ELTE | IK

PROGRAMOZÁS

4. előadás

Horváth Győző, Horváth Gyula, Szlávi Péter

Ismétlés

Feladatmegoldás lépései

1. Specifikáció
   a) Példa
   b) Bemenet, kimenet
      i. egyszerű adat?
      ii. több különböző? – rekord
      iii. több azonos? – tömb
   c) Előfeltétel
   d) Utófeltétel
2. Algoritmus
   a) Adat → változók
   b) Új halmazok → típusok
   c) Beolvasás
   d) Feldolgozás
      i. támpontok az uf-ben
      ii. végrehajtható spec.
      iii. nem, és, vagy, ->, ∀, ∃
      iv. nevezetes minták
   e) Kiírás
3. Kód

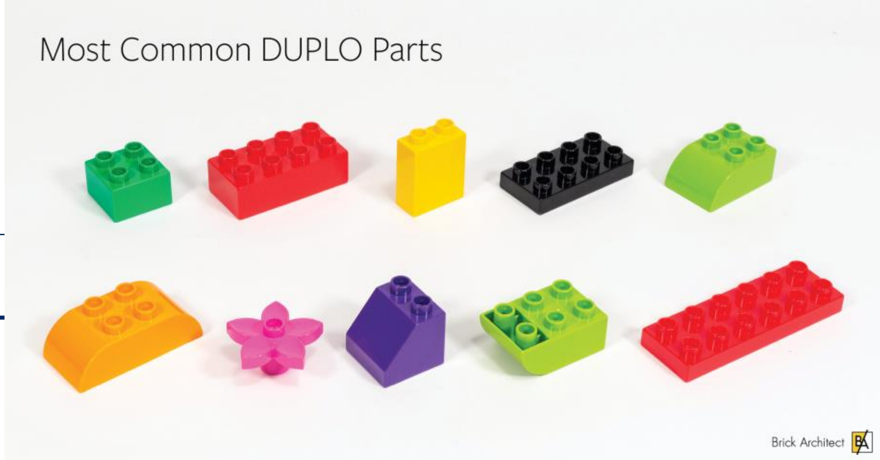
Megfeleltetések

Példa adat	Specifikáció halmaz	Algoritmus típus	Kód type
3	N	Egész	int
-3	Z	Egész	int
3,3	R	Valós	double
igaz	L	Logikai	bool
"alma"	S	Szöveg	string
"a"	K	Karakter	char
(név:"Győző", jegy: 5)	Név x Jegy, S x N	Rekord	struct
[3, 5, -6, 2]	Z[1..n]	Tömb	int[]

Analóg programozás – visszavezetés

- Visszavezetés
  - Konkrét feladat felírása
  - Összevetés a minta sablonjával
  - Különbségek felírása egy táblázatba
  - Különbségek alkalmazása a sablon algoritmusában
  - → Konkrét feladat algoritmusa

Programozási minták



1. Összegzés
2. Megszámolás
3. Maximumkiválasztás
4. Feltételes maximumkeresés

szummás, mindenes feladat
↓
számlálós ciklus

5. Keresés
6. Eldöntés
7. Kiválasztás

létezikes feladat
↓
feltételes ciklus

Másolás függvényszámítás

Másolás

Feladatok:

- Egy **számsorozat tagjai**nak adjuk meg az abszolút értékét!
- Egy szöveget alakítsunk át **csupa** kisbetűssé!
- **Számoljuk ki** két vektor összegét!
- **Készítsünk** függvény**táblázat**ot a sin(x) függvényről!
- **Ismerünk N** dátumot 'éé.hh.nn' alakban, adjuk meg **őket** 'éé. hónapnév nn.' alakban!

Mi bennük a közös?

n darab „valamihez" kell hozzárendelni másik n darab „valamit", ami akár az előbbitől különböző típusú is lehet.
A darabszám, a sorrend is marad.
Az elemeken operáló függvény ugyanaz.

Példa – abszolút értékek
algoritmikus gondolkodással

	x		y
1	3,3	→	3,3
2	-5,8	→	5,8
3	4,5	→	4,5
4	-2,2	→	2,2

Feladat:

Egy **számsorozat tagjai**nak adjuk meg az abszolút értékét!

Specifikáció:

```
Be: n∈N, x∈R[1..n]
Ki: y∈R[1..n]
Ef: -
Uf: ∀i∈[1..n]:(y[i]=abs(x[i]))
```

Algoritmus:

```
i=1..n
    y[i]:=abs(x[i])
```

Másolás
sablon

i		y
e	→	f(e)
e+1	→	f(e+1)
…	→	…
u	→	f(u)

Feladat

Adott az egész számok egy [e..u] intervalluma és egy f:[e..u]→H függvény. Rendeljük az [e..u] intervallum minden értékéhez az f függvény hozzá tartozó értékét!

Specifikáció

```
Be: e∈Z, u∈Z
Ki: y∈H[1..u-e+1]
Ef: -
Uf: ∀i∈[e..u]:(y[i-e+1]=f(i))
```

Rövidítve:

```
Uf: y=MÁSOL(i=e..u, f(i))
```

Algoritmus

```
i=e..u
    y[i-e+1]:=f(i)
```

Példa – két vektor összege
visszavezetés

Számoljuk ki két vektor összegét!

Feladatsablon
(mintafeladat)

```
Be: e∈Z, u∈Z
Ki: y∈H[1..u-e+1]
Ef: -
Uf: y=MÁSOL(i=e..u, f(i))
```

Két vektor összege
(konkrét feladat)

```
Be: n∈N, p∈R[1..n], q∈R[1..n]
Ki: r∈R[1..n]
Ef: -
Uf: r=MÁSOL(i=1..n,p[i]+q[i])
```

Visszavezetés:

```
y     ~ r
e..u  ~ 1..n
f(i)  ~ p[i]+q[i]
```

Algoritmus:

```
i=e..u
    y[i-e+1]:=f(i)
```

```
i=1..n
    r[i]:=p[i]+q[i]
```

Kiválogatás

Kiválogatás

Feladatok:

- **Adjuk** meg egy osztály kitűnő tanuló**it**!
- **Adjuk** meg egy természetes szám **összes** osztóját!
- **Adjuk** meg egy mondat magas hangrendű szava**it**!
- **Adjuk** meg emberek egy halmazából a 180 cm feletti**eket**!
- **Adjuk** meg egy év azon napja**it**, amikor délben nem fagyott!
- **Soroljuk** föl egy szó magánhangzó**it**!

Mi bennük a közös?

n darab „valami” közül kell megadni az összes, adott T tulajdonsággal rendelkezőt!

Példa – nem fagyos napok
algoritmikus gondolkodással

	hőm
1	-2,2
2	1,5
3	2,8
n=4	-1,0

→

	poz
1	2
db=2	3

Feladat:

Adjuk meg egy év azon napja**it**, amikor délben nem fagyott!

Specifikáció:

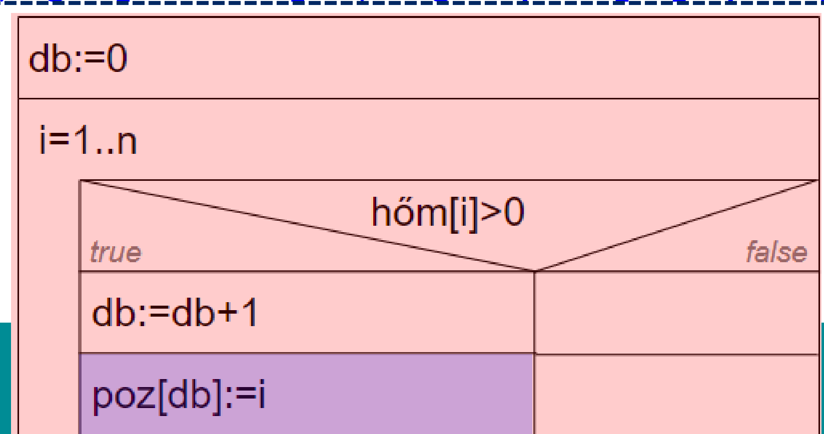
megszámolás

a kiválogatott indexekhez tartozó értékek pozitívak

nincs két egyforma index a poz tömbben, részsorozata az indextartományának

```
Be: n∈N, hőm∈R[1..n]
Ki: db∈N, poz∈N[1..db]
Ef: ∀i∈[1..n]:(-100<=hőm[i]<=100)
Uf: db=DARAB(i=1..n,hőm[i]>0) és
    ∀i∈[1..db]:(hőm[poz[i]]>0) és
    ∀i∈[1..db]:(∀j∈[1..db]:(i<j->poz[i]<poz[j]))
```

Algoritmus:



Példa – nem fagyos napok
algoritmikus gondolkodással

	hőm			poz
1	-2,2		1	2
2	1,5	→	db=2	3
3	2,8			
n=4	-1,0			

Feladat:

Adjuk meg egy év azon napja**it**, amikor délben nem fagyott!

Specifikáció:

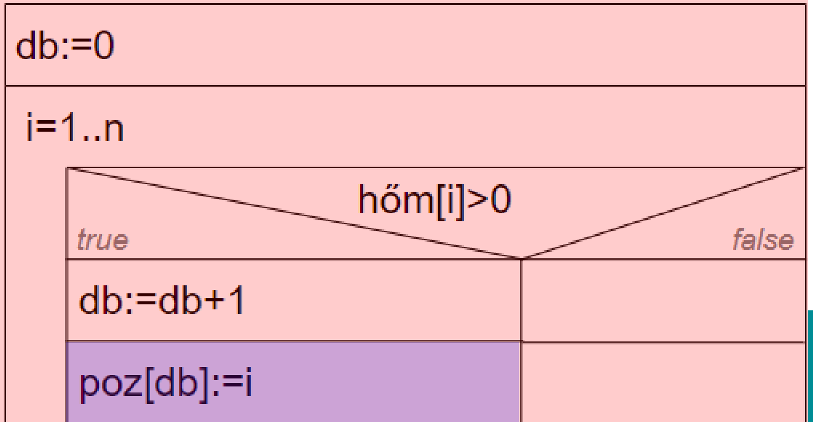
```
Be: n∈N, hőm∈R[1..n]
Ki: db∈N, poz∈N[1..db]
Ef: ∀i∈[1..n]:(-100<=hőm[i]<=100)
Uf: db=DARAB(i=1..n,hőm[i]>0) és
    ∀i∈[1..db]:(hőm[poz[i]]>0) és
    poz⊆[1..n]
```

megszámolás

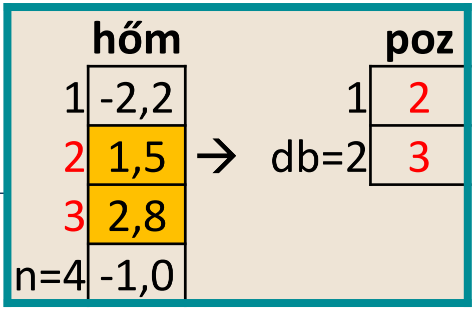
a kiválogatott indexekhez tartozó értékek pozitívak

nincs két egyforma index a poz tömbben

Algoritmus:



Példa – nem fagyos napok
algoritmikus gondolkodással



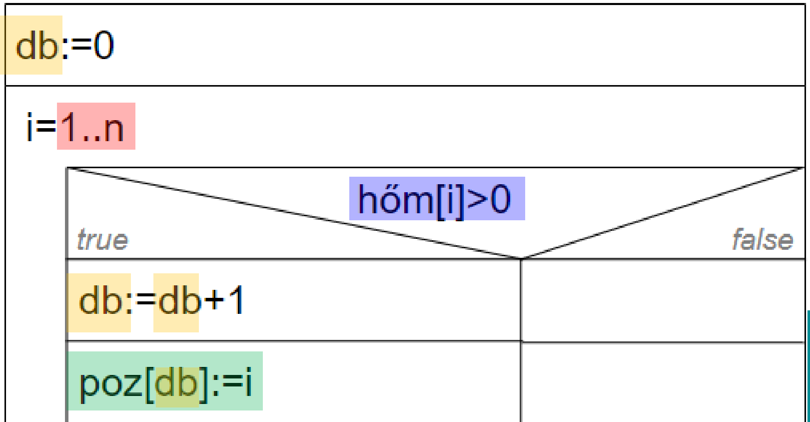
Feladat:

Adjuk meg egy év azon napja**it**, amikor délben nem fagyott!

Specifikáció:

```
Be: n∈N, hőm∈R[1..n]
Ki: db∈N, poz∈N[1..db]
Ef: ∀i∈[1..n]:(-100<=hőm[i]<=100)
Uf: db=DARAB(i=1..n,hőm[i]>0) és
    ∀i∈[1..db]:(hőm[poz[i]]>0) és
    poz⊆[1..n]
```

Algoritmus:



Példa – kitűnő tanulók
algoritmikus gondolkodással

	diákok: név	diákok: jegy			jelesek
1	P	4		1	F
2	F	5	→	db=2	G
3	E	2			
n=4	G	5			

Feladat:

Adjuk meg egy osztály kitűnő tanuló**it**!

Specifikáció és algoritmus:

megszámolás

```
Be: n∈N, diákok∈Diák[1..n], Diák=Név x Jegy, Név=S, Jegy=N
Ki: db∈N, jelesek∈S[1..db]
Ef: ∀i∈[1..n]:(1<=diákok[i].jegy<=5)
Uf: db=DARAB(i=1..n,diákok[i].jegy=5) és
    ∀i∈[1..db]:(∃j∈[1..n]:(
      diákok[j].jegy=5 és jelesek[i]=diákok[j].név)) és
    ∀i∈[1..db]:(∀j∈[1..db]:(i<j->
      ∃ii∈[1..n]:(∃jj∈[1..n]:(
        ii<jj és
        diákok[ii].név=jelesek[i]
        diákok[jj].név=jelesek[j]
      ))
    ))
```

a kiválogatott nevek olyan indexhez tartoznak, ahol a jegy 5-ös

a kiválogatott nevek különböző indexekhez tartoznak, sőt részsorozat!

```
db:=0
i=1..n
  diákok[i].jegy=5
  true:
    db:=db+1
    jelesek[db]:=diákok[i].név
  false:
```

Példa – kitűnő tanulók
algoritmikus gondolkodással

	diákok név	diákok jegy
1	P	4
2	F	5
3	E	2
n=4	G	5

→

jelesek	
1	F
db=2	G

Variáció:

Részsorozat rövidítve

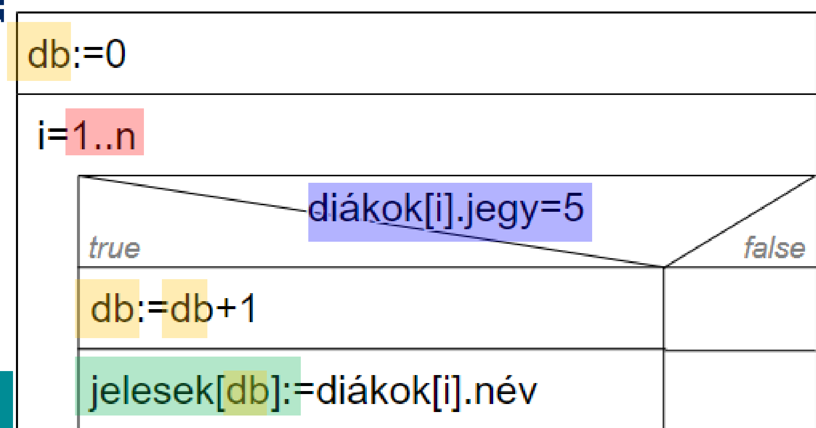
Specifikáció és algoritmus:

```
Be: n∈N, diákok∈Diák[1..n], Diák=Név x Jegy, Név=S, Jegy=N
Ki: db∈N, jelesek∈S[1..db]
Ef: ∀i∈[1..n]:(1<=diákok[i].jegy<=5)
Uf: db=DARAB(i=1..n,diákok[i].jegy=5) és
    ∀i∈[1..db]:(∃j∈[1..n]:(
      diákok[j].jegy=5 és jelesek[i]=diákok[j].név)) és
    jelesek⊆diákok.név
```

megszámolás

a kiválogatott nevek olyan indexhez tartoznak, ahol a jegy 5-ös

a kiválogatott nevek különböző indexekhez tartoznak, sőt részsorozat!



Példa – kitűnő tanulók
algoritmikus gondolkodással

diákok

	név	jegy
1	P	4
2	F	5
3	E	2
n=4	G	5

→ db=2

jelesek

1	F
db=2	G

Variáció:

Részsorozat rövidítve

Specifikáció és algoritmus:

```
Be: n∈N, diákok∈Diák[1..n], Diák=Név x Jegy, Név=S, Jegy=N
Ki: db∈N, jelesek∈S[1..db]
Ef: ∀i∈[1..n]:(1<=diákok[i].jegy<=5)
Uf: db=DARAB(i=1..n,diákok[i].jegy=5) és
    ∀i∈[1..db]:(∃j∈[1..n]:(
      diákok[j].jegy=5 és jelesek[i]=diákok[j].név)) és
    jelesek⊆diákok.név
```

megszámolás

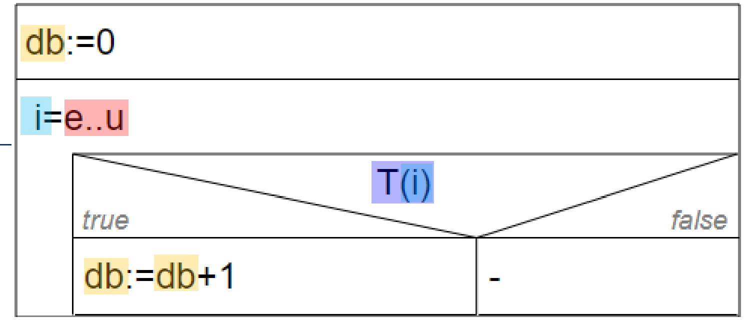
a kiválogatott nevek olyan indexhez tartoznak, ahol a jegy 5-ös

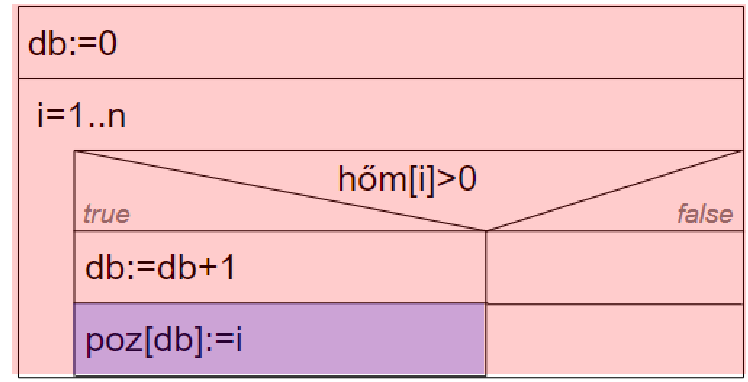
a kiválogatott nevek különböző indexekhez tartoznak

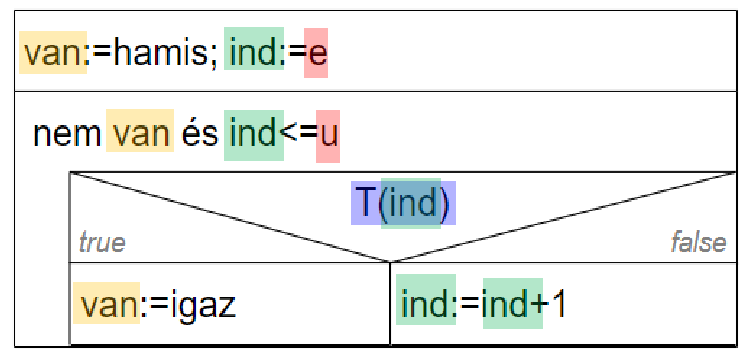
```
db:=0
i=1..n
  diákok[i].jegy=5
  true                              false
    db:=db+1
    jelesek[db]:=diákok[i].név
```

Tanulságok

- A kiválogatás hasonlít:
  - Megszámolásra
    - Hányszor teljesült a T tulajdonság?
    - + mely esetekben?
  - Keresésre
    - Ha teljesült, akkor hol a T tulajdonság?
    - + mindenkire



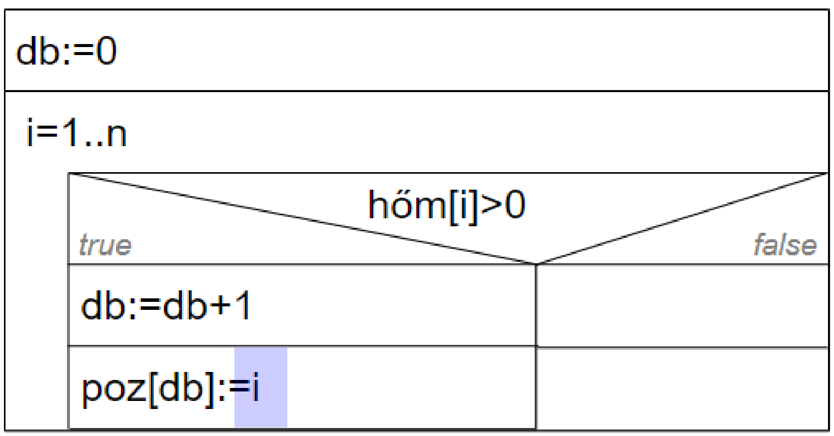


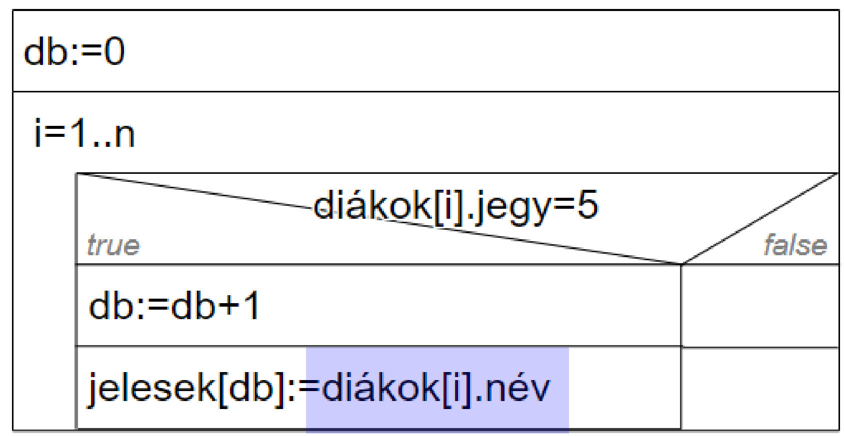


Tanulságok

	diákok			jeles			jelesek
	név	jegy	→		indexek	→	
1	P	4		1	2	1	F
2	F	5		db=2	4	db=2	G
3	E	2					
n=4	G	5					

- Eredményképpen
  - intervallum elemei, vagy
  - ezekhez rendelt értékek





Kiválogatás
sablon

i		T(i)			f(i)
e	→	HAMIS			
e+1	→	IGAZ	→	1	f(e+1)
e+2	→	IGAZ	→	2	f(e+2)
u	→	HAMIS			

Feladat

Adott az egész számok egy [e..u] intervalluma, egy ezen értelmezett T:[e..u]→Logikai feltétel és egy f:[e..u]→H függvény. Határozzuk meg az f függvény az [e..u] intervallum azon értékeinél felvett értékeit, amelyekre a T feltétel teljesül!

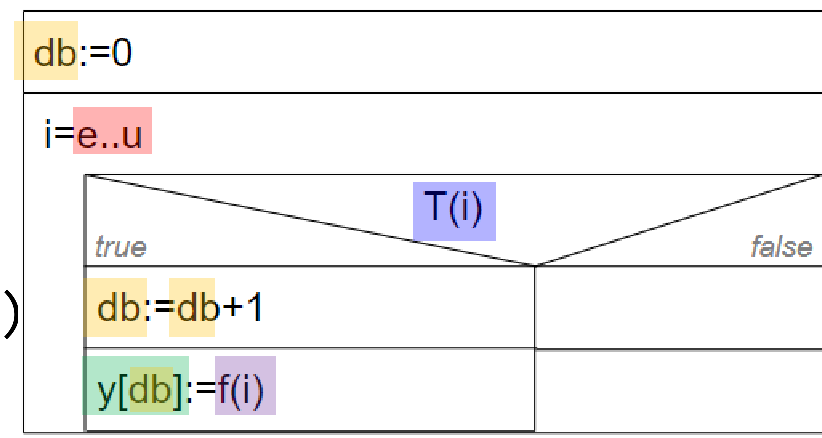
Specifikáció

```
Be: e∈Z, u∈Z
Ki: db∈N, y∈H[1..db]
Ef: -
Uf: db=DARAB(i=e..u,T(i)) és
    ∀i∈[1..db]:(
      ∃j∈[e..u]:T(j) és y[i]=f(j))
 és y⊆(f(e),f(e+1),...,f(u))
```

Rövidítve:

```
Uf: (db,y)=KIVÁLOGAT(i=e..u,T(i),f(i))
```

Algoritmus



f és T között azáltal lesz kapcsolat, hogy a bemenet ugyanazon elemére hivatkoznak

Példa – összes osztó
visszavezetés

Adjuk meg egy természetes szám összes osztóját!

Feladatsablon
(mintafeladat)

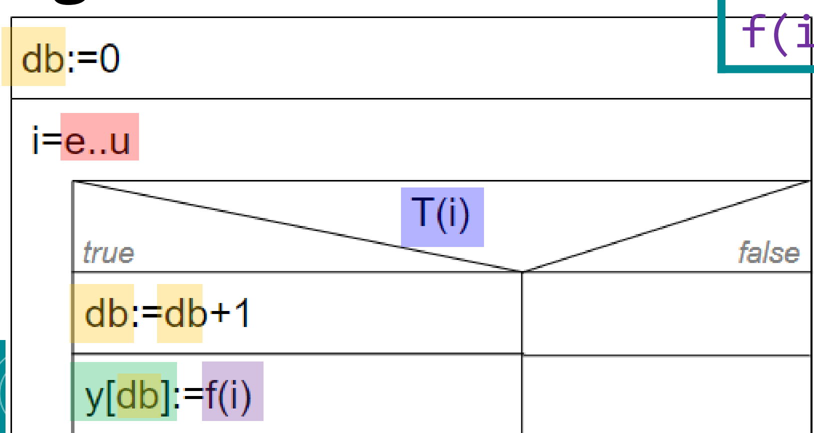
```
Be: e∈Z, u∈Z
Ki: db∈N, y∈H[1..db]
Ef: -
Uf: (db,y)=
 KIVÁLOGAT(i=e..u,T(i),f(i))
```

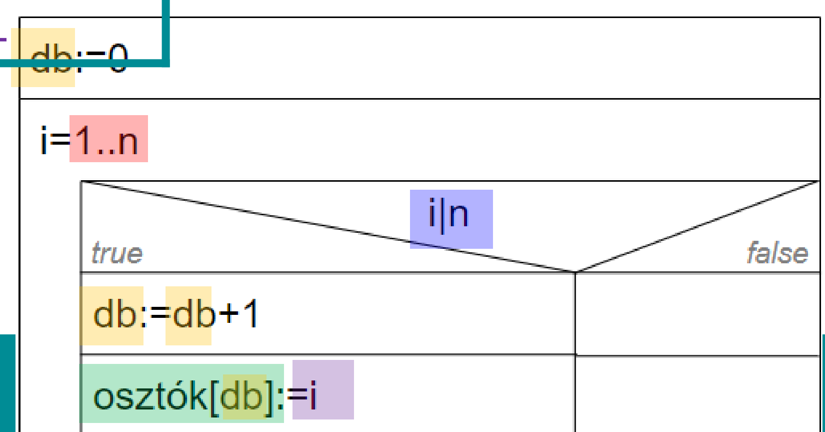
Összes osztó
(konkrét feladat)

```
Be: n∈N
Ki: db∈N, osztók∈N[1..db]
Ef: -
Uf: (db,osztók)=
 KIVÁLOGAT(i=1..n,i|n,i)
```

```
y     ~ osztók
e..u  ~ 1..n
T(i)  ~ i|n
f(i)  ~ i
```

Visszavezetés:
Algoritmus:





Példa – kitűnő tanulók
visszavezetés

Adjuk meg egy osztály kitűnő tanulóit!

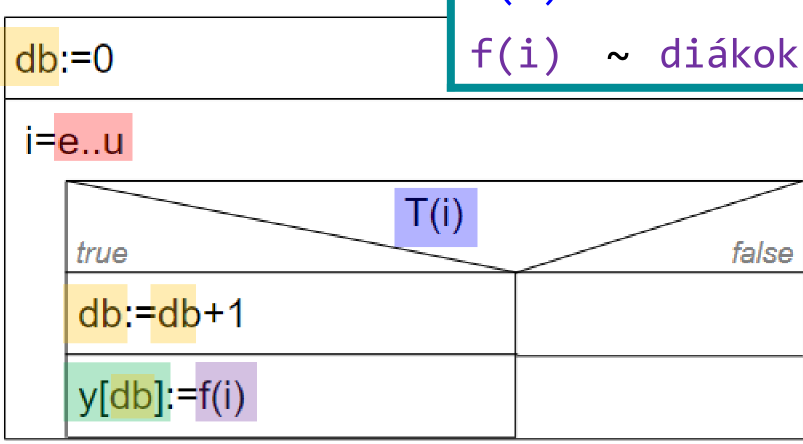
Feladatsablon

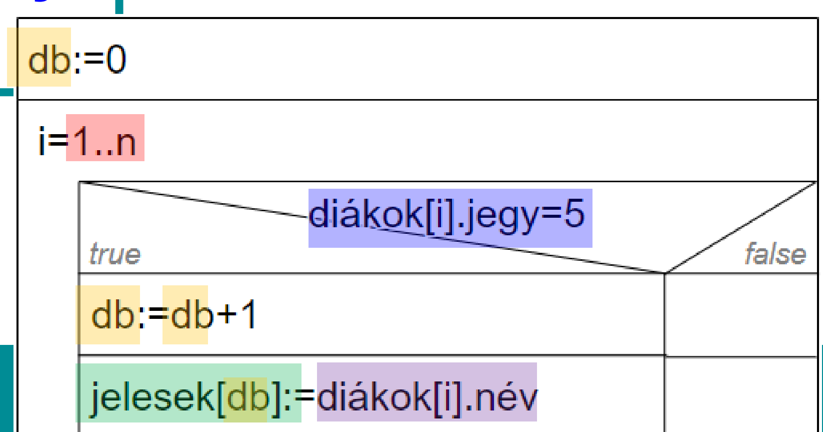
```
Be: e∈Z, u∈Z

Ki: db∈N, y∈H[1..db]
Ef: -
Uf: (db,y)=
       KIVÁLOGAT(i=e..u,
         T(i),
           f(i))
```

Kitűnő tanulók

```
Be: n∈N, diákok∈Diák[1..n],
    Diák=Név x Jegy, Név=S,Jegy=N
Ki: db∈N, jelesek∈S[1..db]
Ef: -
Uf: (db,jelesek)=
       KIVÁLOGAT(i=1..n,
         diákok[i].jegy=5,
           diákok[i].név)
```

```
y      ~ osztók
e..u   ~ 1..n
T(i)   ~ diákok[i].jegy=5
f(i)   ~ diákok[i].név
```





Gyakori programozási minta változatok

Minimumkiválasztás
példa – specifikáció

minimumkiválasztás

Feladat:

Adjuk meg egy adott f:[e..u]→H függvény értékei között a legkisebb elemet!

Specifikáció:

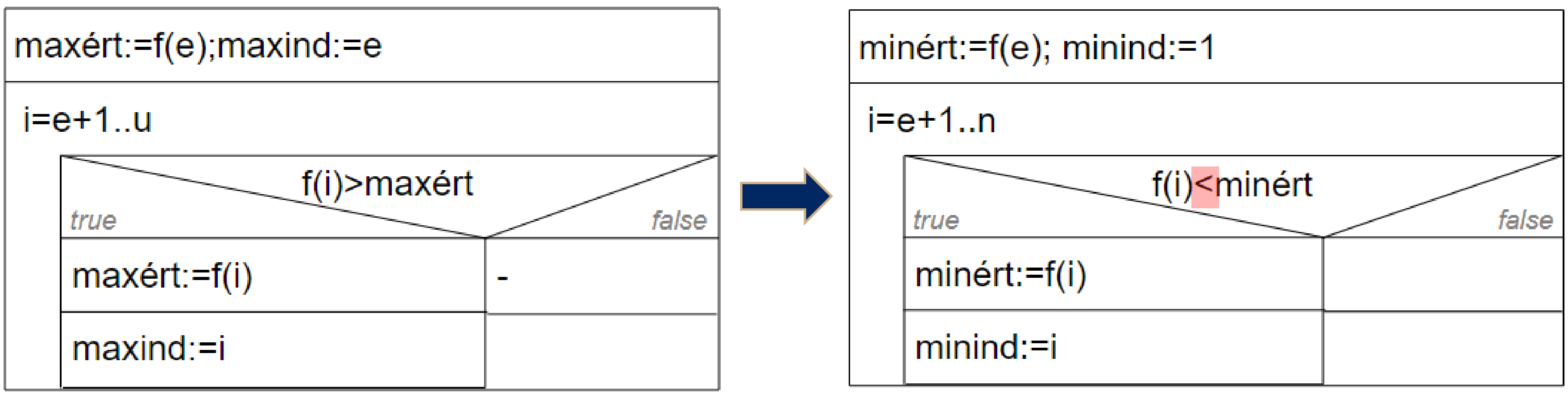
egyetlen különbség a maximumkiválasztáshoz képest az elnevezések mellett

```
Be: e∈Z, u∈Z
Ki: minind∈Z, minért∈H
Ef: e<=u
Uf: minind∈[e..u] és
    ∀i∈[e..u]:(f(minind)<=f(i)) és
    minért=f(minind)
Uf: (minind,minért)=MIN(i=e..u,f(i))
```

Minimumkiválasztás
algoritmus – analóg algoritmikus gondolkodással

Algoritmus:

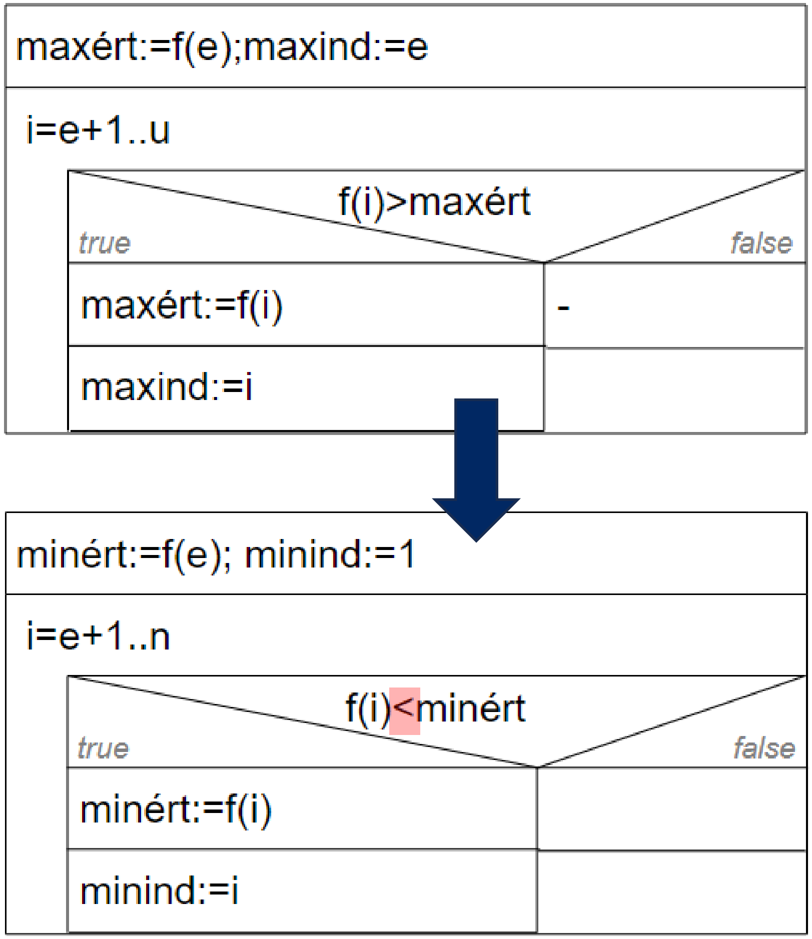
másik reláció használata



Minimumkiválasztás
algoritmus – visszavezetéssel

Algoritmus:

- ≤ „felüldefíniálása”
- eddig csak azt használtuk ki, hogy ez teljes rendezési reláció
- jelentése nem volt érdekes
- helyettesíthető bármilyen teljes rendezési relációval
- azaz ≤(a,b)=a≥b



Minimumkiválasztás
algoritmus – visszavezetéssel

Algoritmus:

- ötlet: alakítsuk át a feladatot!
- a függvényértékek „**ellentettjei**” között keressünk maximumot!
- a megtalált maximum a keresett minimum „**ellentettje**” lesz

Maximumkiválasztás

```
Be: e∈Z, u∈Z
Ki: maxind∈Z, maxért∈H
Ef: e<=u
Uf: (maxind,maxért)=
                   MAX(i=e..u,f(i))
```

Minimumkiválasztás

```
Be: e∈Z, u∈Z
Ki: minind∈Z, minért
Ef: e<=u
Uf: (minind,-minért)=
                   MAX(i=e..u,-f(i))
```

Ötlet, ha H számhalmaz!

Visszavezetés:

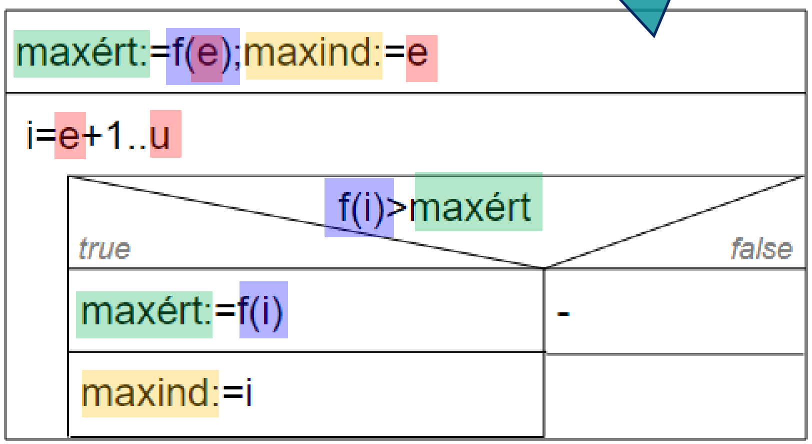
```
maxind, maxért   ~   minind, -minért
f(i)             ~   -f(i)
```

Minimumkiválasztás algoritmus – visszavezetéssel

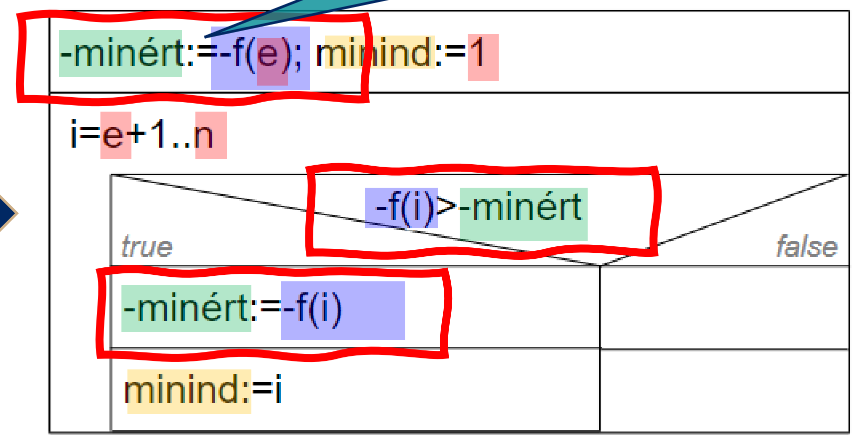
```
maxind,maxért  ~  minind,-minért
f(i)           ~  -f(i)
```

maximumkiválasztás sablonja

Algoritmus:

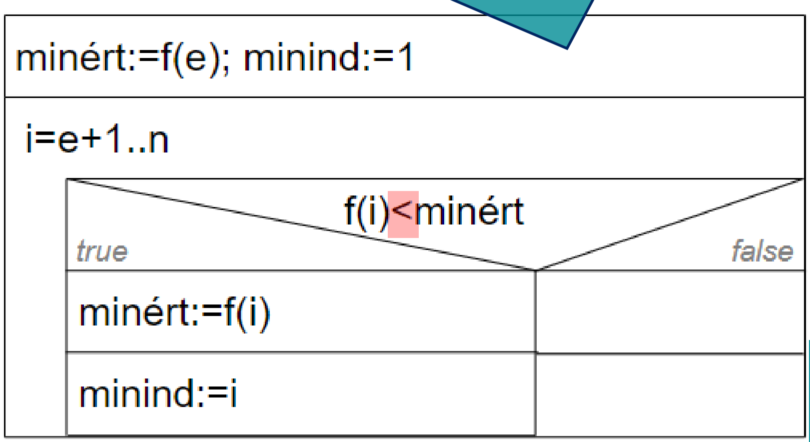


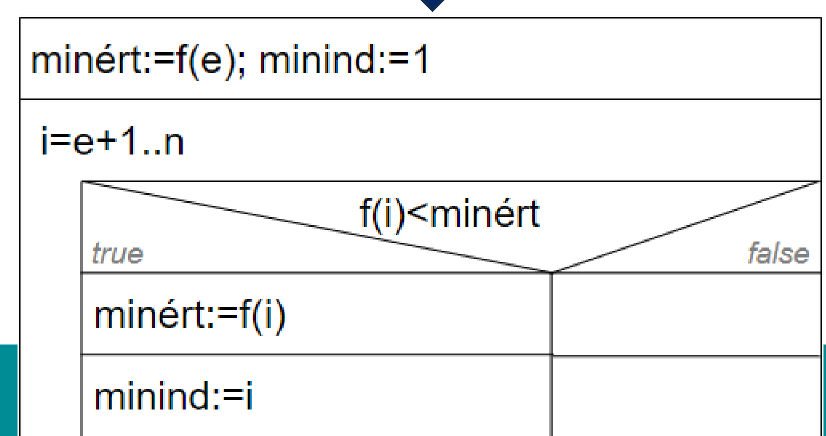
nem megengedett értékadás



szorozzunk -1-gyel a szükséges helyeken!

algoritmikus gondolkodással





Minimumkiválasztás
sablon

Feladat

Adott az egész számok egy [e..u] intervalluma és egy f:[e..u]→H függvény. A H halmaz elemein értelmezett egy teljes rendezési reláció. Határozzuk meg, hogy az f függvény hol veszi fel az [e..u] nem üres intervallumon a legkisebb értéket, és mondjuk meg, mekkora ez a minimális érték!

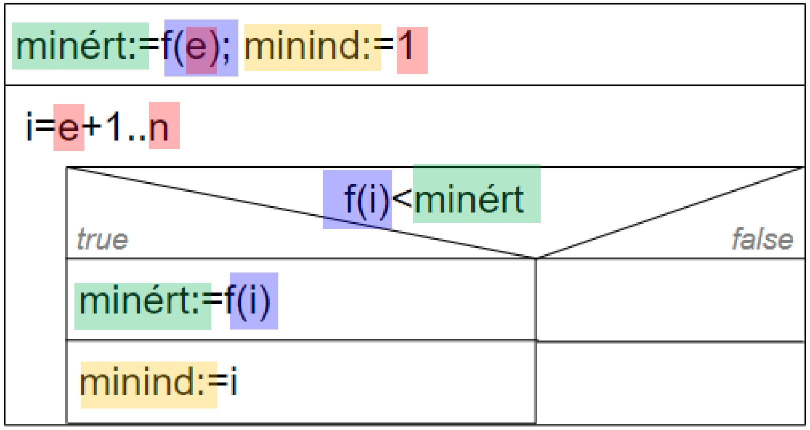
Specifikáció

```
Be: e∈Z, u∈Z
Ki: minind∈Z, minért∈H
Ef: e<=u
Uf: minind∈[e..u] és
    ∀i∈[e..u]:(f(minind)<=f(i)) és
    minért=f(minind)
```

Rövidítve:

```
Uf: (minind,minért)=MIN(i=e..u,f(i))
```

Algoritmus



Hátulról keresés
példa

Feladat:

Keressük meg a legutolsó T tulajdonságú elemet!

Specifikáció:

```
Be: e∈Z, u∈Z
Ki: van∈L, ind∈Z
Ef: -
Uf: van=∃i∈[e..u]:(T(i)) és
    van->(ind∈[e..u] és T(ind) és
          ∀i∈[ind+1..u]:(nem T(i)))
```

vö. az elölről keresésnél: [e..ind-1]

Hátulról keresés
algoritmus – algoritmikus gondolkodással

Algoritmus:

induljunk hátulról, lépegessünk előre, amíg az intervallumon belül vagyunk és nem találtunk T tulajdonságú elemet

```
ind:=e
ind<=u és nem T(ind)
    ind:=ind+1
van:=ind<=u
```

```
ind:=u
ind>=e és nem T(ind)
    ind:=ind-1
van:=ind>=e
```

Hátulról keresés
algoritmus – visszavezetéssel

Algoritmus:

- ötlet: alakítsuk át a feladatot!
- az [e..u] intervallumot tükrözzük az origóra, és az így kapott [-u..-e] intervallumban keressük az elsőt adott tulajdonságú elemet
- a talált ind helyett a –ind lesz az eredmény

Elölről keresés sablonja

```
Be: e∈Z, u∈Z
Ki: van∈L, ind∈Z
Ef: -
Uf: (van,ind)=KERES(i=e..u,T(i))
```

Hátulról keresés

```
Be: e∈Z, u∈Z
Ki: van∈L, ind∈Z
Ef: -
Uf: (van,-ind)=KERES(i=-u..-e,T(i))
```

Visszavezetés:

```
ind    ~  -ind
e..u   ~  -u..-e
```

Hátulról keresés algoritmus – visszavezetéssel

keresés sablonja

ind:=e
ind<=u és nem T(ind)
ind:=ind+1
van:=ind<=u

e..u ~ -u..-e

ind:=-u
ind<=-e és nem T(-ind)
ind:=ind+1
van:=ind<=-e

ind ~ -ind

nem megengedett értékadás

-ind:=-u
-ind<=-e és nem T(ind)
-ind:=-ind+1
van:=-ind<=-e

ind ~ -ind

e..u ~ -u..-e

szorozzunk -1-gyel a szükséges helyeken!

ind:=u
ind>=e és nem T(ind)
ind:=ind-1
van:=ind>=e

algoritmikus gondolkodással

ind:=u
ind>=e és nem T(ind)
ind:=ind-1
van:=ind>=e

Mind eldöntés
példa

Feladat:

Adjuk meg, hogy egy [e..u] intervallum minden eleme rendelkezik-e egy adott T tulajdonsággal!

Specifikáció:

eldöntésben: ∃i∈[e..u]:(T(i))

```
Be: e∈Z, u∈Z
Ki: mind∈L
Ef: -
Uf: mind=∀i∈[e..u]:(T(i))
```

Rövidítve:

```
Uf: mind=MIND(i=e..u,T(i))
```

Mind eldöntés
algoritmus – algoritmikus gondolkodással

Algoritmus:

Addig lépkedjünk előre, amíg jót találunk. Ha lelépünk az intervallumról, akkor mindegyik jó volt.

```
i:=e
i<=u és nem T(i)
    i:=i+1
van:=i<=u
```

```
i:=e
i<=u és T(i)
    i:=i+1
mind:=i>u
```

Mind eldöntés
algoritmus – visszavezetéssel

Algoritmus:

- ötlet: alakítsuk át a feladatot!
- van-e olyan elem, ami nem rendelkezik a T tulajdonsággal?
- a válasz negáltja lesz a keresett eredmény

Van-e olyan?

```
Be: e∈Z, u∈Z
Ki: van∈L
Ef: -
Uf: van=VAN(i=e..u,T(i))
```

Mind olyan-e?

```
Be: e∈Z, u∈Z
Ki: mind∈L
Ef: -
Uf: mind=nem VAN(i=e..u,nem T(i))
Uf: nem mind=VAN(i=e..u,nem T(i))
```

Visszavezetés:

```
van      ~   nem mind
T(i)     ~   nem T(i)
```

Mind eldöntés
algoritmus – visszavezetéssel

van	~	nem mind
T(i)	~	nem T(i)

eldöntés sablonja

```
i:=e
i<=u és nem T(i)
    i:=i+1
van:=i<=u
```

nem megengedett értékadás

```
i:=e
i<=u és nem nem T(i)
    i:=i+1
nem mind:=i<=u
```

negálunk a szükséges helyeken

algoritmikus gondolkodással

```
i:=e
i<=u és T(i)
    i:=i+1
mind:=i>u
```

```
i:=e
i<=u és T(i)
    i:=i+1
mind:=i>u
```

Mind eldöntés (vagy optimista eldöntés) sablon

Feladat

Adott az egész számok egy [e..u] intervalluma és egy T:[e..u]→Logikai feltétel. Határozzuk meg, hogy az [e..u] intervallumnak mindegyik eleme olyan-e, amely kielégíti a T feltételt!

Specifikáció

```
Be: e∈Z, u∈Z
Ki: mind∈L
Ef: -
Uf: mind=∀i∈[e..u]:(T(i))
```

Rövidítve:

```
Uf: mind=MIND(i=e..u,T(i))
```

Algoritmus

```
i:=e
i<=u és T(i)
    i:=i+1
mind:=i>u
```

Futóindex mint sablonrész

Példa – legmelegebb vasárnap

visszavezetés

Hétfői naptól kezdve mértük a déli hőmérsékletet.
Add meg a legmelegebb vasárnapot!

Feladatsablon

Be: e∈Z, u∈Z

Ki: maxind∈Z, maxért∈H

Ef: e<=u

Uf: (maxind,maxért)=
MAX(i=e..u,f(i))

Legmelegebb vasárnap

Be: n∈N, hőm∈R[1..n]

Ki: lmv∈N, mh∈R

Ef: n>=7

Uf: (lmv,mh)=
MAX(v=1..n div 7,hőm[v*7])

Visszavezetés:

maxind,maxért	~	lmv,mh
i	~	v
e..u	~	1..n div 7
f(i)	~	hőm[v*7]

Algoritmus:

maxért:=f(e);maxind:=e
i=e+1..u
f(i)>maxért
true
false
maxért:=f(i)
-
maxind:=i

mh:=hőm[1*7]; lmv:=1
v=2..n div 7
hőm[v*7]>mh
true
mh:=hőm[v*7]
lmv:=v

	i	hőm
	1	23,3
	2	20,1
	3	
	4	
	5	
	6	
e	7	23,3
	8	
	9	
	10	
	11	
	12	
	13	
e+1	14	12,7
	15	
	16	
	17	
	18	
	19	
	20	
e+2	21	25,6
	22	
	23	

Összefoglalás

Programozási minták

1. Összegzés
2. Megszámolás
3. Maximumkiválasztás
   a. Minimumkiválasztás
4. Feltételes maximumkeresés
5. Keresés
6. Eldöntés
   a. Mind eldöntés
7. Kiválasztás
8. Másolás
9. Kiválogatás
   a. Értékek