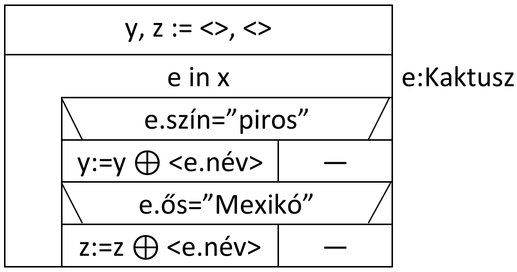
5. Algoritmus minták felsorolóra I.

1. Válogassuk ki kaktuszok sorozatából egyrészt a piros virágú kaktuszoknak, másrészt a mexikói őshazájú kaktuszoknak neveit!

Specifikáció:

$A = (x : enor(Kaktusz), y, z : \mathbb{S}^*)$
  $Kaktusz=rec(név:\mathbb{S}, szín:\mathbb{S}, ős:\mathbb{S}, méret:\mathbb{N})$
$Ef = (x=x_0)$
$Uf = (y=\bigoplus_{\substack{e \text{ in } x_0 \\ e.szín=\text{"piros"}}} \langle e.név\rangle \wedge z=\bigoplus_{\substack{e \text{ in } x_0 \\ e.ős=\text{"Mexikó"}}} \langle e.név\rangle)$

Algoritmus:

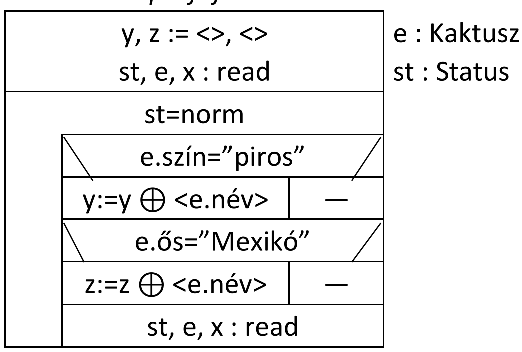


2 összegzés (kiválogatás) közös ciklusba vonva

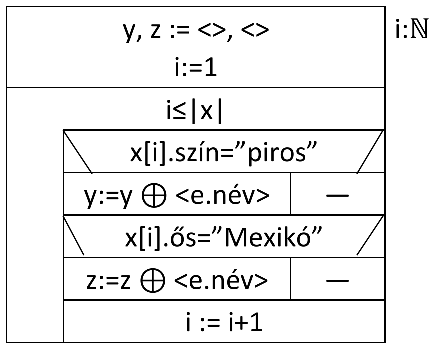
H, +, 0	~ $(\mathbb{S}^*, \oplus, \langle\rangle), (\mathbb{S}^*, \oplus, \langle\rangle)$
$f_1(e)$	~ <e.név> ha e.szín=”piros”
$f_2(e)$	~ <e.név> ha e.ős=”Mexikó”

Lehetne indexeléssel is implementálni, és ehhez akár számlálós ciklust (i=1 .. |x|) is használhatunk.

Szekvenciális inputfájlra:



Tömbre:



2. Keressük meg egy pozitív egész számokat tartalmazó nem üres sorozatban a legnagyobb számot, és közben döntsük el azt is, hogy vajon van-e páros szám.

Specifikáció:

$A = (x : enor(\mathbb{N}^+), l : \mathbb{L}, m : \mathbb{N})$
$Ef = (x=x_0 \wedge |x|\geq 1)$
$Uf = ((m, \_) = \mathbf{MAX}_{e \text{ in } x_0}\, e \wedge (l, \_) = \mathbf{SEARCH}_{e \text{ in } x_0} (e \text{ páros}))$

Az utófeltétel másképpen: két összegzéssel

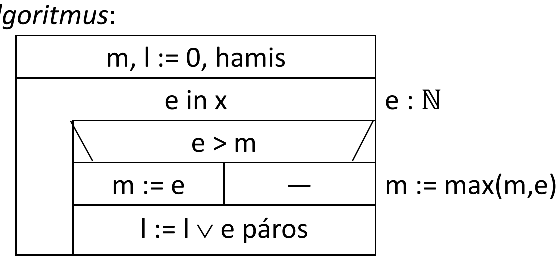
$Uf = (m = \underline{\mathbf{MAX}}_{e \text{ in } x_0}\, e \wedge l = \bigvee_{e \text{ in } x_0} (e \text{ páros}))$

ahol $\underline{max}: \mathbb{N}\times\mathbb{N}\to\mathbb{N}$, $\underline{max}(a,b)::= max(a,b)$ neutrális elem: 0
$\vee: \mathbb{L}\times\mathbb{L}\to\mathbb{L}$, $\vee(a,b)::= a\vee b$ neutrális elem: igaz

Két összegzés összevonva

H, +, 0	~ $(\mathbb{N}, \underline{max}, 0), (\mathbb{L}, \vee, hamis)$
f(e)	~ e, e páros
s	~ m, l

Algoritmus:



3. Adott egész számoknak egy felsorolása.
(Az alábbi példában a specifikációban használt jelöléseken van a hangsúly, a visszavezetést, az előállított algoritmusokat nem minden esetben részletezni.)
a) Hány páros szám van az első negatív szám előtt?

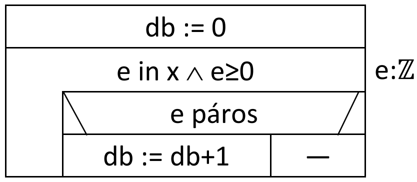
Specifikáció:

A = (x:enor(ℤ), db:ℕ)
Ef = (x=x_0)
Uf = (db = $\sum_{\substack{e \text{ in } x_0 \\ e \text{ páros}}}^{e\geq 0} 1$)

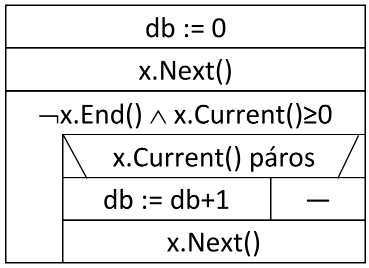
A $\sum$ jobbfelső sarkába írt feltétel mutatja, hogy meddig kell a felsorolást folytatni.

Számlálás, feltétel fennállásáig
felt(e) ~ e páros
c ~ db
t:enor(E) ~ x:enor(ℤ) amíg: e≥0

Algoritmus:



vagy



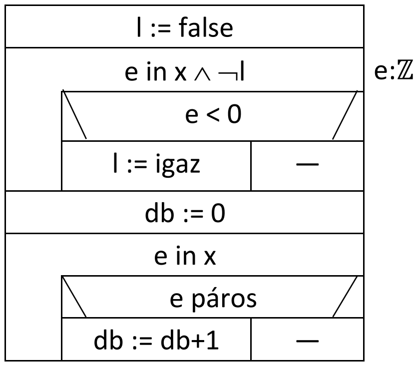
b) Hány páros szám van az első negatív szám után?

Specifikáció:

A = (x:enor(ℤ), db:ℕ)
Ef = (x=x_0)
Uf = ((_ , _ , x’) = **SEARCH**$_{e \text{ in } x_0}$ (e<0) ∧
∧ db = $\sum_{\substack{e \text{ in } x' \\ e \text{ páros}}} 1$)

A linker elsődleges outputjaira (a logikai értékre és a keresett elemre) most nincs szükség, csak a helyüket kell jelölni azért, hogy nyilvánvaló legyen, hogy az x’ a másodlagos outputot (a még fel nem sorolt elemek felsorolását) jelöli. Ezt folytatja a számlálás.

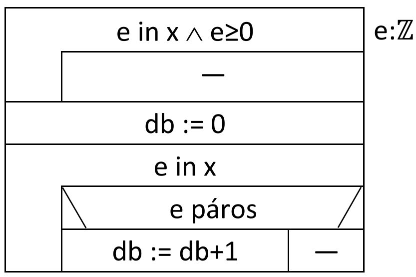
Algoritmus:



Másik megoldás:

Uf = ((_ , x’) = **SELECT**$_{e \text{ in } x_0}$ (e<0 ∨ |x|=0) ∧
∧ db = $\sum_{\substack{e \text{ in } x' \\ e \text{ páros}}} 1$)

A keresés feltétele biztosan teljesül, ha vagylagosan tartalmazza az |x|=0-t is: vagy találunk negatív elemet a felsorolásban vagy a felsorolás végére érünk). Ezért a lineáris keresés helyett a kiválasztás mintát is alkalmazhatjuk, aminek itt is a másodlagos outputja kell (x’).



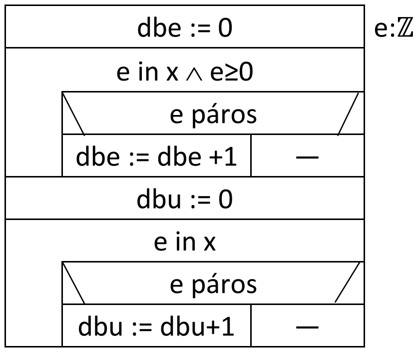
c) Hány páros szám van az első negatív szám előtt, és hány azután?

Specifikáció:

$A = (\ x: enor(\mathbb{Z}),\ dbe,\ dbu{:}\mathbb{N}\ )$
$Ef = (\ x=x_0\ )$
$Uf = ((dbe,\ \_\ ,\ x') = \sum_{\substack{e \text{ in } x_0 \\ e \text{ páros}}}^{e\geq 0} 1 \wedge dbu = \sum_{\substack{e \text{ in } x' \\ e \text{ páros}}} 1)$

A feltételig tartó összegzésnek másodlagos outputjai a feltételt ki nem elégítő első elem (de erre nincs szükségünk, csak a helyét jelöljük), és az ezt követő x' felsorolás.

Algoritmus:



d) Hány páros szám van az első negatív szám előtt, és vele kezdődően hány utána?

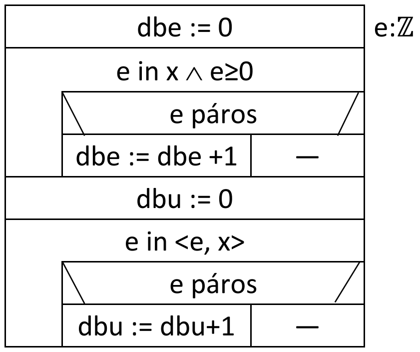
Specifikáció:

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

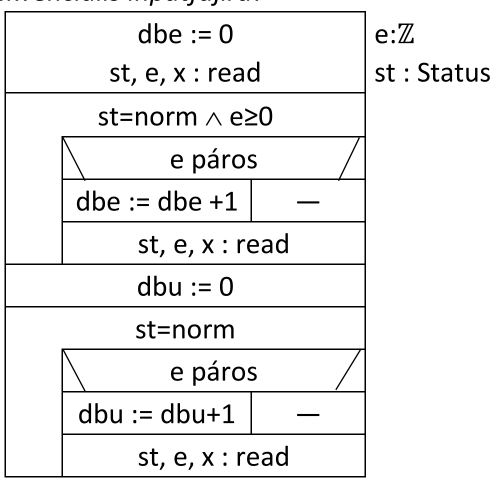
$\wedge\ dbu = \sum_{\substack{e \text{ in } <e'>\oplus x' \\ e \text{ páros}}} 1)$

Az e' az e változó értéke, x' az x felsoroló állapota az első számlálás leállásakor.
A második számlálás úgy folytatja x felsorolását, hogy ehhez figyelembe veszi az e' elemet is.

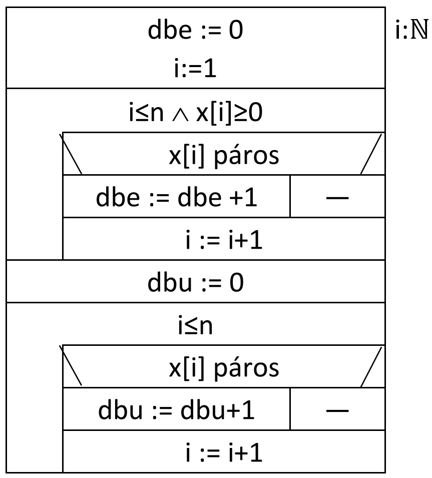
Algoritmus:



Szekvenciális inputfájlra:



Tömbre:



4. Egymás utáni napok átlaghőmérsékleteit egy szekvenciális inputfájl tartalmazza. Mennyi az első fagypont alatti értéket megelőző napok (ilyenek biztosan vannak) hőmérsékleteinek átlaga, továbbá az első fagypont alatti értéktől kezdődően vajon minden nap fagypont alatt maradt-e a hőmérséklet, és ezek között mi volt a legalacsonyabb hőmérséklet?

Specifikáció:

A = (x:infile($\mathbb{R}$), átl:$\mathbb{R}$, l:$\mathbb{L}$, min:$\mathbb{R}$)

Ef = ($x=x_0 \land |x|\geq 2 \land \exists i\in[2..|x|]: x[i]<0 \land x[1]\geq 0$)

Uf = ($(sum, e', x') = \sum_{e \text{ in } x_0}^{e\geq 0}(e)$ $\land$ $(db, e', x') = \sum_{e \text{ in } x_0}^{e\geq 0} 1$ $\land$ átl = sum/db $\land$

$\land\ (l, \_) = \forall\mathbf{SEARCH}_{e \text{ in } <e'>\oplus x'}\ (e<0)\ \land\ (min, \_) = \mathbf{MIN}_{e \text{ in } <e'>\oplus x'}\ e$)

$l = \bigwedge_{e \text{ in } <e'>\oplus x'} (e<0)$

Algoritmus:

e, sum:$\mathbb{R}$, db:$\mathbb{N}$ st:Status

Két összegzés összevonva

(H, +, 0) ~ ($\mathbb{R}$, +, 0), ($\mathbb{N}$, +, 0)
f(e) ~ e, 1
s ~ s, db
t:enor(E) ~ x:infile($\mathbb{R}$), st, e, x : read
amíg e≥0

átlagszámítás

Opt.linker helyett összeéselés (összegzés) és a minimum kiválasztás összevonva

(H, +, 0) ~ ($\mathbb{L}$, $\land$, igaz) (H, <) ~ ($\mathbb{R}$, >)
f(e) ~ e<0 f(e) ~ e
s ~ l
t:enor(E) ~ x:infile($\mathbb{R}$), st, e, x : read
folytatás előreolvasás nélkül

ciklus előtti (l, min := e, e<0) helyett, mivel az e biztosan egy negatív szám, (l, min := igaz, e) írható

```
sum, db := 0.0, 0
st, e, x : read
st=norm ∧ e≥0
    sum, db := sum+e, db +1
    st, e, x : read
átl := sum / db
l, min := igaz, e
st, e, x : read
st=norm
    l := l ∧ e<0
    e < min
        min := e    |    —
    st, e, x : read
```