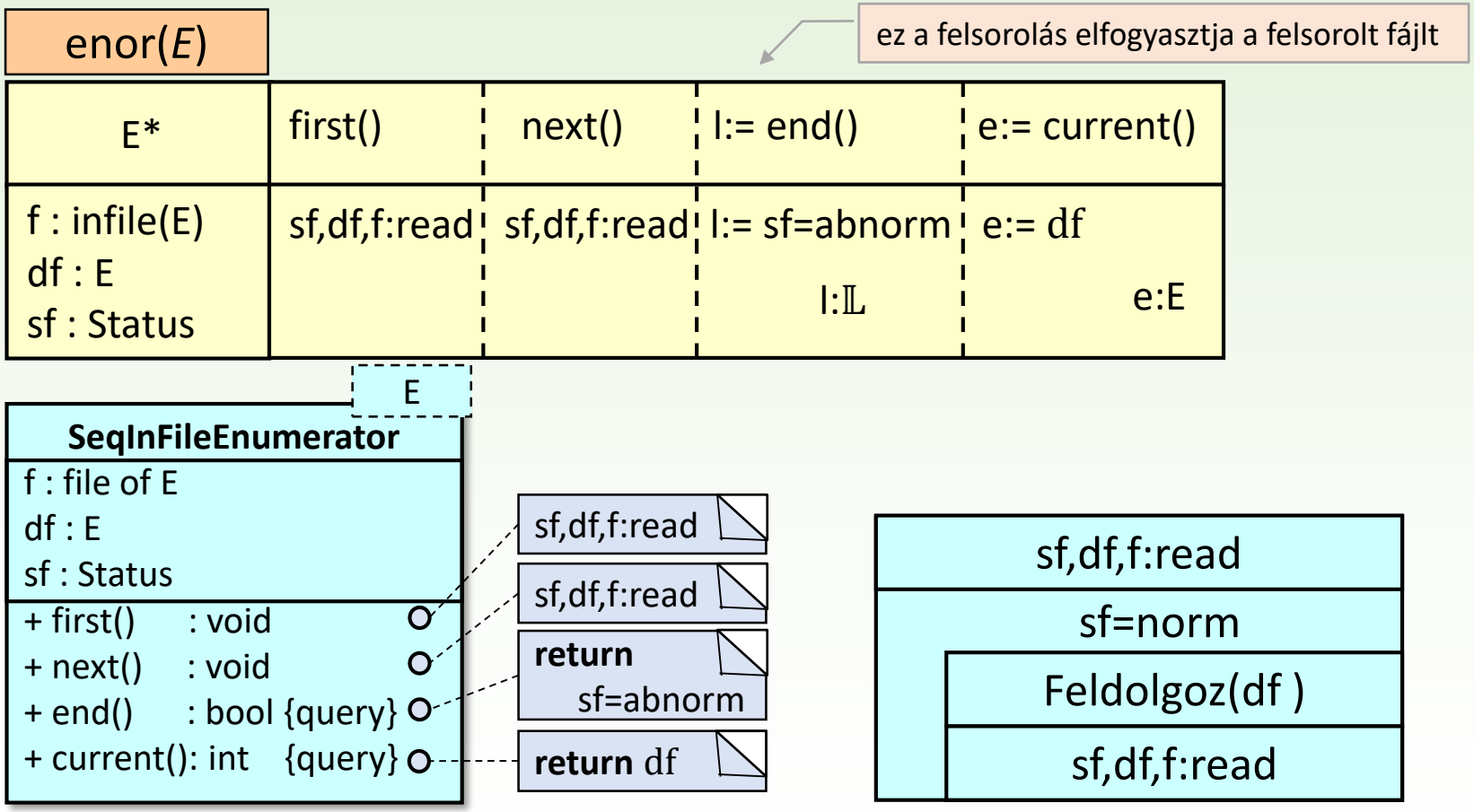
enor(*E*)				
E*	first()	next()	l:= end()	e:= current()
h : set(E)	skip	h:=h-{mem(h)}	l:= h = ∅ l:𝕃	e:= mem(h) e:E





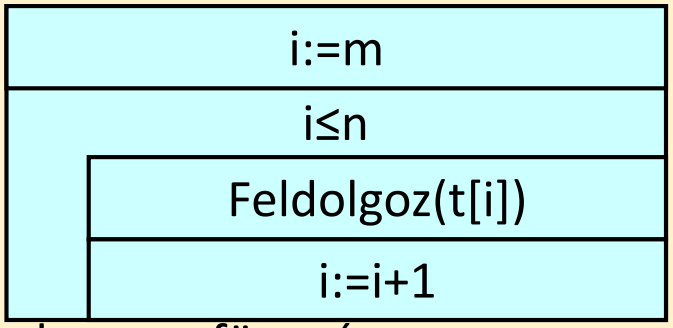
Szekvenciális inputfájl felsorolója

E-beli értékeket tartalmazó szekvenciális inputfájl elemeinek felsorolása

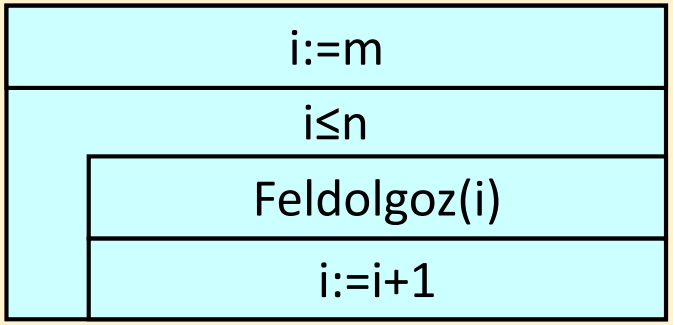


Programozási tételek általánosítása

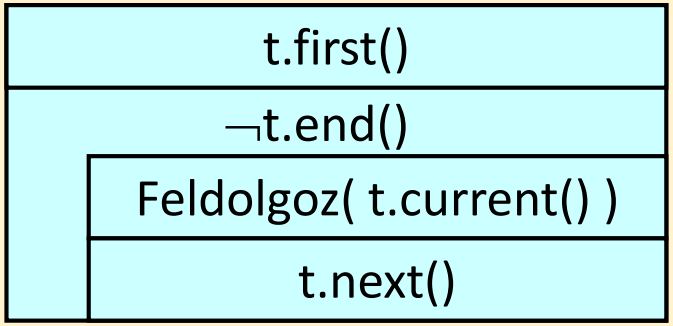
- Programozási tételek tömbre:
  - $t : E^{m..n}$ $(E^{1..n} = E^n)$
  - $f : E \to H$, $felt: E \to \mathbb{L}$



- Programozási tételek intervallumon értelmezett függvényre:
  - [m .. n]
  - $f:[m .. n] \to H$, $felt:[m .. n] \to \mathbb{L}$



- Programozási tételek felsorolóra:
  - t : enor(E)
  - $f:E \to H$, $felt:E \to \mathbb{L}$



Összegzés

Összegezzük egy felsorolás elemeihez rendelt értékeket!

A : t:enor(E), s:H

Ef : t = t'

Uf: s = $\Sigma_{e \in t'} f(e)$

$f : E \to H$

$+ : H \times H \to H$

$0 \in H$ *baloldali neutrális elemmel*

$\Sigma_{e \in t'} f(e) = (\ldots(f(e_1) + f(e_2)) + \ldots) + f(e_n)$,
ahol $e_1, \ldots, e_n$ a t' felsorolás elemei

speciális eset: feltételes összegzés:

$$\Sigma_{\substack{e \in t' \\ felt(e)}} g(e) \text{ azaz } f(e) = \begin{cases} g(e) & \text{ha felt(e)} \\ 0 & \text{különben} \end{cases}$$

```
s := 0
t.first()
¬t.end()
    s := s + f(t.current())
    t.next()
```

Számlálás

Számoljuk meg egy felsorolás adott tulajdonságú elemeit!

A : t:enor(E), c:$\mathbb{N}$

Ef : t = t'

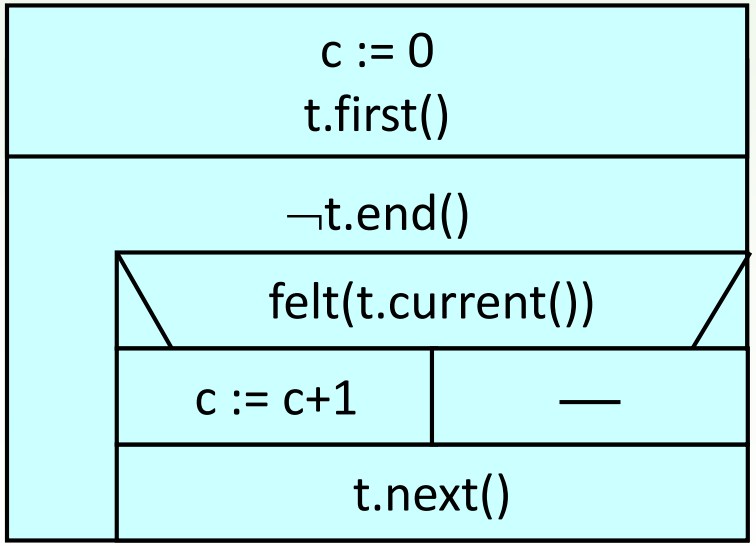
Uf: $c = \Sigma_{e \in t'} 1$ (felt(e))

felt:E → $\mathbb{L}$

a természetes számokon értelmezett összeadás

számlálás egy speciális összegzés:

$\Sigma_{e \in t'} f(e)$ azaz $f(e) = \begin{cases} 1 & \text{ha felt(e)} \\ 0 & \text{különben} \end{cases}$



Maximum kiválasztás

Adjuk meg egy felsorolás adott szempont szerinti egyik legnagyobb elemét és annak értékét!

A : t:enor(E), elem:E, max:H

Ef : t = t' $\wedge$ |t|>0)

Uf: (max , elem)=**MAX** $_{e \in t'}$ f(e)

f:E $\rightarrow$ H
H *halmaz elemei rendezhetőek*

max = *f*(*elem*) = **MAX** $_{e \in t'}$ *f*(*e*)
$\wedge$ *elem* $\in$ *t'*

- MAX *helyett lehet* MIN
- elem *elhagyható,* max *nem*

t.first()

max, elem := f(t.current()), t.current()

t.next()

¬t.end()

f(t.current())>max

max, elem := f(t.current()), t.current()

—

t.next()

Kiválasztás (biztosra megy)

Keressük meg egy felsorolás adott tulajdonságú első elemét, ha tudjuk, hogy van ilyen!

A : t:enor(E), elem:E

Ef : t = t' $\wedge$ $\exists e \in t$: felt(e)

Uf : (elem, t) = **SELECT** $_{e \in t'}$ felt(e)

felt:E $\rightarrow \mathbb{L}$

A *t* felsorolása a kiválasztás végén még *„folyamatban van”* , vannak még elemei.

Megkeresi a *t'* felsorolás első olyan elemét (ez lesz az *elem*), amelyre a feltétel teljesül.
Formálisan:
$felt(e_i) \wedge \forall_{k=1..i-1} \neg felt(e_k) \wedge elem=e_i$,
ahol e_1, e_2 , … a *t'* felsorolás elemei

```
t.first()
¬felt(t.current())
    t.next()
elem := t.current()
```

Lineáris keresés (bizonytalan a siker)

Keressük meg egy felsorolás adott tulajdonságú első elemét!

A : t:enor(E), l:𝕃, elem:E

Ef : t = t'

Uf : (l, elem, t) = **SEARCH** $_{e \in t'}$ felt(e)

felt:E → 𝕃

A *t* felsorolása csak sikertelen keresés esetén lesz *„befejeződött”*; egy sikeres keresés végén vannak még elemei, még *„folyamatban van”*.

Megkeresi a *t'* felsorolás első olyan elemét (ez lesz az *elem*), amelyre a feltétel teljesül. Ha talál ilyet, akkor *l* igaz, különben hamis lesz.
Formálisan:
$l = \exists_{e \in t'}\ felt(e) \wedge (l \rightarrow felt(e_i) \wedge \forall_{k=1..i-1} \neg felt(e_k) \wedge elem = e_i$, ahol $e_1, \ldots, e_n$ a *t'* felsorolás elemei)

speciálisan eldöntésre használjuk:
l = **SEARCH** $_{e \in t'}$ felt(e) vagy l = $\exists_{e \in t'}$ felt(e)

```
l := hamis; t.first()
¬l ∧ ¬t.end()
    elem := t.current()
    l := felt(elem)
    t.next()
```

Optimista lineáris keresés

Ellenőrizzük, hogy egy felsorolás minden elemére igaz egy adott tulajdonság, de ha nem, megadjuk az első olyan elemet, amelyikre nem teljesül!

A : t:enor(E), l:𝕃, elem:E

Ef : t = t'

Uf : (l, elem, t) = ∀**SEARCH** $_{e \in t'}$ felt(e)

felt:E → 𝕃

A *t* felsorolása csak sikertelen keresés esetén lesz *„befejeződött”*; egy sikeres keresés végén vannak még elemei, még *„folyamatban van”*.

Ha a *t'* felsorolás minden elemére teljesül a feltétel, akkor *l* igaz lesz, különben hamis lesz. Ezesetben az *elem* a felsorolás első olyan eleme, amelyre nem teljesül a feltétel.
Formálisan:
$l = \forall_{e \in t'}\ felt(e) \wedge (\neg l \rightarrow \neg felt(e_i) \wedge \forall_{k=1..i-1}\ felt(e_k)$
$\wedge\ elem = e_i$, ahol $e_1, \ldots, e_n$ a *t'* felsorolás elemei)

speciálisan eldöntésre használjuk:
l = ∀**SEARCH** $_{e \in t'}$ felt(e) vagy l = ∀ $_{e \in t'}$ felt(e)

l := igaz; t.first()

l ∧ ¬t.end()

elem := t.current()

l := felt(elem)

t.next()

Feltételes maximum keresés

Keressük egy felsorolás adott tulajdonságú elemei között egy adott szempont szerinti egyik legnagyobbat és annak értékét!

A : t:enor(E), l:$\mathbb{L}$, elem:E, max:H

Ef : t = t'

Uf : (l, max, elem) = $\mathbf{MAX}_{\substack{e \in t' \\ felt(e)}} f(e)$

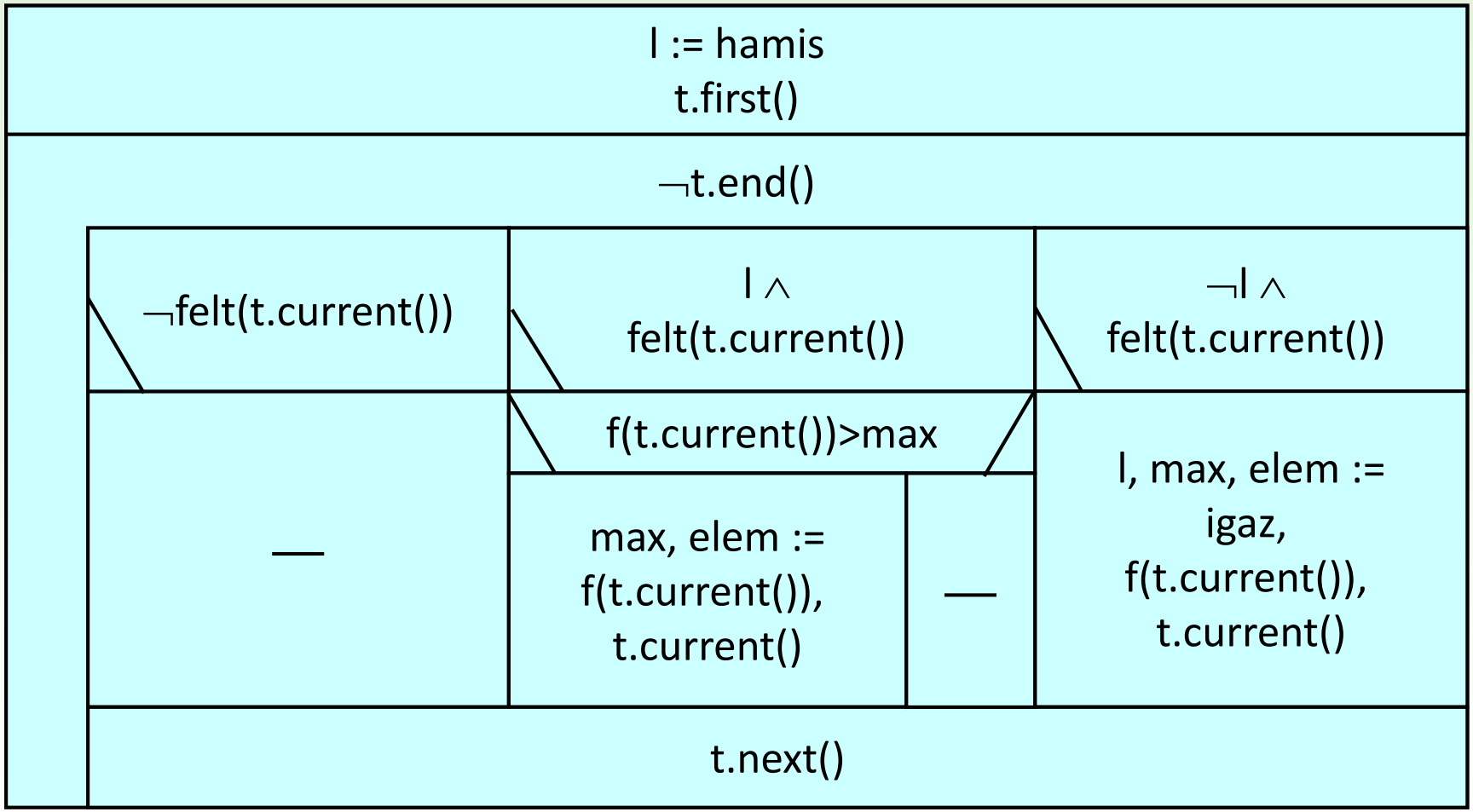
f:E → H

felt:E → $\mathbb{L}$

H *halmaz elemei rendezhetőek*

$l = \exists_{e \in t'} felt(e) \wedge (l \longrightarrow max = f(elem) = \mathbf{MAX}_{\substack{e \in t' \\ felt(e)}} f(e) \wedge elem \in t')$

Feltételes maximum keresés



- MAX *helyett lehet* MIN
- elem *elhagyható,* max *nem*

Visszavezetés lépései

1. Megsejtjük a feladatot (részfeladatot) megoldó programozási tételt.
2. Specifikáljuk a feladatot a programozási tételre utaló végrehajtható utófeltétellel.
3. Megadjuk a feladat és a programozási tétel közötti eltéréseket:
   - *felsoroló* típusát a *felsorolt elemek* típusával együtt
   - *függvények* (f:[m..n]→H, felt:[m..n]→$\mathbb{L}$) konkrét megfelelőit
   - a H *műveletét*, ha kell
     - (H, >) helyett például ($\mathbb{Z}$, >) vagy ($\mathbb{Z}$, <)
     - (H, +) helyett például ($\mathbb{Z}$, +) vagy ($\mathbb{R}$, *) vagy ($\mathbb{L}$, $\wedge$)
   - *változók átnevezéseit*
4. Ráírjuk a programozási tétel algoritmusára a fenti különbségeket, hogy megkapjuk a feladatot megoldó algoritmust.

Tesztelési szempontok

- *Felsoroló* szerint (mindegyik tétel esetén)
  - *hosszúság*: nulla, egy illetve hosszabb felsorolásokra
  - *eleje* ill. *vége*: amikor a kitüntetett (az összegzendő, vagy a *felt* tulajdonságú, vagy a maximális) elem a felsorolás legelső illetve a legutolsó eleme.
- *Funkció* szerint
  - *keresés*: nincs illetve van *felt* tulajdonságú elem a felsorolásban
  - *max. ker.*: egyetlen illetve több azonos maximális érték van
  - *összegzés*: terheléses vizsgálat
- A *felt*(*i*) és *f*(*i*) kifejezések kiszámolásánál használt műveletek sajátosságai.