ELTE | IK

PROGRAMOZÁS

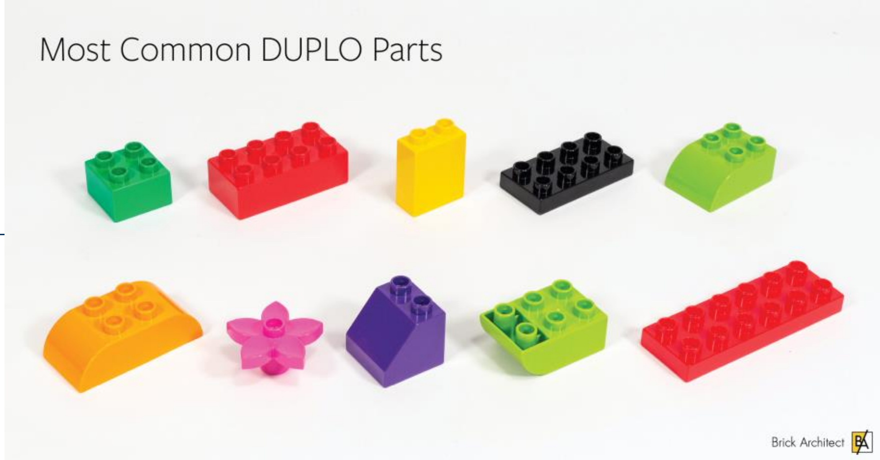
6. előadás

Horváth Győző, Horváth Gyula, Szlávi Péter

Ismétlés

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



Függvények

- Függvények szerepe
  - Részfeladatok csoportosítása (alprogram)
  - Általánosítás (paraméterekkel)

Programtranszformációk

Cél, szerkezet…

Az algoritmus ekvivalens átalakítása, melynek célja

- hatékonyabbra írás
- egyszerűsítés
- megvalósíthatóság

Vö. Programozási tétel szerkezetével!

- specifikáció
- algoritmus

Szerkezete:

- *algoritmus*$_1$, *algoritmus*$_2$
- *feltétel*

Vö. Programozási tétel állításával!

Ha a specifikáció beli Ef a bemeneti adatokra teljesül, akkor az algoritmus végrehajtása után az Uf teljesül.

Állítás:

Ha *feltétel* teljesül, akkor

$$\textit{algoritmus}_1 \approx_{\text{szemantikusan}} \textit{algoritmus}_2$$

Bevezető példa

Maximumkiválasztás:
$<_{\text{Távolság}}$

Melyik az origótól legmesszebb levő mP pont ($p \in Pont[1..n]$, $Pont = X \times Y$)

```
Be: e∈Z, u∈Z

Ki: maxind∈Z, maxért∈H
Ef: e<=u
Uf: maxind∈[e..u] és
    ∀i∈[e..u]:(f(maxind)>=f(i)) és
    maxért=f(maxind)
```

Rövidítve:

```
Uf: (maxind,maxért)=
                    MAX(i=e..u,f(i))
```

```
Be: n∈N, p∈Pont[1..n],
    Pont=X x Y, X,Y=R
Ki: mP∈N
Ef: n>0
Uf: mP∈[1..n] és
    ∀i∈[1..n]:(
```

$$\sqrt{p[mP].x^2 + p[mP].y^2} >= \sqrt{p[i].x^2 + p[i].y^2}\ ) \text{ és}$$

$$maxért=\sqrt{p[mP].x^2 + p[mP].y^2}$$

Rövidítve:

$$Uf: (mP,)= MAX(i=1..n,\sqrt{p[i].x^2 + p[i].y^2})$$

Bevezető példa

Be: e∈Z, u∈Z

Ki: maxind∈Z, maxért∈H

Ef: e<=u

Uf: maxind∈[e..u] és
∀i∈[e..u]:(f(maxind)>=f(i)) és
maxért=f(maxind)

Rövidítve:

Uf: (maxind,maxért)=
MAX(i=e..u,f(i))

Be: n∈N, p∈Pont[1..n],
Pont=X x Y, X,Y=R

Ki: mP∈N

Ef: n>0

Uf: mP∈[1..n] és
∀i∈[1..n]:(
$\sqrt{p[mP].x^2 + p[mP].y^2}$ >=
$\sqrt{p[i].x^2 + p[i].y^2}$) és
maxért=$\sqrt{p[mP].x^2 + p[mP].y^2}$

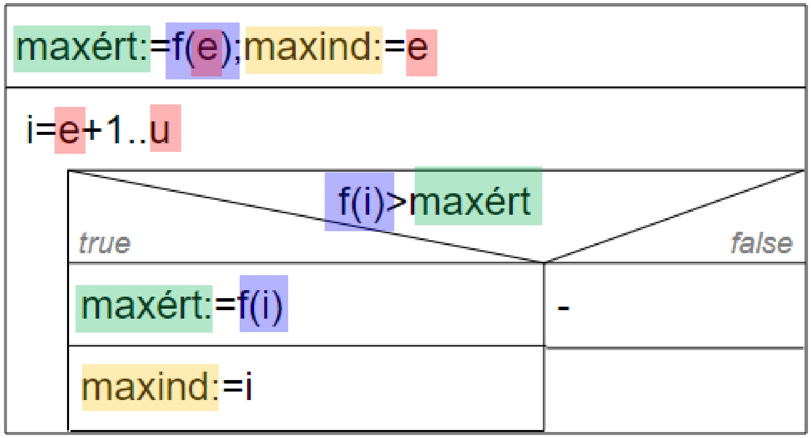
Rövidítve:

Uf: (mP,)=
MAX(i=1..n,$\sqrt{p[i].x^2 + p[i].y^2}$)

Visszavezetés:

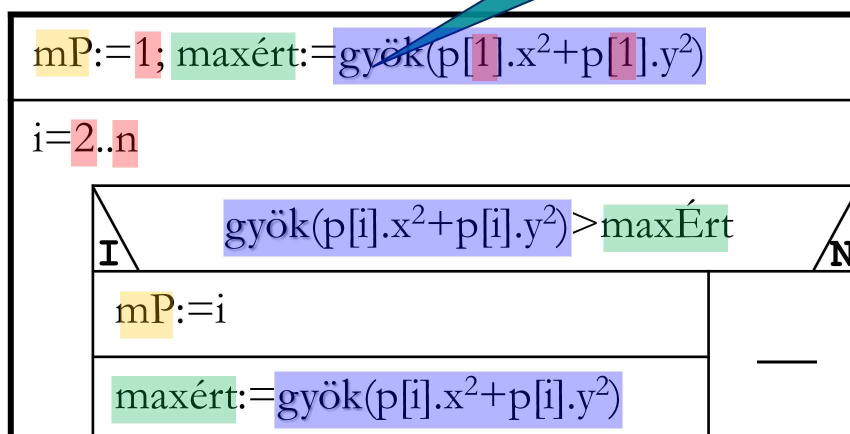
maxind, maxért	~	mP, maxért
e..u	~	1..n
f(i)	~	$\sqrt{p[i].x^2 + p[i].y^2}$

Bevezető példa



maxind, maxért	~	mP, maxért
e..u	~	1..n
f(i)	~	$\sqrt{p[i].x^2 + p[i].y^2}$

Négyzetgyök függvény



Változó
i:**Egész**
maxért:**Valós**

Bevezető példa

Változó
i:**Egész**
maxért:**Valós**

mP:=1; maxért:=gyök(p[1].x^2+p[1].y^2)

i=2..n

gyök(p[i].x^2+p[i].y^2)>maxért

I	N
mP:=i maxért:=gyök(p[i].x^2+p[i].y^2)	—

A gyök függvényt tartalmazó kifejezéseket emeljük négyzetre!

Bevezető példa

Változó
i:**Egész**
maxért:**Valós**

mP:=1; maxért:=$\text{gyök}(p[1].x^2+p[1].y^2)^2$

i=2..n

I	$\text{gyök}(p[i].x^2+p[i].y^2)^2$>maxért	N
mP:=i maxért:=$\text{gyök}(p[i].x^2+p[i].y^2)^2$		—

A maxért jelentését újrafogalmaztuk: legyen a távolság-négyzetek maximuma! Így kap értéket a 2 értékadásban.

Ez legális mivel, $0 \le a \le b \rightarrow a^2 \le b^2$, ezért a feltételben szereplő reláció szemantikusan változatlan marad.

A maxért nem kimeneti adat, így a specifikációt sem sérti.

Bevezető példa

Változó
i:**Egész**
maxért:**Valós**

$mP:=1;\ maxért:=gyök(p[1].x^2+p[1].y^2)^2$

$i=2..n$

I	$gyök(p[i].x^2+p[i].y^2)^2>maxért$	N
$mP:=i$		—
$maxért:=gyök(p[i].x^2+p[i].y^2)^2$		

A négyzetre emelés és gyökvonás egymás inverzei, tehát kiejtik egymást, és gyök(A)<gyök(B) → A<B, miatt a feltétel szemantikusan változatlan.

Bevezető példa

Változó
i:**Egész**
maxért:**Valós**

mP:=1; maxért:=$p[1].x^2+p[1].y^2$

i=2..n

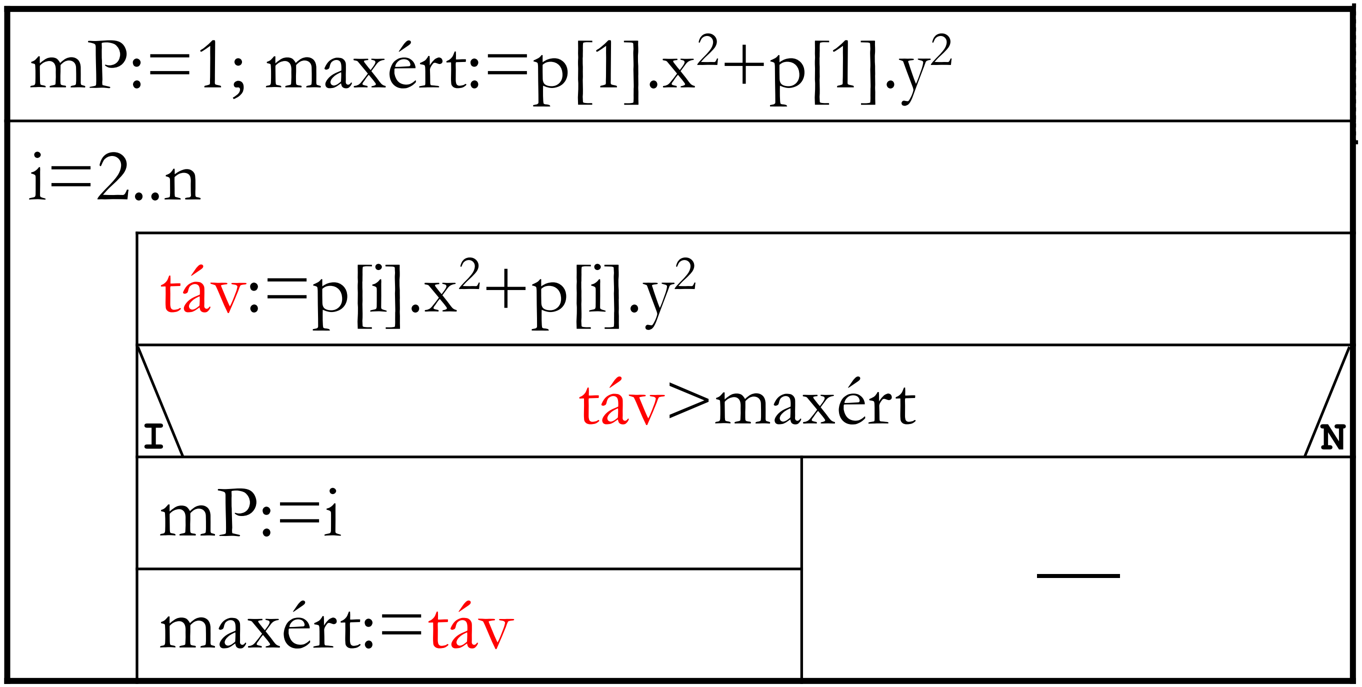
I — $p[i].x^2+p[i].y^2$>maxért — N

I	N
mP:=i	—
maxért:=$p[i].x^2+p[i].y^2$	

Itt még ugyanazt a képletet többször számítjuk ki (a ciklusban).
Használjunk egy segéd változót!

Bevezető példa

Változó
i:**Egész**
maxÉrt,
táv:**Valós**



Nevezetes programtranszformációk

Párhuzamos értékadás kifejtése:

$$a,b,c:=f(x),g(x),h(x)$$

Egymás utáni kiszámításra bontható, ha az összefüggés körmentes:

$$a:=f(x);\ b:=g(x);\ c:=h(x)$$

Nevezetes programtranszformációk

Párhuzamos értékadás kifejtése (ellenpélda):

```
a,b,c:=b,c,a
```

A szabály szerinti „szekvenciális párja”:

```
a:=b; b:=c; c:=a
```

Baj van: a kör-körös hivatkozás miatt a szekvenciális végrehajtás során megváltozott érték kerül a később értéket kapó változóba.

Nevezetes programtranszformációk

Párhuzamos értékadás kifejtése$_2$:

a,b,c:=b,c,a

segédváltozóval egymás utáni kiszámításra bontható, ha az összefüggés kört tartalmaz:

segéd:=a; a:=b; b:=c; c:=segéd

Változó
segéd:TH

Nevezetes programtranszformációk

Párhuzamos értékadás kifejtése$_{\text{általános}}$**:**

$\boxed{a,b,c:=f(x,a,b,c),g(x,a,b,c),h(x,a,b,c)}$

segédváltozókkal egymás utáni kiszámításra bontható, ha az összefüggés kört tartalmaz:

Változó

sa,sb,sc:TH

sa:=a; sb:=b; sc:=c

a:=f(x,sa,sb,sc); b:=g(x,sa,sb,sc); c:=h(x,sa,sb,sc)

Nevezetes programtranszformációk

Függvénykompozíció:

$$\boxed{B:=g(A);\ C:=f(B)}$$

Ha

, $k \in N$

1. az *f* függvény nem változtatja meg a paraméterét (B-t), és
2. a B értékére nincs később szükség,

akkor

$$\boxed{C:=f(g(A))}$$

Nevezetes programtranszformációk

Utasítások cseréje:

$$y{:=}f(x);\;\; z{:=}g(x)$$

Ha az *f* és *g* függvények nem változtatják meg a bemenő x paramétert, akkor

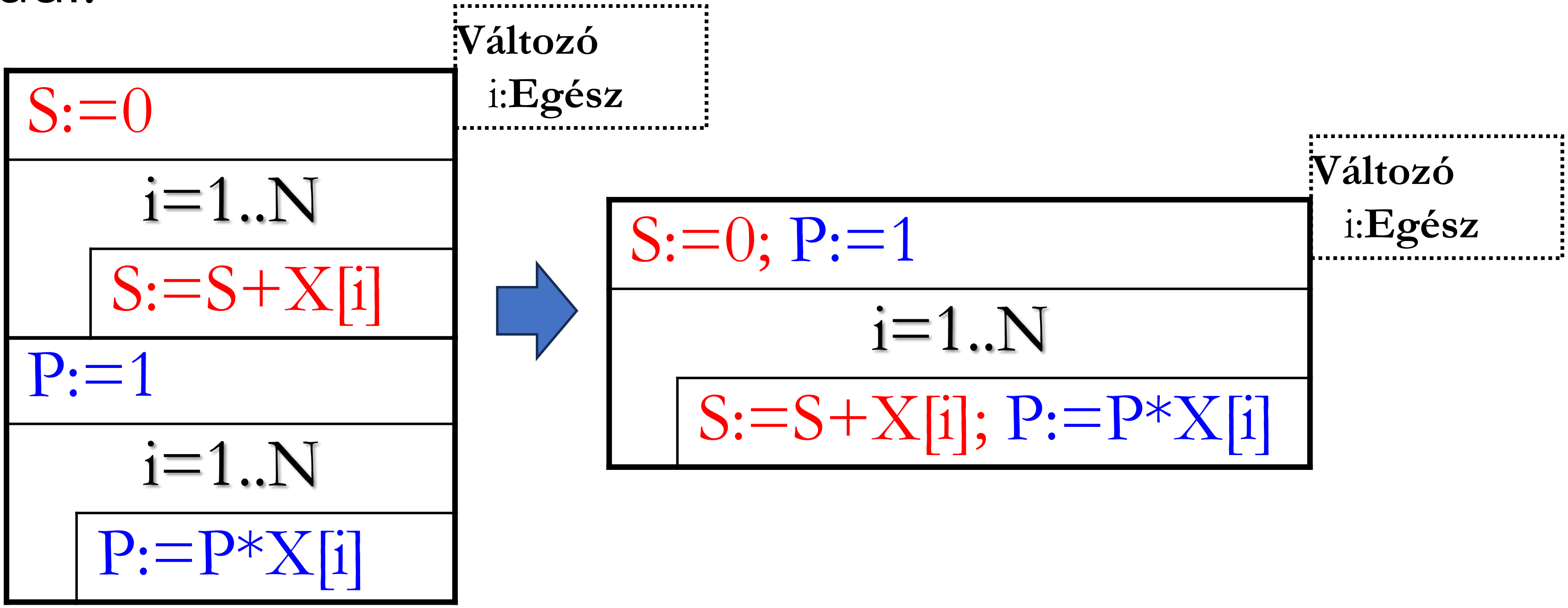
$$z{:=}g(x);\; y{:=}f(x)$$

Nevezetes programtranszformációk

Ciklusok összevonása:

Azonos lépésszámú ciklusok összevonhatóak, ha függetlenek egymástól.

Például:

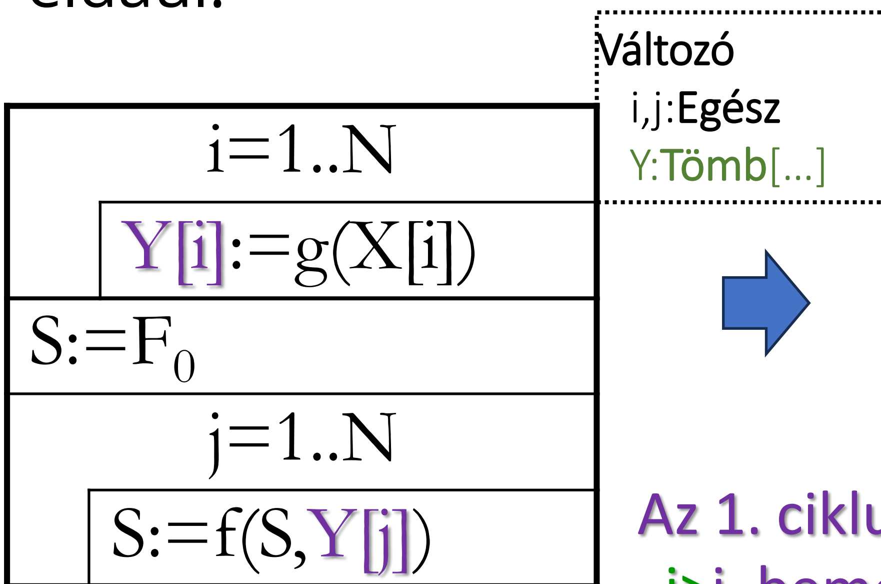


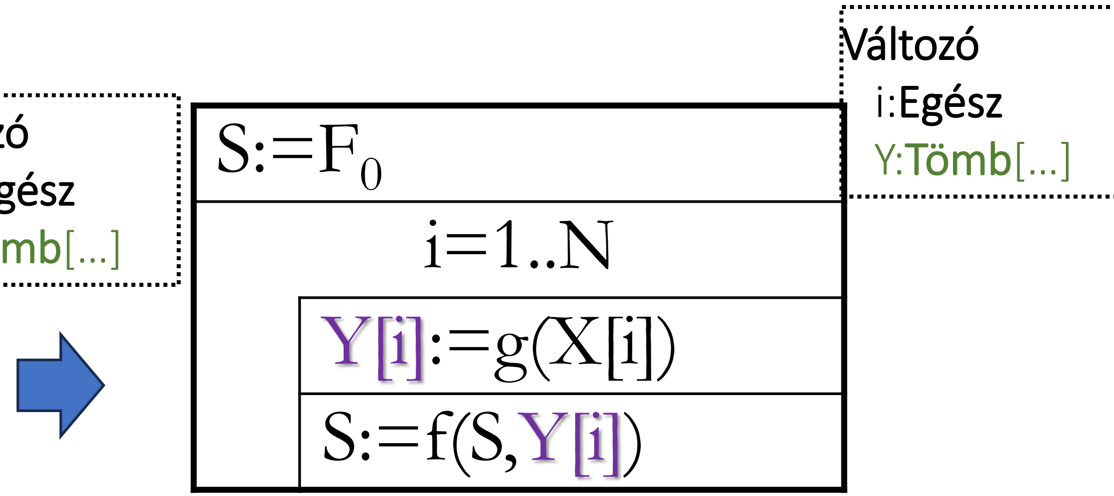
Nevezetes programtranszformációk

Ciklusok összevonása (gyenge függés):

„Gyenge” függés megengedhető.

Például:





Az 1. ciklusmag „i. kimenete” lehet a 2. ciklus „j>i. bemenete”, de a 2. ciklus „i. kimenete” nem lehet az 1. ciklus „j>i bemenete”.

Nevezetes programtranszformációk

Ciklusok összevonása (ellenpélda):

A várhatóérték (M) és szórás kiszámolása (S):

Változó
  i:**Egész**

M:=0
i=1..N
  M:=M+X[i]
M:=M/N
S:=0
i=1..N
  S:=S+(M–X[i])^2
S:=Gyök(S/N)

Változó
  i:**Egész**

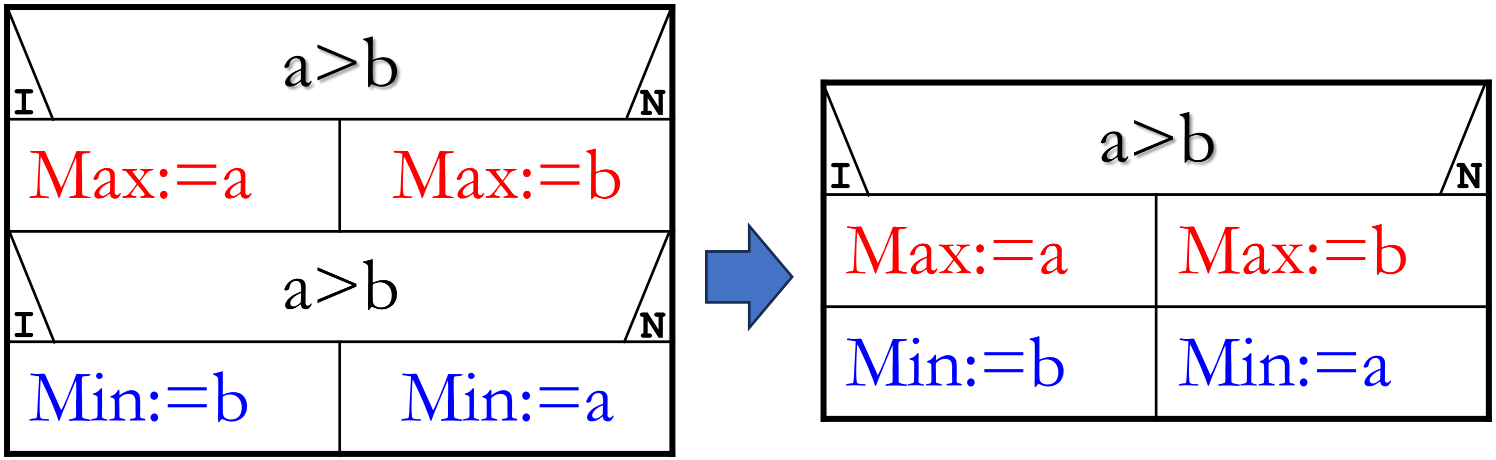
M:=0; S:=0
i=1..N
  M:=M+X[i]
  S:=S+(M–X[i])^2
M:=M/N; S:=Gyök(S/N)

Baj van: M még nem „kész”, amikor felhasználásra kerül.

Nevezetes programtranszformációk

Elágazások összevonása:

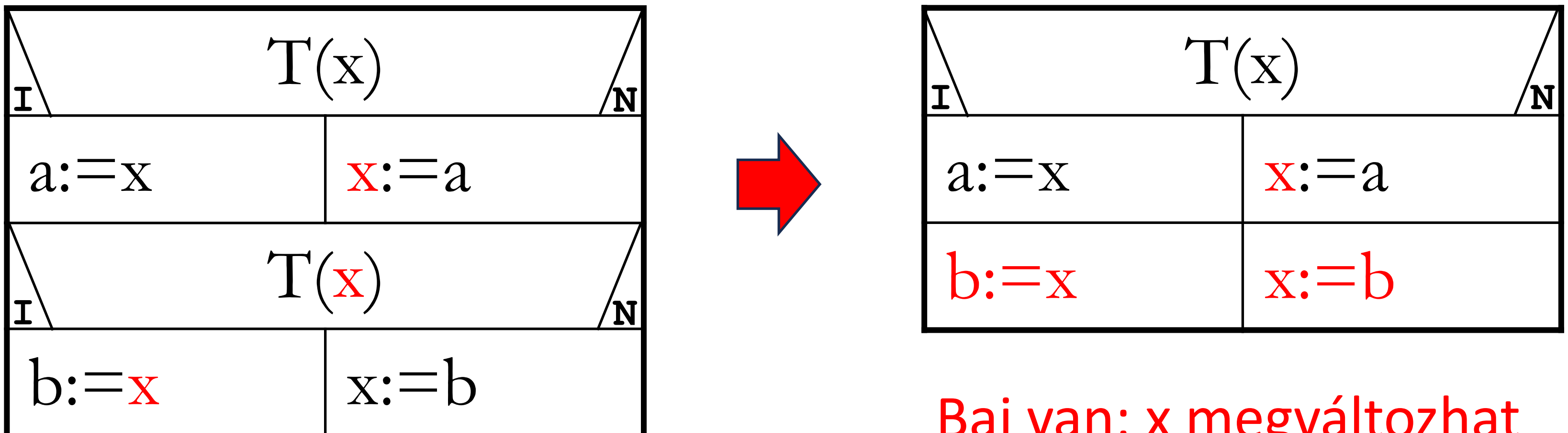
Azonos feltételű elágazások összevonhatóak, ha függetlenek egymástól.



Függetlenek, ha az 1. feltétel egyik ágán sem változik meg sem az ‚a' , sem a ‚b' változó (kifejezés). Gondolja meg: mikor nem független a két elágazás, ha ‚feltétel(a,b)' függvény a közös feltétel?

Nevezetes programtranszformációk

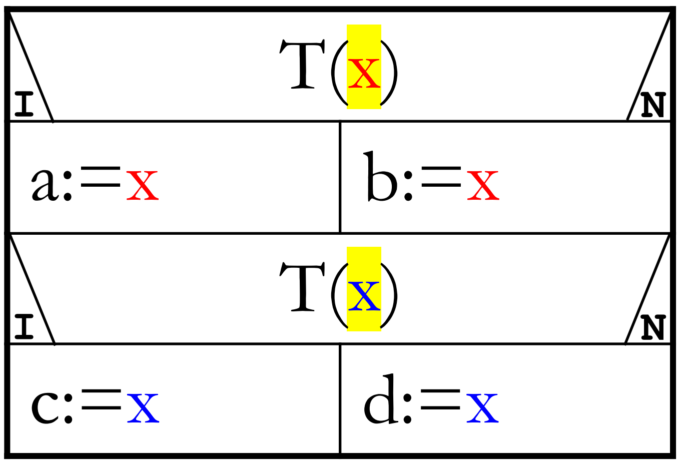
Elágazások összevonása (ellenpélda):

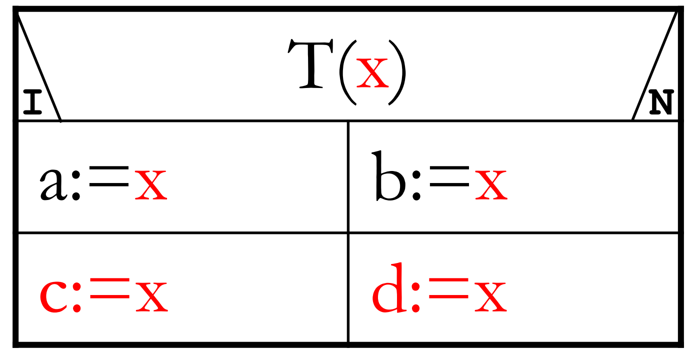


Baj van: x megváltozhat
a második elágazásig.

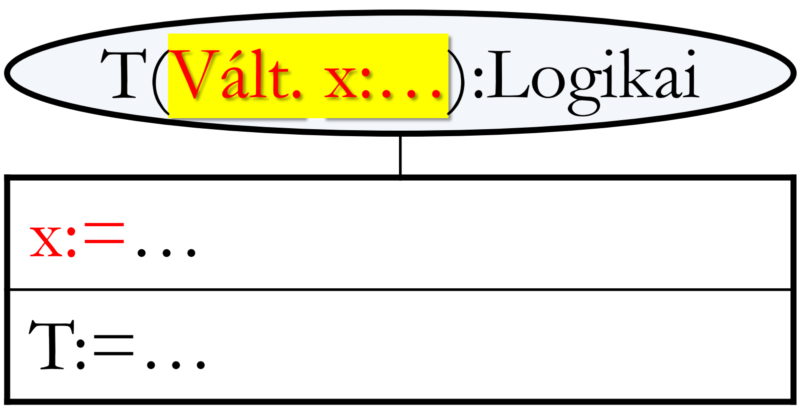
Nevezetes programtranszformációk

Elágazások összevonása (ellenpélda):





Baj van: x megváltozhat az első feltételben.



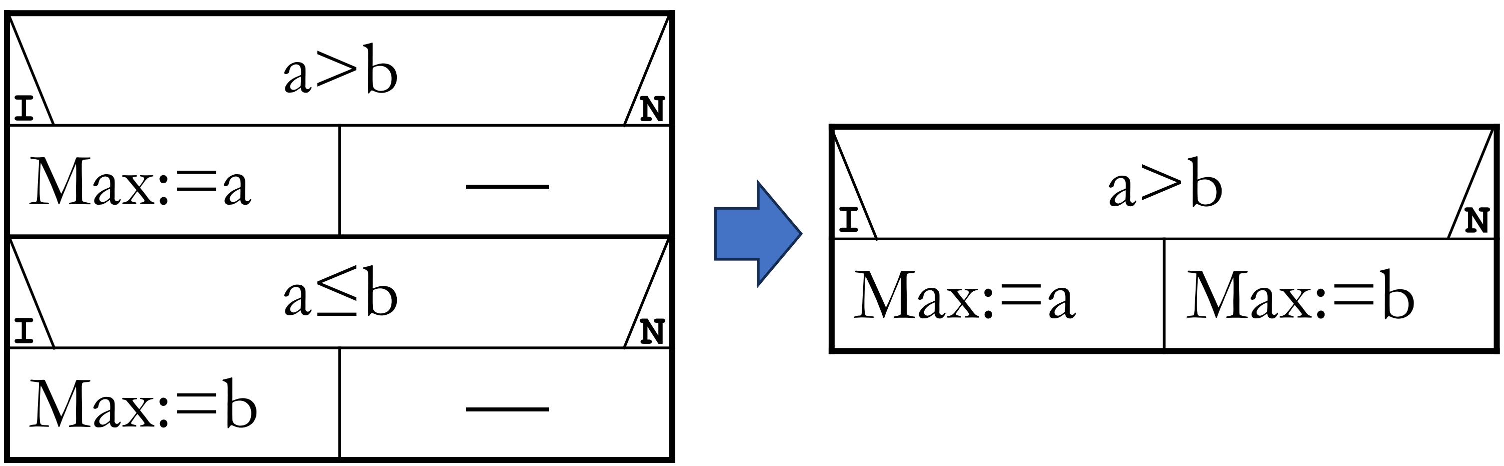
Nevezetes programtranszformációk

a>b és a≤b ≡ Hamis

a>b vagy a≤b ≡ Igaz

Elágazások összevonása:

Kizáró feltételű, teljes (egyágú) elágazások is összevonhatók, ha függetlenek egymástól.

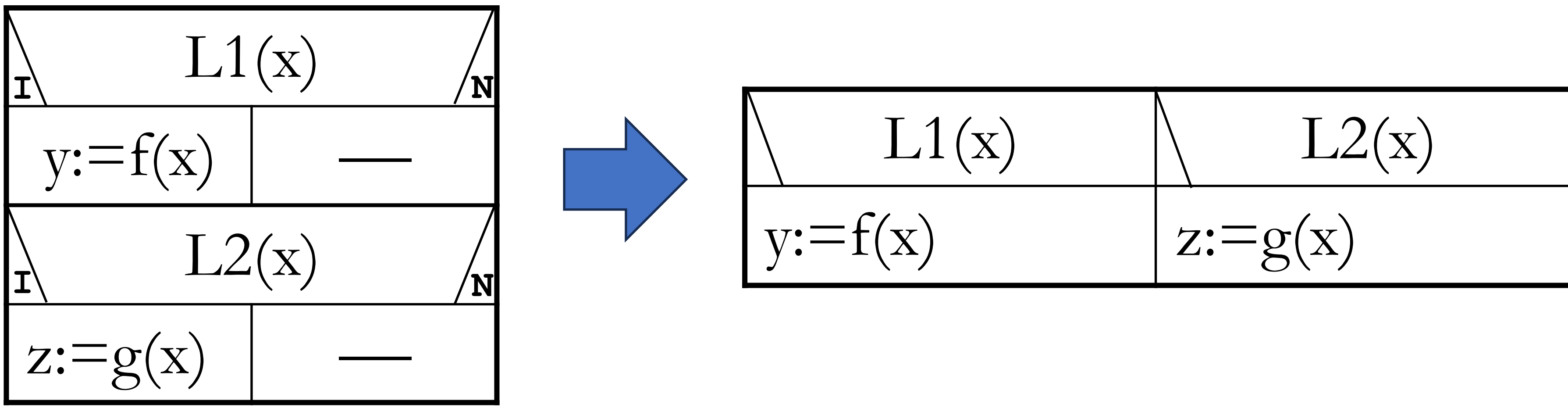


Nevezetes programtranszformációk

L1(x) és L2(x) ≡ Hamis

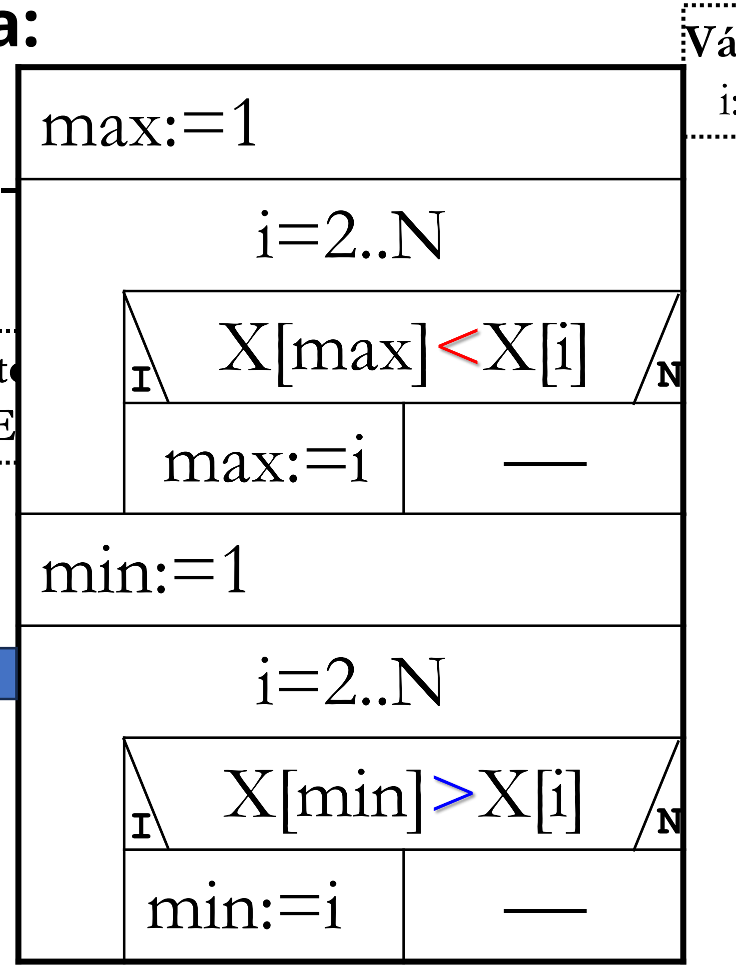
Elágazások összevonása:

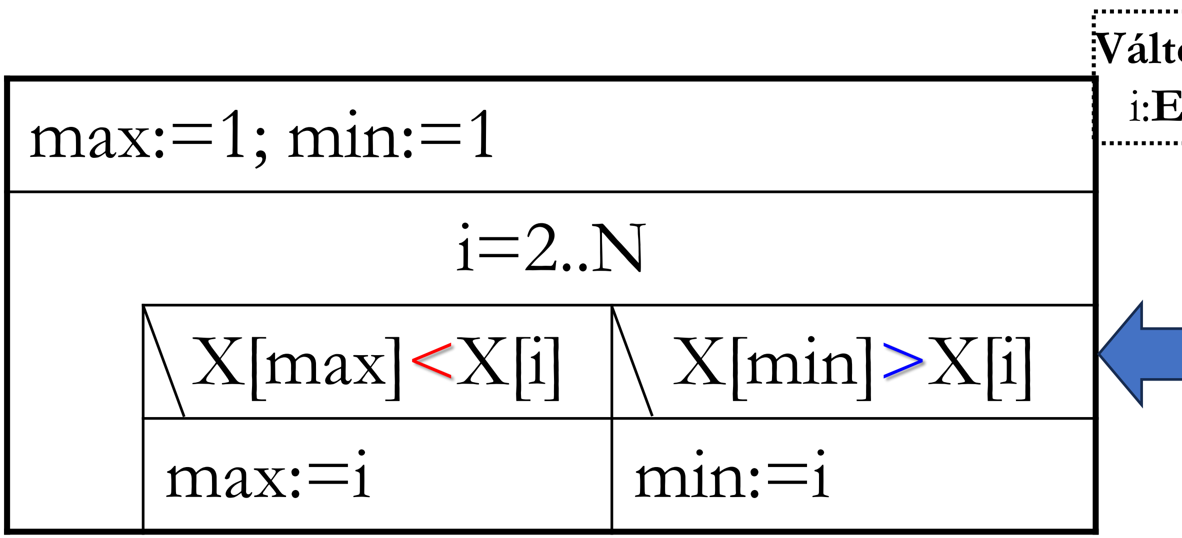
Kizáró feltételű (egyágú) elágazásokkal is összevonhatók, ha függetlenek egymástól.



Nevezetes programtranszformációk

Ciklusok és elágazások összevonása:
Azonos lépésszámú ciklusok, bennük kizáró feltételű elágazásokkal is összevonhatók, ha függetlenek egymástól.

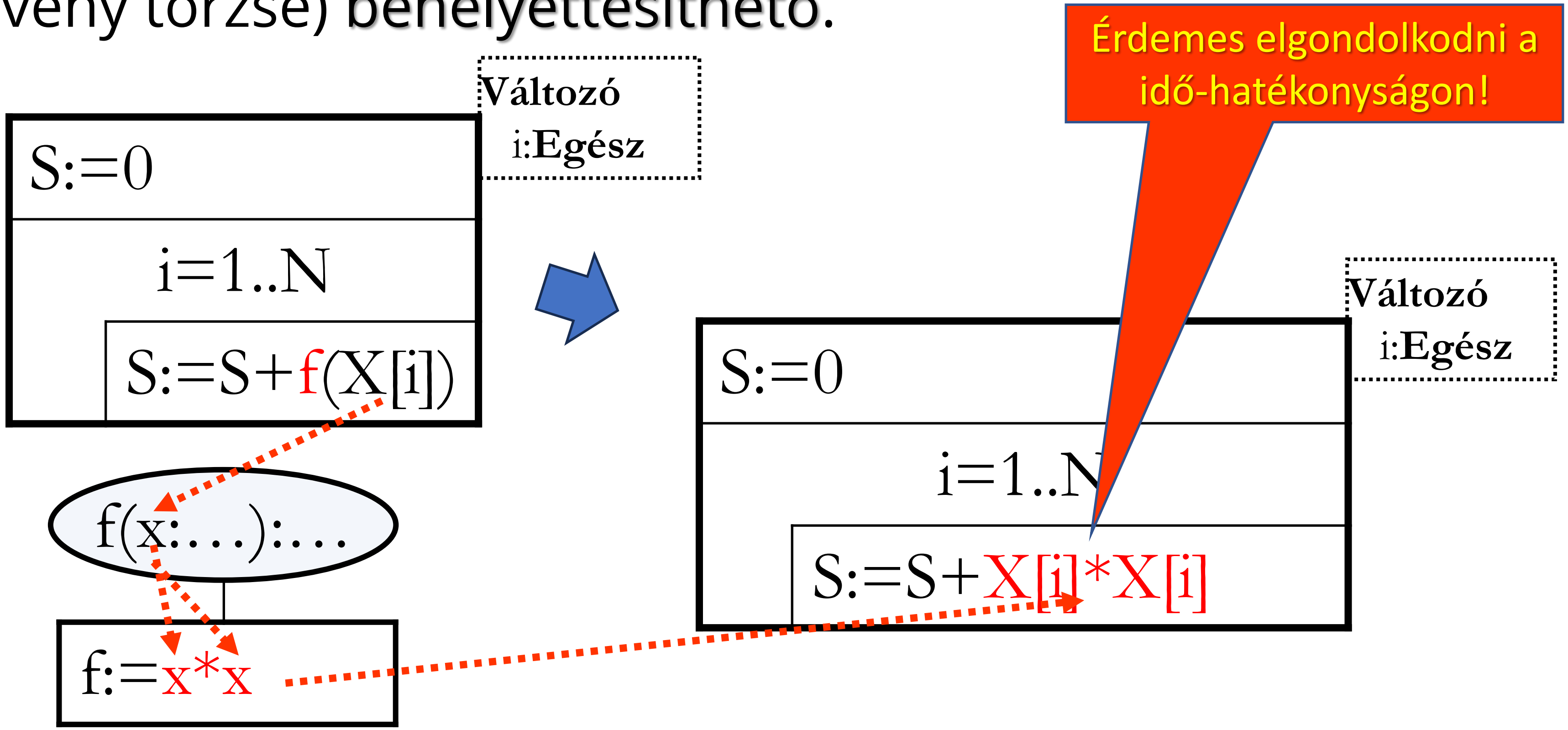




Nevezetes programtranszformációk

Függvény behelyettesítése:

Függvényhívás helyére egy (egyszerű) függvény képlete (a függvény törzse) behelyettesíthető.



Nevezetes programtranszformációk

Utasítás kiemelése ciklusból:

A ciklus magjából a ciklustól független utasítások kiemelhetők.

(A fordítók ilyen optimalizálást többnyire el tudnak végezni.)

Változó
i:**Egész**

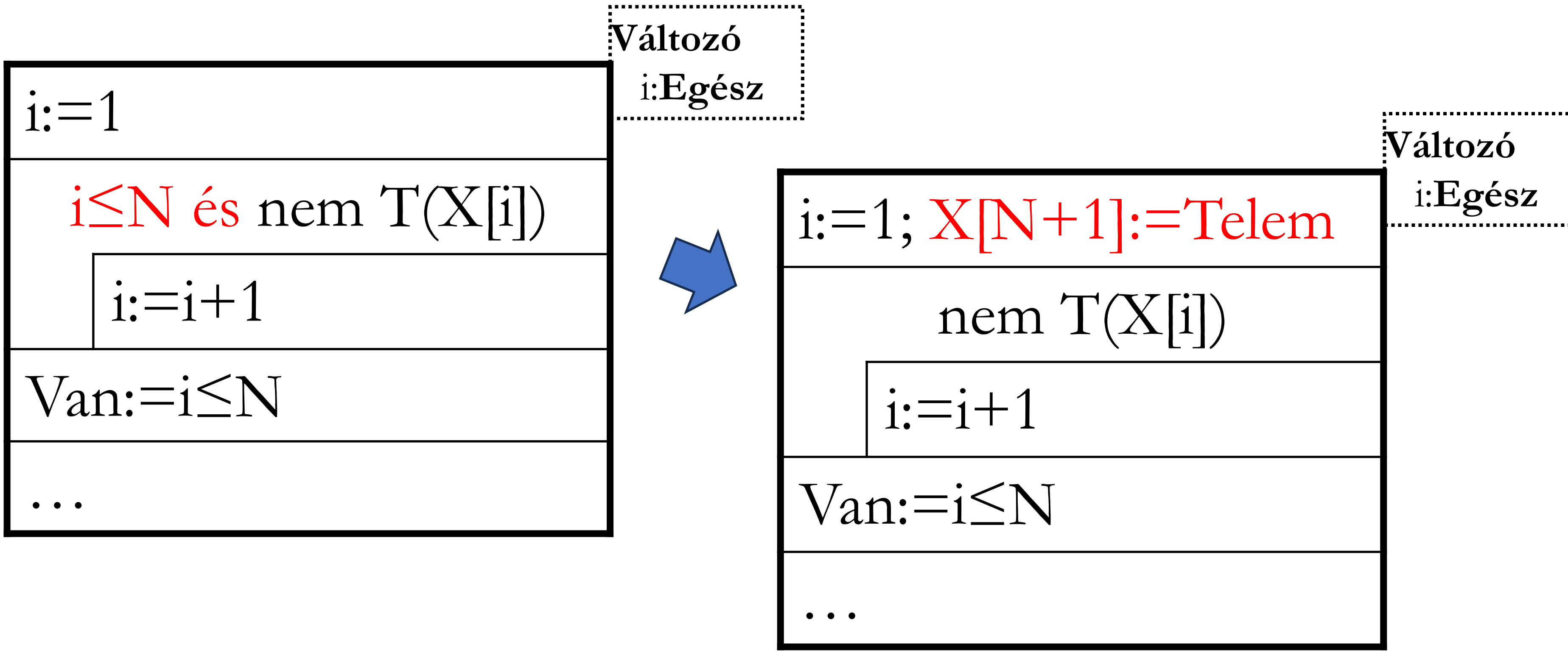
S:=0
i=1..N
S:=S+h[i]*r^2*pi

Változó
i:**Egész**
rpi:**Valós**

S:=0; rpi:=r^2*pi
i=1..N
S:=S+h[i]*rpi

Nevezetes programtranszformációk

„Keresés, eldöntés → kiválasztás” transzformáció:
A vizsgálandó sorozat végére helyezzünk egy T tulajdonságú elemet (=Telem) → biztosan találunk ilyet!



Programtranszformációk alkalmazása

Tételek összeépítése

Tételek összeépítése elé…

- Tétel-kombinálás „módszertana”:
    - mechanikusan vagy
    - „okosan”

- Az elkövetkezőkben sorozatokon értelmezzük a programozási tételeket

Tételek összeépítése elé…

Milyen tételek lehetnek „főszereplők” (T_0) a kombináláskor?

Tekintsük a tételeket mint függvényeket (függvény-szignatúra):

tétel: bemenet (értelmezési tartomány) $\rightarrow$ *kimenet (értékkészlet)*

A bemenet minden tétel esetében legalább 1 sorozat.

Tételek összeépítése elé…

Milyen tételek lehetnek „főszereplők” (T_0) a kombináláskor?

$T_0 \rightarrow T_i$

- T_0 kimenete (legalább 1) sorozat, akkor
  T_0 : másolás, kiválogatás (halmazos tételek, rendezés)
  T_i : bármely tétel

Tételek összeépítése elé…

Milyen tételek lehetnek „főszereplők” (T_0) a kombináláskor?

- T_0 kimenete logikai érték (is), akkor
  T_0 implementálhat egy tulajdonság-függvényt,
  pl. eldöntés (keresés)
  T_i : megszámolás, eldöntés, kiválasztás, keresés,
  kiválogatás, feltételes tételek

Másolással összeépítés

```
  Be: n∈N, x∈H[1..n]
  Ki: y∈H[1..n]
  Ef: -
  Uf: ∀i∈[1..n]:(y[i]=f(x[i]))
Rövidítve:
  Uf: y=MÁSOL(i=e..u, f(x[i]))
```

A **másolás** programozási tétellel összeépítés minden programozási tételre működik. (sorozat→sorozat)

Csupán annyi a teendő, hogy a bemenetben szereplő sorozat-értékek helyett az i-edik feldolgozandó elemként a másolásban szereplő f-transzformáltat kell írni. Például:

Összegzéssel összeépítés:

`SZUMMA(i=1..n,x[i]) → SZUMMA(i=1..n,f(x[i]))` vagy

Maximumkiválasztással összeépítés:

`MAX(i=1..n,x[i]) → MAX(i=1..n,f(x[i]))`

Itt tehát a „másik” tétel bemeneti sorozatára vonatkozik az f-transzformálás.

Másolással összeépítés

```
  Be: n∈N, x∈H[1..n]
  Ki: y∈H[1..n]
  Ef: -
  Uf: ∀i∈[1..n]:(y[i]=f(x[i]))
Rövidítve:
  Uf: y=MÁSOL(i=e..u, f(x[i]))
```

A **másolás** programozási tétellel összeépítés minden programozási tételre működik. (sorozat→sorozat)

Csupán annyi a teendő, hogy a kimenetben szereplő sorozat-értékek helyett az i-edik feldolgozandó elemként a másolásban szereplő f-transzformáltat kell írni. Például:

Kiválogatással összeépítés:

```
KIVÁLOGAT(i=1..n,T(x[i]),x[i]) → KIVÁLOGAT(i=1..n,T(x[i]),f(x[i]))
```

Itt tehát a „másik" tétel kimeneti sorozatára vonatkozik az f-transzformálás.

másolás+összegzés

Feladat: határozzuk meg az x sorozat elemei f transzformált-jainak az összegét !

Megoldás alapja az összegzés tétel.

Kérdés: Hogyan látható be a

```
SZUMMA(i=1..n,x[i]) → SZUMMA(i=1..n,f(x[i]))
```

formula-átalakítás helyessége?

másolás+összegzés

Feladat: határozzuk meg az x sorozat elemei f transzformáltjainak az összegét !

H1,H2 valamely számhalmaz

```
Be: n∈N, x∈H1[1..n]
Fv: f:H1->H2
Ki: s∈H2
Ef: -
Uf: s=SZUMMA(i=1..n,f(x[i]))
```

másolás+összegzés

Feladat: határozzuk meg az x sorozat elemei f transzformált-jainak az összegét !

Visszavezetjük a másolás és az összegzés tétel egymásutánjára:

```
Be: n∈N, x∈H1[1..n]
Sa: y∈H2[1..n]
Fv: f:H1->H2
Ki: s∈H2
Ef: -
Uf: y=MÁSOL(i=1..n,f(x[i])) és
    s=SZUMMA(i=1..n,y[i])
```

másolás+összegzés

Algoritmus:

```
Uf: y=MÁSOL(i=1..n,f(x[i])) és
    s=SZUMMA(i=1..n,y[i])
```

Változó
i:Egész
y:Tömb[...]

```
i=1..n
    y[i]:=f(x[i])
s:=0
i=1..n
    s:=s+y[i]
```

Észrevétel:
Az eredmény helyes, de bántóan nem hatékony.

másolás+összegzés

Az algoritmust programtranszformációkkal alakítsuk át!

Változó
i:**Egész**
y:**Tömb**[…]

i=1..n
y[i]:=f(x[i])
s:=0
i=1..n
s:=s+y[i]

Észrevételek:

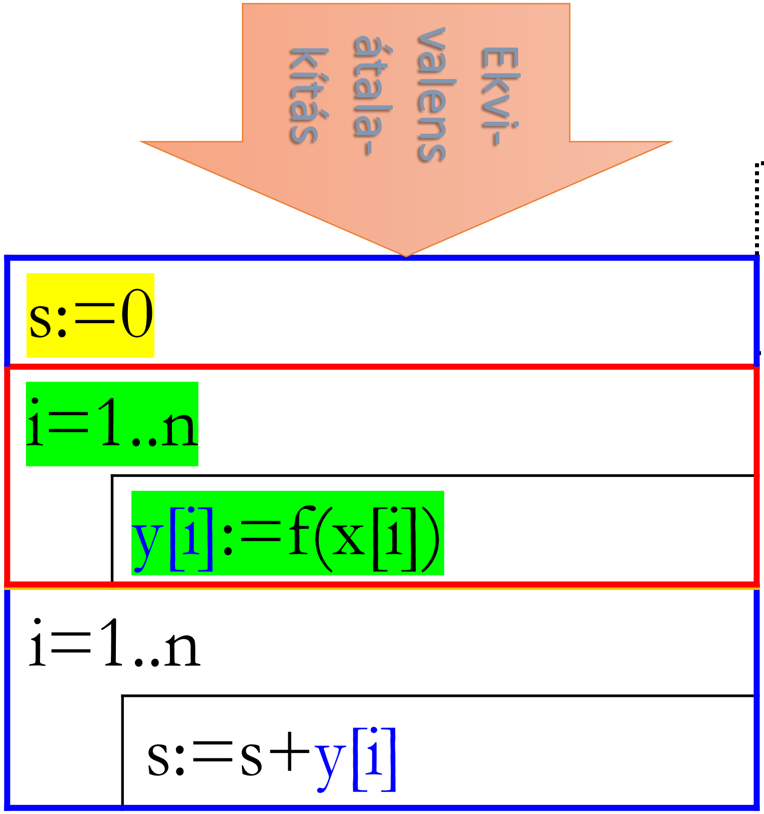
1. Azonos ciklus-szervezés.
2. Az s:=0 értékadás „útban van” a ciklusok esetleges összevonásának.
3. Az s:=0 meg kell előzze a 2. ciklust.

másolás+összegzés

1. programtranszformáció: utasítások cseréje

Észrevételek:

Az **utasítások cseréje** elvégezhető, hiszen az s:=0 nem változtatja meg az 1. ciklus bemenő paramétereit (az 1. ciklus: y[1..n]-re vonatkozó „érték-adás”, amely bemenete: n, x[1..n]).



Változó
i:Egész
y:**Tömb**[…]

Észrevételek:

1. Azonos ciklus-szervezés.
2. A 2. ciklus **i.** lépésben csak az y[i] kell.

másolás+összegzés

2. programtranszformáció: (gyenge függésű) ciklusok összevonása

Ekvivalens átalakítás

```
s:=0
i=1..n
    y[i]:=f(x[i])
    s:=s+y[i])
```

Változó
  i:**Egész**
  y:**Tömb**[…]

Észrevétel:

Az y[i] érték-„kapása” és felhasználása közvetlen egymás mellé került.

másolás+összegzés

3. programtranszformáció: függvénykompozíció

s:=0
i=1..n
y[i]:=f(x[i])
s:=s+y[i]

Észrevétel:

B:=*g*(A); C:=*f*(B)

↔

C:=*f*(*g*(A))

Ekvivalens átalakítás

Változó
i:Egész

s:=0
i=1..n
s:=s+f(x[i])

Végülis beláttuk: `SZUMMA(i=1..n,f(x[i]))`

kiválogatás+összegzés

Feladat: adott tulajdonságúak összege (feltételes összegzés).

```
Be: n∈N, x∈H[1..n]
Ki: s∈H
Ef: -
Uf: s=SZUMMA(i=1..n,x[i],T(x[i]))
```

H valamely számhalmaz

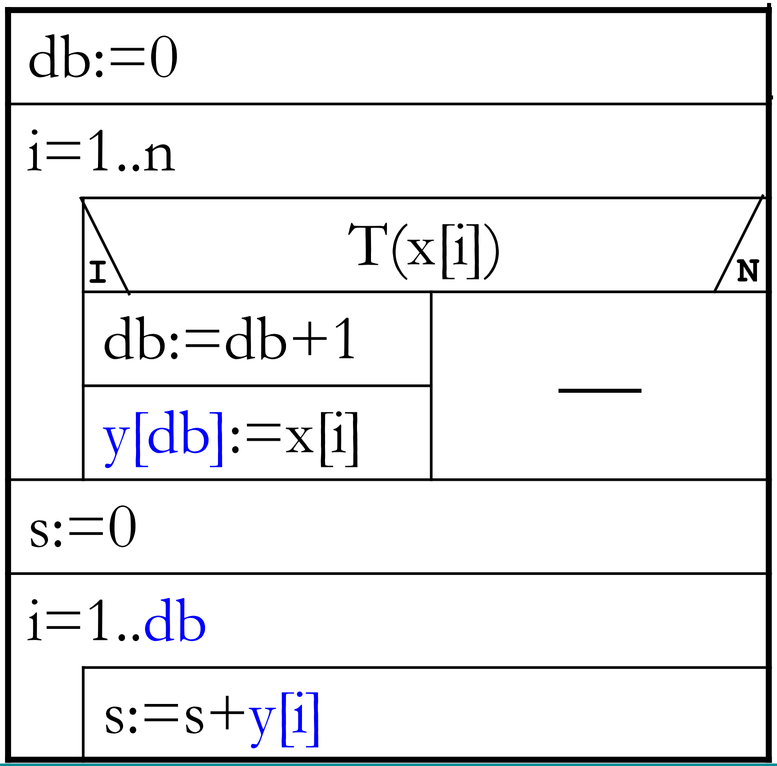
Visszavezetjük a kiválogatás és az összegzés tétel egymásutánjára:

```
Be: n∈N, x∈H[1..n]
Sa: db∈N, y∈H[1..db]
Ki: s∈H2
Ef: -
Uf: (db,y)=KIVÁLOGAT(i=1..n,T(x[i]),x[i]) és
    s=SZUMMA(i=1..db,y[i])
```

kiválogatás+összegzés

Algoritmus:

```
Uf: (db,y)=KIVÁLOGAT(i=1..n,T(x[i]),x[i]) és
    s=SZUMMA(i=1..db,y[i])
```



Változó
i,db:**Egész**
y:Tömb[…]

Észrevétel:
Az eredmény helyes, de bántóan nem hatékony.

kiválogatás+összegzés

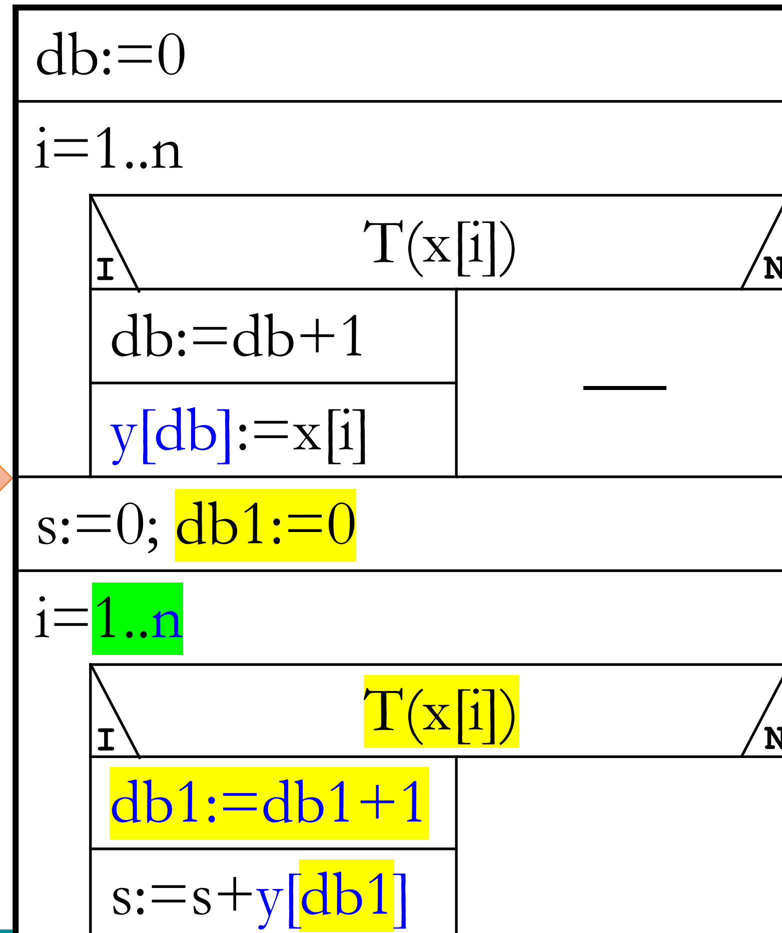
Az algoritmust programtranszformációkkal alakítsuk át!

Változó
i,db,
db1:**Egész**
y:Tömb[…]

Észrevétel:

A két különböző szervezésű ciklus hasonlóvá tétele, a második ciklus szemantikus ekvivalenciájának biztosítása mellett.

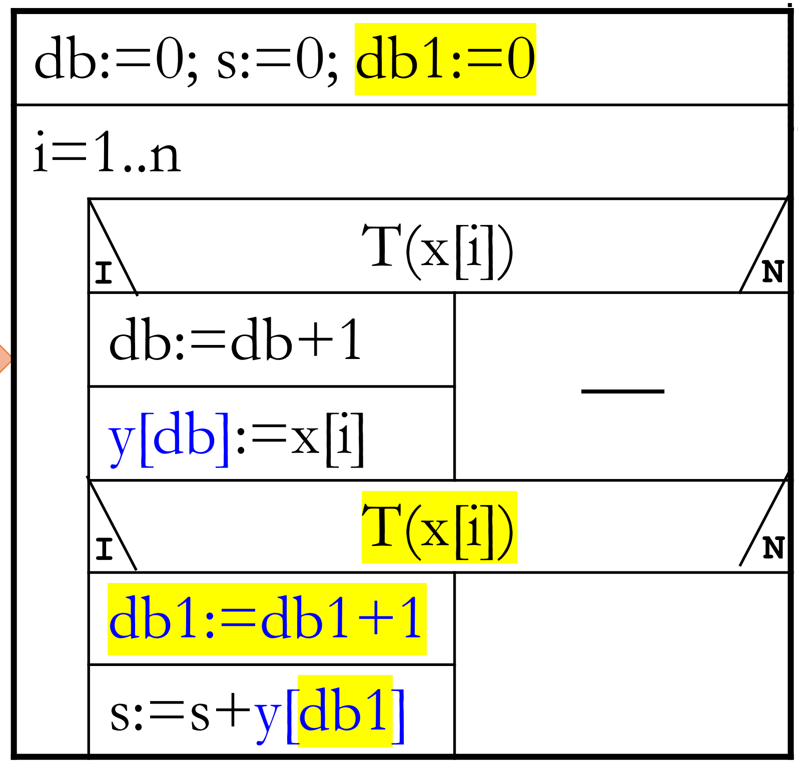
Ekvivalens átalakítás



kiválogatás+összegzés

Programtranszformáció: (gyengén függő) ciklusok összevonása

Ekvivalens átalakítás



Változó
i,db,
db1:**Egész**
y:Tömb[…]

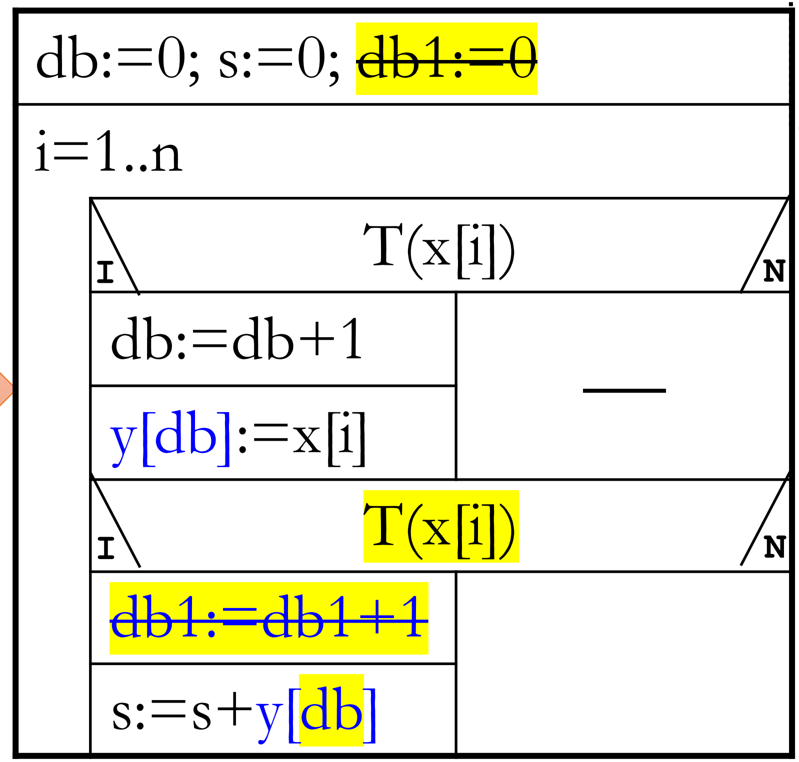
kiválogatás+összegzés

Változó
i,db,
~~db1~~:Egész
y:Tömb[…]

Észrevétel:

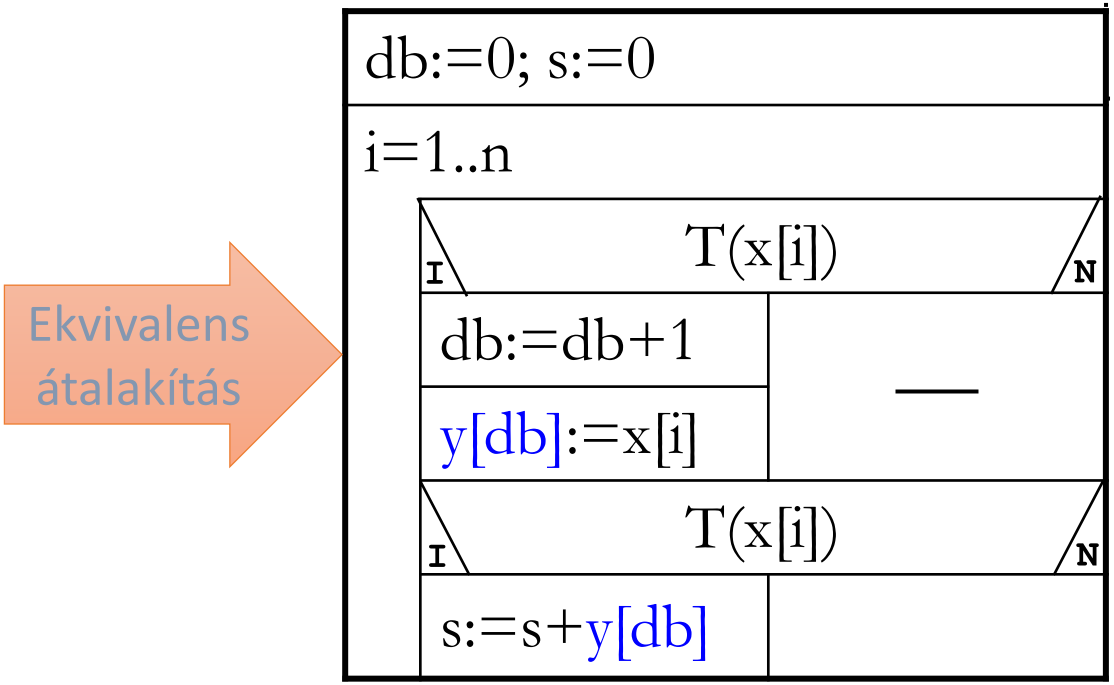
A db és db1 változók szinkronban változnak, minden cikluslépésben azonos értékűek. Így a db1 változó elhagyható, a rá vonatkozó értékadások elhagyhatók, az y[db1] helyett y[db] írandó.

Ekvivalens átalakítás



kiválogatás+összegzés

Változó
i,db:**Egész**
y:Tömb[…]



Észrevétel:
Két azonos feltételű elágazás egymás-után.

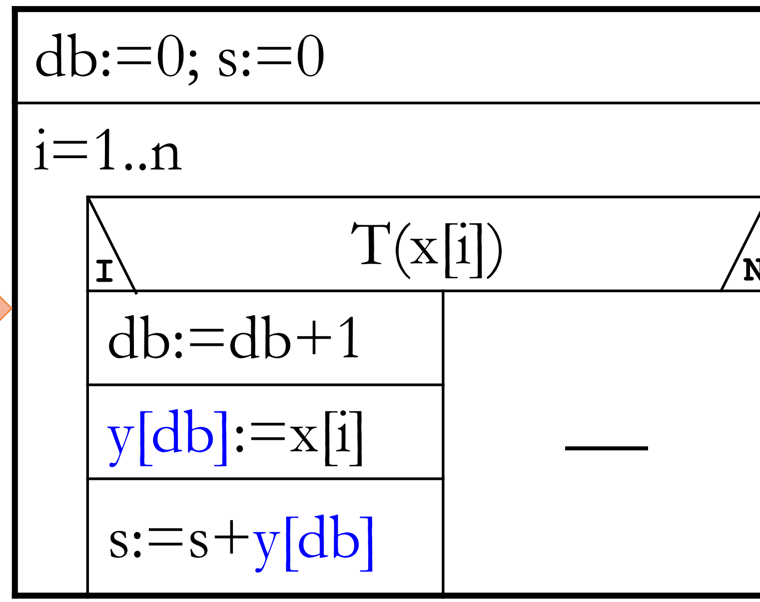
kiválogatás+összegzés

Programtranszformáció: elágazások összevonása

Észrevétel:

2, azonos feltételű elágazás összevonható (mivel a feltétel paramétere az elágazásban nem változik meg)

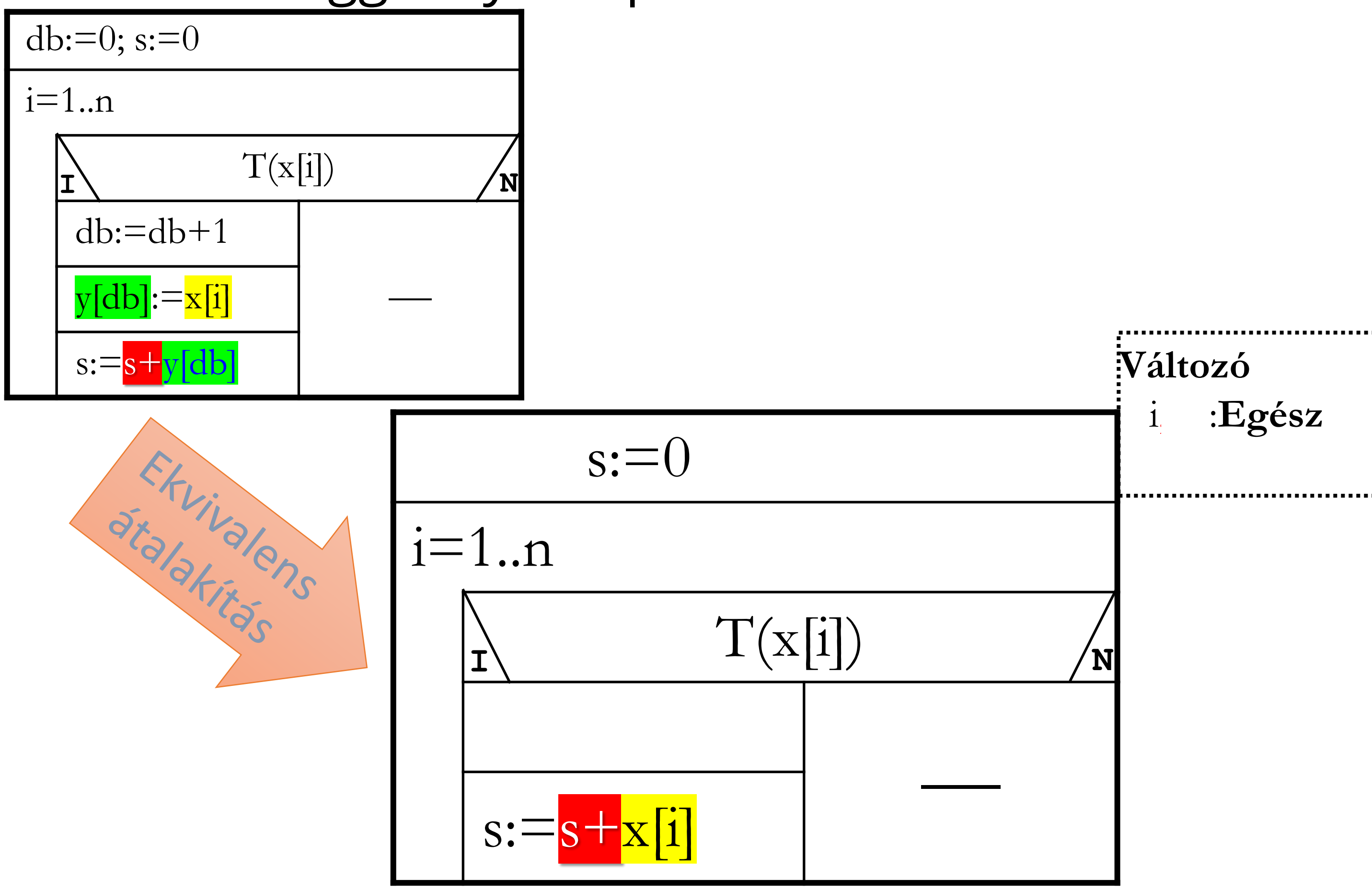
Ekvivalens átalakítás



Változó
i,db:**Egész**
y:Tömb[…]

kiválogatás+összegzés

Programtranszformáció: függvénykompozíció



Észrevétel:

B:=*g*(A); C:=*f*(B)

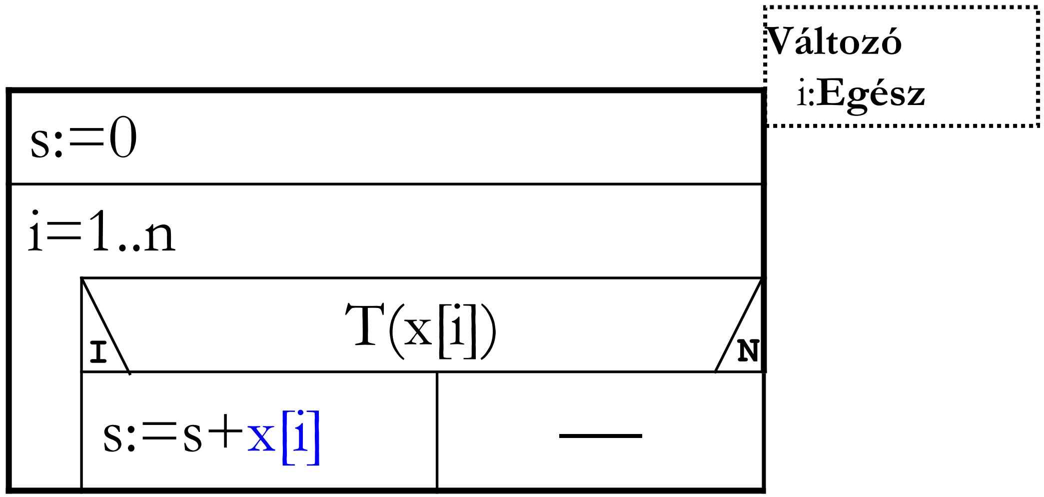
↔

C:=*f*(*g*(A))

→

y és db elhagyható

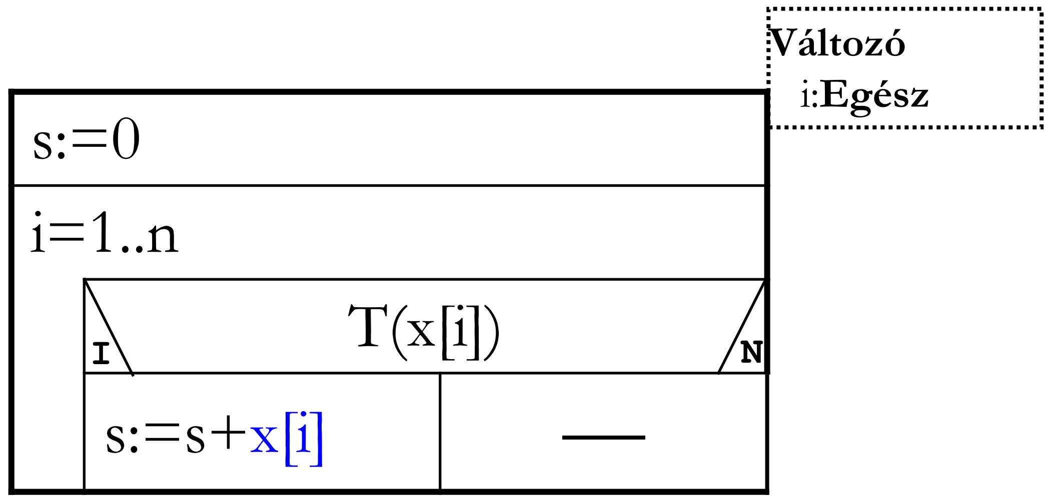
kiválogatás+összegzés



Végülis beláttuk: `s=SZUMMA(i=1..n,x[i],T(x[i]))`

kiválogatás+összegzés helyességbizonyítással

Algoritmikus gondolkodással: kiválogatás nélkül azonnal adjuk össze a megfelelő elemeket!



Ez esetben bizonyítanunk kell a helyességet!

Bepillantunk a programozási tételek bizonyításának módszertanába is.

kiválogatás+összegzés helyességbizonyítással

Helyességbizonyítás: az algoritmus kielégíti-e a specifikációt?

Uf: `s=SZUMMA(i=1..n,x[i],T(x[i]))`

Ciklusinvariáns (`C(i)`) állítás, a ciklus-magba belépéskor kiértékelendő:

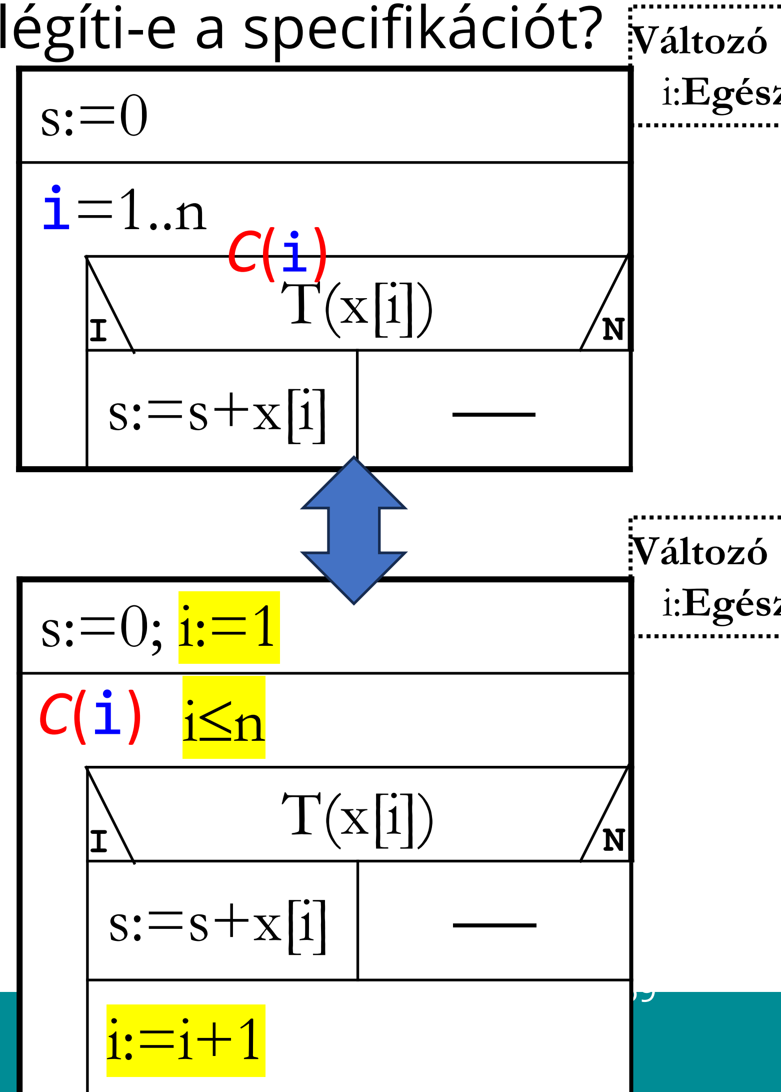
`s=SZUMMA(j=1..i-1,x[j],T(x[j]))`

Indukciós bizonyítás lépései:

Ciklusba belépéskor teljesül a `C(1)`.

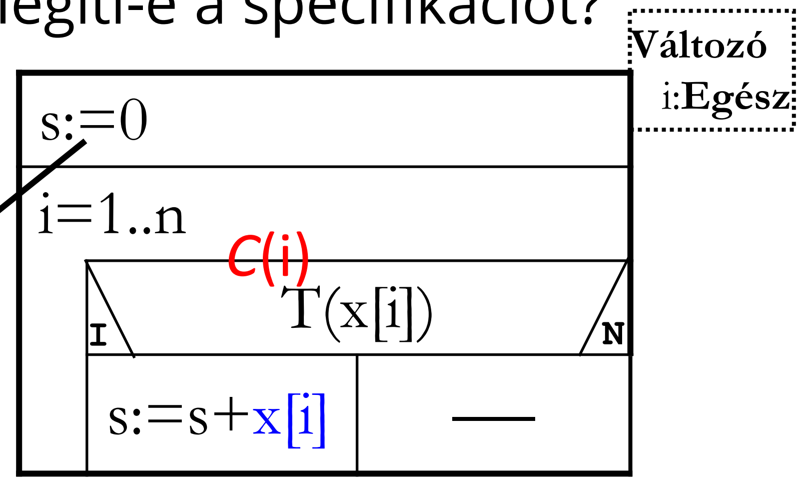
Ciklus egyszeri végrehajtás után a igaz marad: `C(i)`→`C(i+1)`.

Ciklusból kilépéskor `C(n+1)`→Uf.



kiválogatás+összegzés helyességbizonyítással

Helyességbizonyítás: az algoritmus kielégíti-e a specifikációt?



Uf: `s=SZUMMA(i=1..n,x[i],T(x[i]))`

Ciklusinvariáns (`C(i)`) állítás, a ciklus-magba belépéskor kiértékelendő:

`s=SZUMMA(j=1..i-1,x[j],T(x[j]))`

Indukciós bizonyítás:

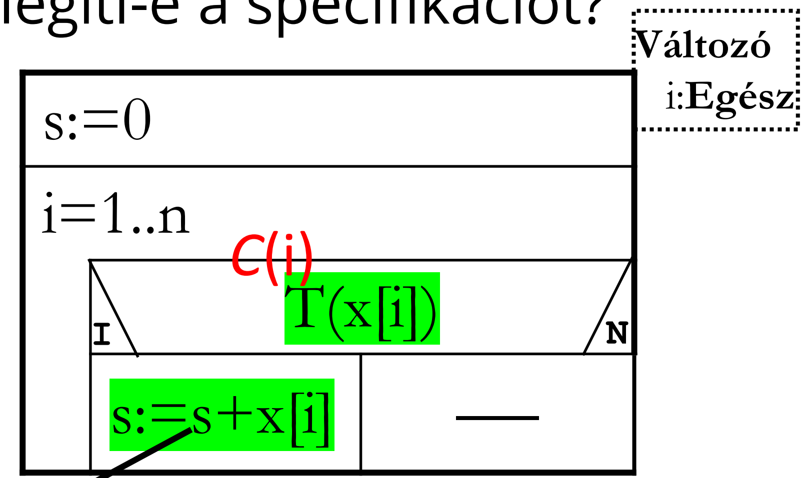
Ciklusba belépéskor (`i`=1):

```
C(1)= s=SZUMMA(j=1..i-1,x[j],T(x[j]))
        SZUMMA(j=1..1-1,x[j],T(x[j]))=0
      s=0
      0=0
```

✓

kiválogatás+összegzés helyességbizonyítással

Helyességbizonyítás: az algoritmus kielégíti-e a specifikációt?



Uf: `s=SZUMMA(i=1..n,x[i],T(x[i]))`

Ciklusinvariáns (`C(i)`) állítás, a ciklusmagba belépéskor kiértékelendő:

`s=SZUMMA(j=1..i-1,x[j],T(x[j]))`

Indukciós lépés:

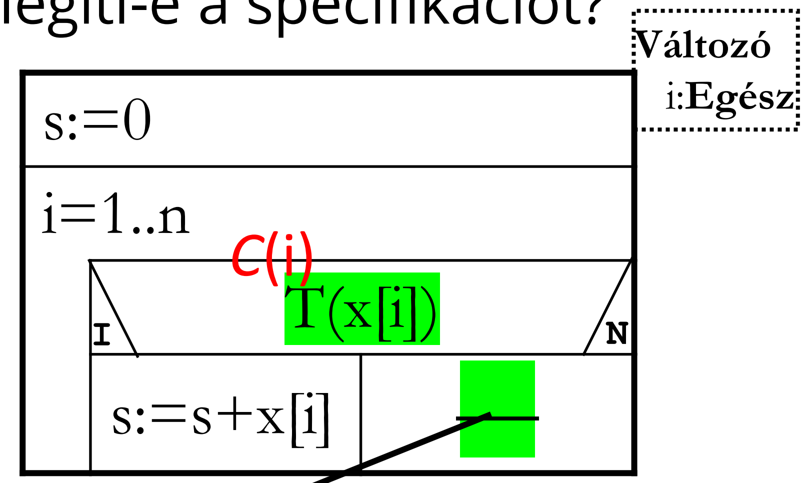
Ciklusmag egyszeri végrehajtása után (i→i+1):

```
C(i) és T(x[i]) → s:=s+x[i] →
    s=SZUMMA(j=1..i-1,x[j],T(x[j]))+x[i]=
      SZUMMA(j=1..i  ,x[j],T(x[j]))=
      C(i+1)
```

✓

kiválogatás+összegzés helyességbizonyítással

Helyességbizonyítás: az algoritmus kielégíti-e a specifikációt?



Uf: `s=SZUMMA(i=1..n,x[i],T(x[i]))`

Ciklusinvariáns (`C(i)`) állítás, a ciklusmagba belépéskor kiértékelendő:

`s=SZUMMA(j=1..i-1,x[j],T(x[j]))`

Indukciós lépés:

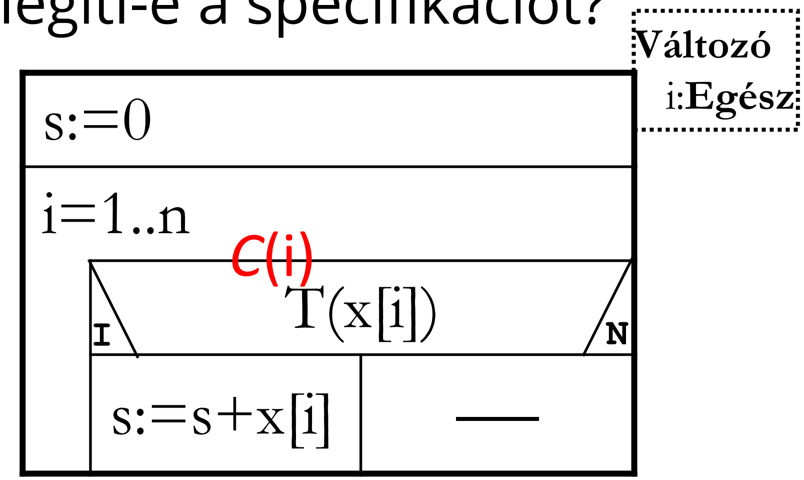
Ciklusmag egyszeri végrehajtása után (i→i+1):

`C(i) és nem T(x[i])` → `s:=s+0` →
`s=SZUMMA(j=1..i-1,x[j],T(x[j]))+0=`
`SZUMMA(j=1..i ,x[j],T(x[j]))=`
`C(i+1)`

✓

kiválogatás+összegzés helyességbizonyítással

Helyességbizonyítás: az algoritmus kielégíti-e a specifikációt?



Uf: `s=SZUMMA(i=1..n,x[i],T(x[i]))`

Ciklusinvariáns (`C(i)`) állítás, a ciklusmagba belépéskor kiértékelendő:

`s=SZUMMA(j=1..i-1,x[j],T(x[j]))`

Ciklusból kilépéskor($i \rightarrow n+1$):

`C(n+1)=`

`s=SZUMMA(j=1..i-1,x[j],T(x[j]))+0=`

`SZUMMA(j=1..n ,x[j],T(x[j]))=`

`Uf`

✓

Maximumkiválasztás+kiválogatás

Feladat: összes maximális elem kiválogatása.

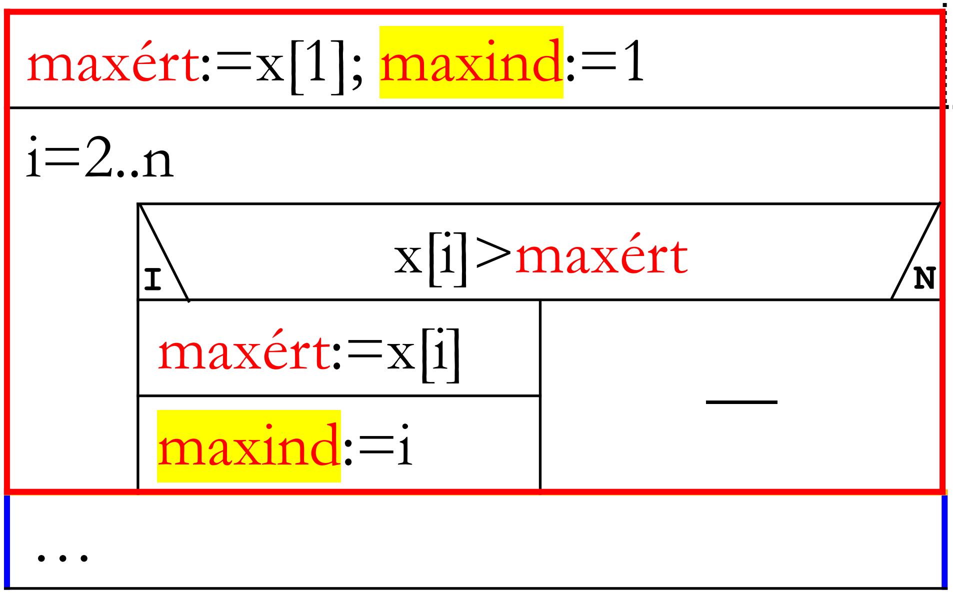
```
Be: n∈N, x∈H[1..n]
Ki: db∈N, maxI∈N[1..db]
Sa: maxért∈H
Fv: legnagyobb:H->L, legnagyobb(h)=h=maxért
Ef: n>0
Uf: (,maxért)=MAX(i=1..n,x[i]) és
    (db,maxI)=KIVÁLOGAT(i=1..n,legnagyobb(x[i]),i)
```

Maximumkiválasztás+kiválogatás

Algoritmus:

```
Uf: (,maxért)=MAX(i=1..n,x[i]) és
    (db,maxI)=KIVÁLOGAT(i=1..n,legnagyobb(x[i]),i)
```

Változó
  i,maxind:**Egész**
  maxért:**TH**

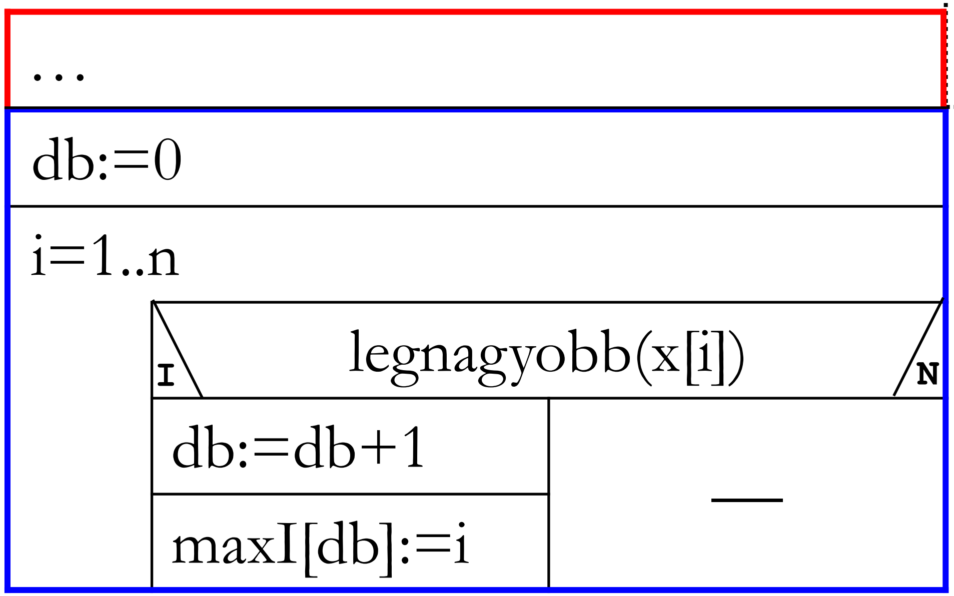


Maximumkiválasztás+kiválogatás

Algoritmus:

```
Uf: (,maxért)=MAX(i=1..n,x[i]) és
    (db,maxI)=KIVÁLOGAT(i=1..n,legnagyobb(x[i]),i)
```

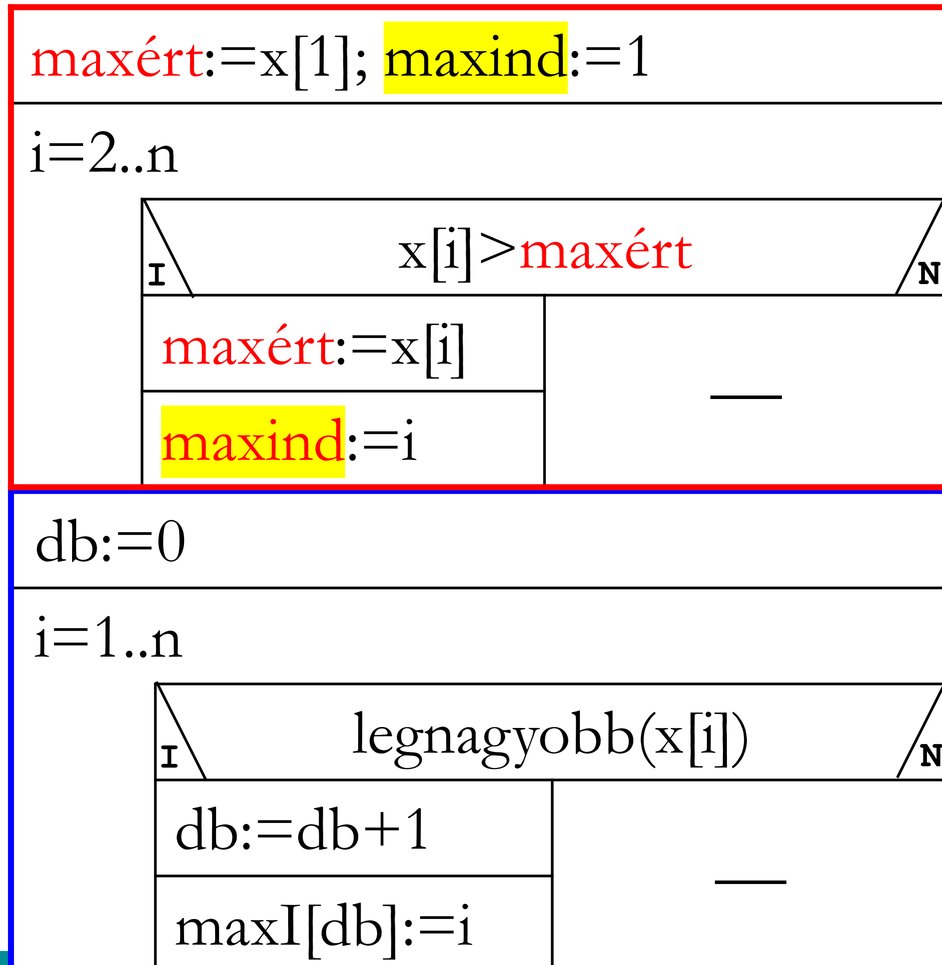
Változó
  i,maxind:**Egész**
  maxért:**TH**



Maximumkiválasztás+kiválogatás

Algoritmus:

Változó
i,maxind:**Egész**
maxért:**TH**



Észrevétel:
Az eredmény helyes, de bántóan nem hatékony.

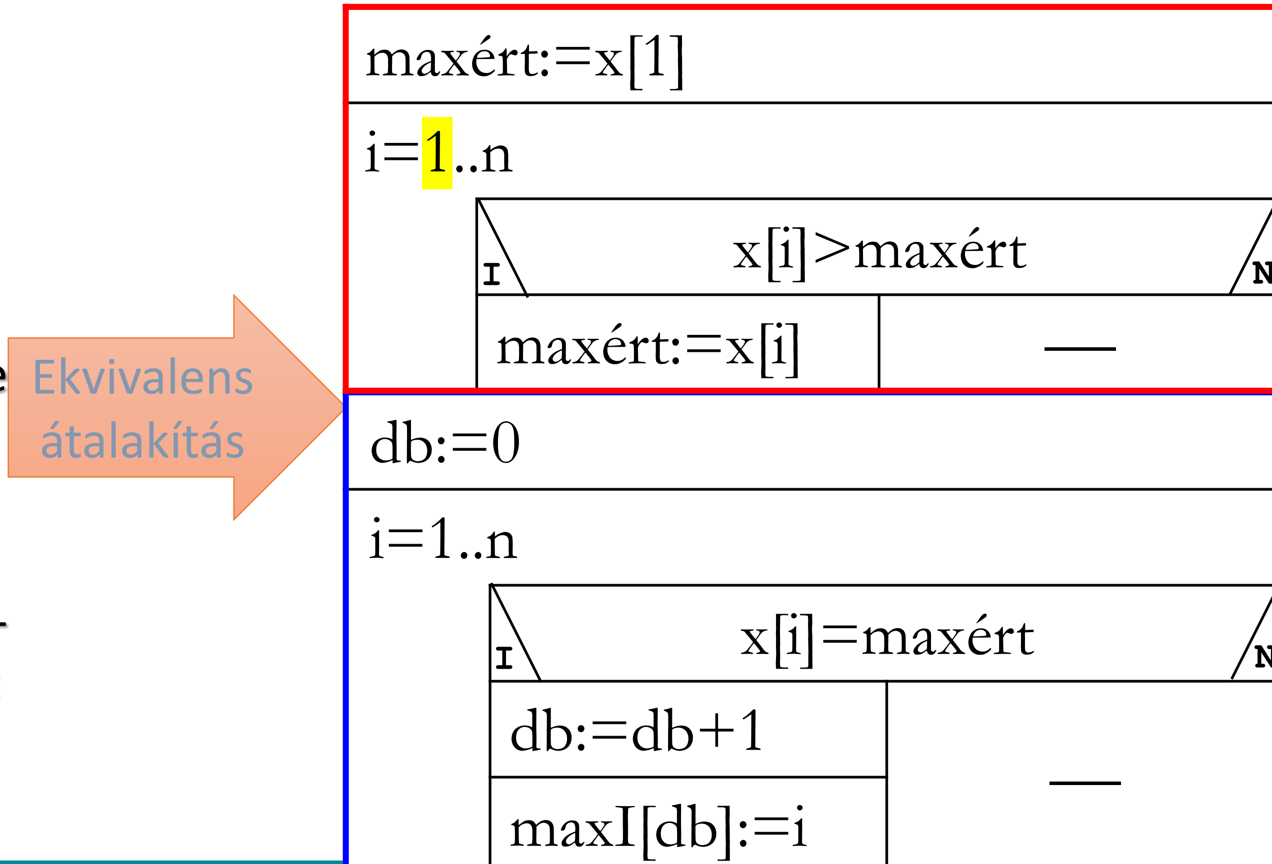
Maximumkiválasztás+kiválogatás

Az algoritmust programtranszformációkkal alakítsuk át!

Változó
i:**Egész**
maxért:**TH**

Észrevétel:

- A fölösleges maxind változót hagyjuk el!
- Függvénytörzset helyezzük a hívás helyére!
- Hozzuk szin-kronba a ciklus-szerve-zéseket: i=1..n.



Maximumkiválasztás+kiválogatás

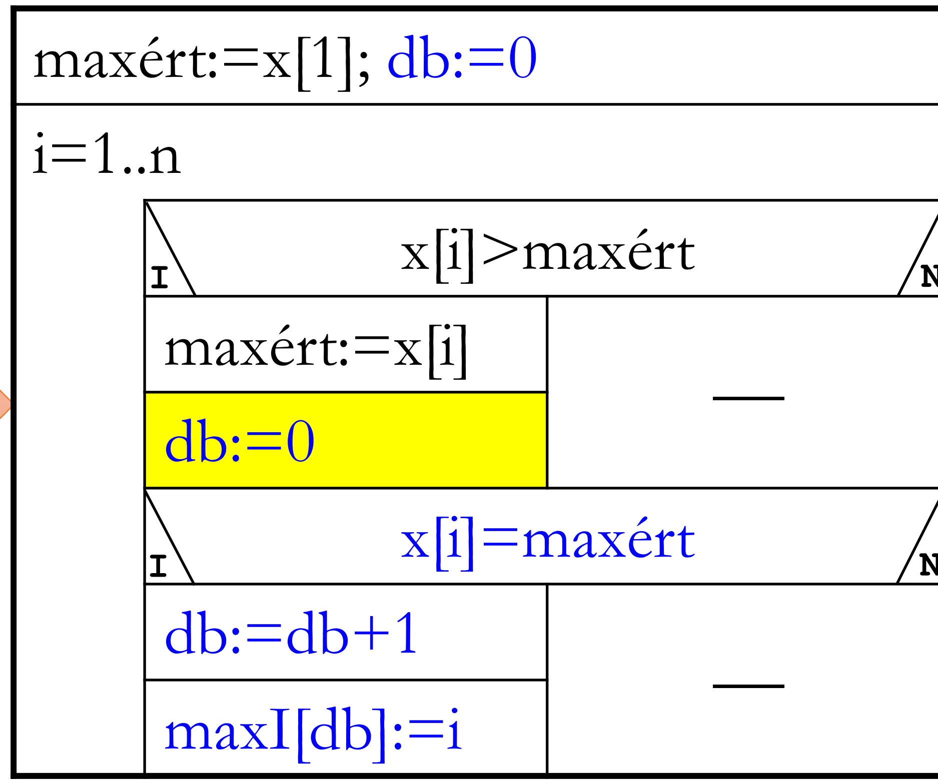
Programtranszformáció: ciklusok és elágazások összevonása, utasítások cseréje

Változó
  i:**Egész**
  maxért:**TH**

Észrevétel:
Az összevonás csak így lehet-séges:

- ha a maxért megváltozik, akkor db nullázandó,
- a 2. feltétel-vizsgálat ez esetben igaz lesz, és az 1. max helye feljegyződik.

Ekvivalens átalakítás



Maximumkiválasztás+kiválogatás

Programtranszformáció: kizáró feltételű elágazások összevonása

Változó
i:**Egész**
maxért:**TH**

Észrevétel:
A program-transzformáció függetlensége feltétele nem teljesül, de ötletnek jó.
Ez esetben az új maxért-et elsőként fel is kell jegyezni.

Ekvivalens átalakítás

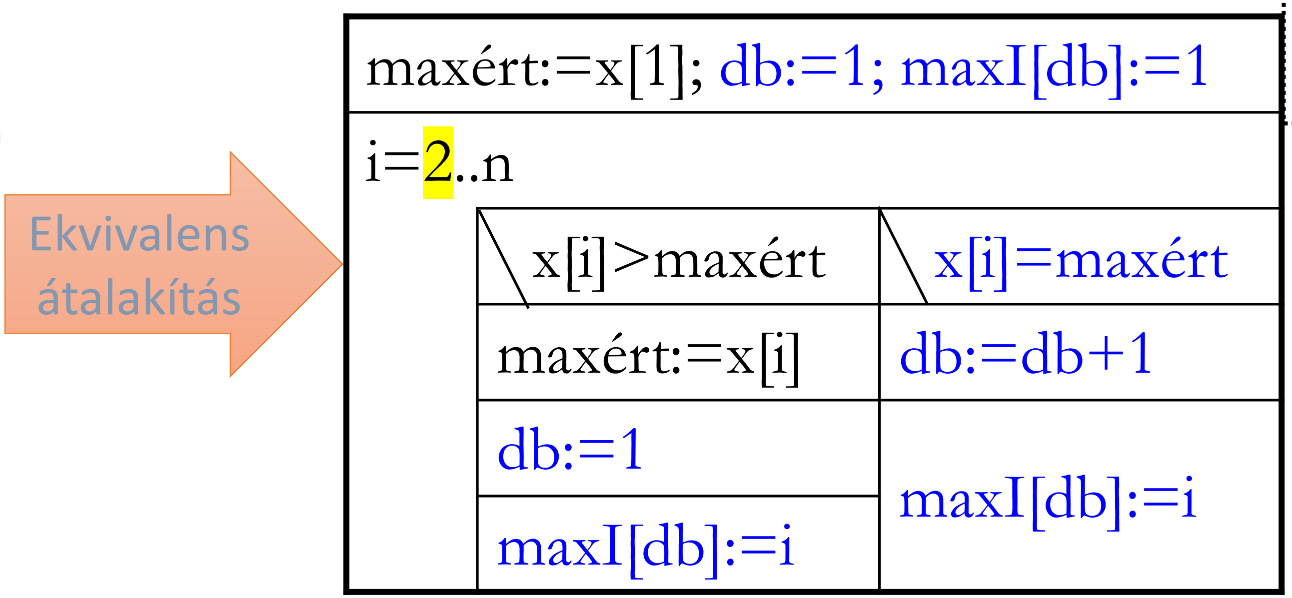
maxért:=x[1]; db:=0

i=1..n

x[i]>maxért	x[i]=maxért
maxért:=x[i]	db:=db+1
db:=1	maxI[db]:=i
maxI[db]:=i	

Maximumkiválasztás+kiválogatás

Észrevétel:
A ciklus indítható 2-től is „okos” inicializálások után.



Változó
i:**Egész**
maxért:**TH**

Eldöntés+megszámolás

Feladat: Van-e egy sorozatban legalább k darab adott tulajdon-ságú elem?

```
Be: n∈N, x∈H[1..n], k∈N
Ki: van∈L
Sa: db∈N
Ef: k>0
Uf: db=DARAB(i=1..n,T(x[i])) és van=db≥k
```

megszámolás

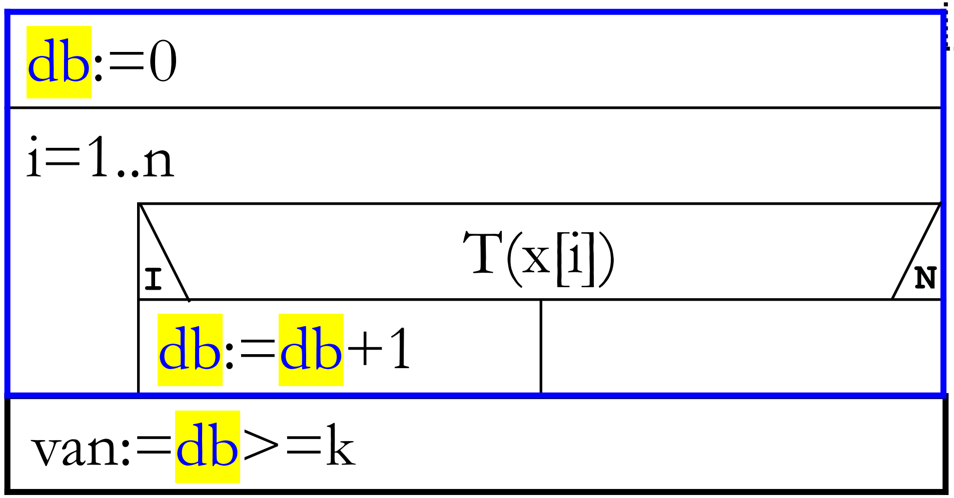
~~eldöntés~~

Eldöntés+megszámolás

Algoritmus:

```
Uf: db=DARAB(i=1..n,T(x[i])) és van=db≥k
```

Változó
  i,db:Egész



Észrevétel:
Helyes, de nem hatékony megoldás!
Ha már találtunk K darab adott tulajdonságút, akkor ne nézzük tovább!

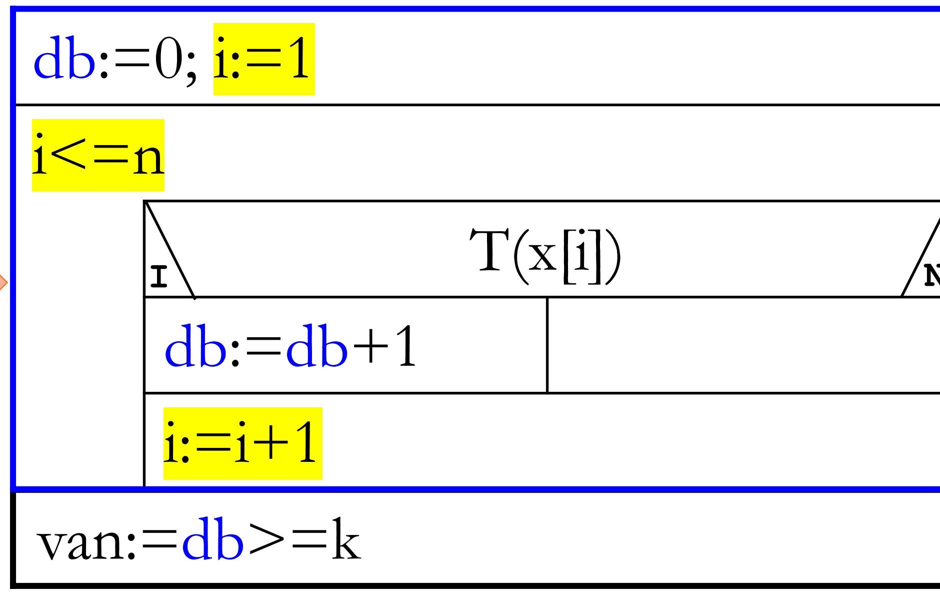
Eldöntés+megszámolás

Az algoritmust programtranszformációkkal alakítsuk át!

Észrevétel:
Teremtsünk lehetőséget arra, hogy „időben” kiléphessünk a ciklusból!
Alakítsuk át a számlálós ciklust feltételes ciklussá!

Ekvivalens átalakítás

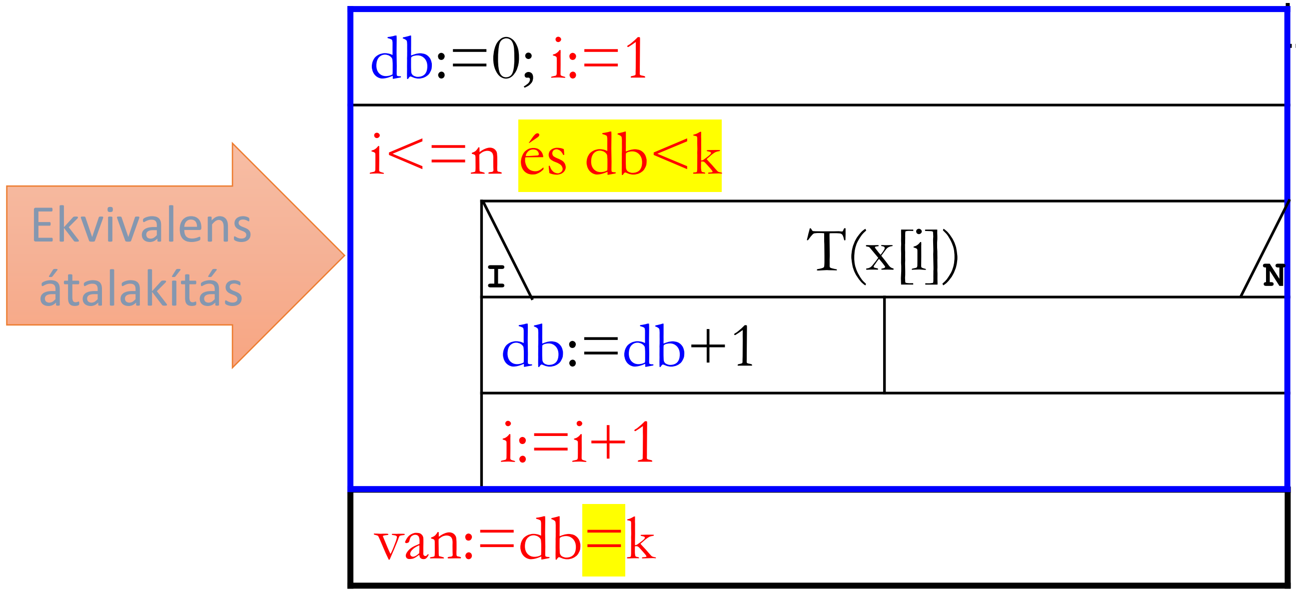
Változó
i,db:**Egész**



Eldöntés+megszámolás

Változó
i,db:**Egész**

Észrevétel:
Bővítsük a cik-lusfeltételt a kívánttal!



Megjegyzés: ehhez „illeszkedő” utófeltétel:

`Uf: van=VAN(i=1..n,DARAB(j=1..i,T(x[j])=k).`

Igaz, ebből is csak programtranszformációkkal nyerhető a fenti algoritmus.

Dinamikus tömb

Statikus tömb

- Eddig statikus tömbökkel dolgoztunk
  - Futás során fix a mérete
    - előre lefoglalni MAXN hosszúságúra
    - n beolvasása után lefoglalni
  - A bemeneti tömböknél ez még jó is
  - A kimeneti tömböknél azonban kényelmetlen
    - ld. kiválogatás

Példa – kitűnő tanulók
visszavezetés

Adjuk meg egy osztály kitűnő tanulóit!

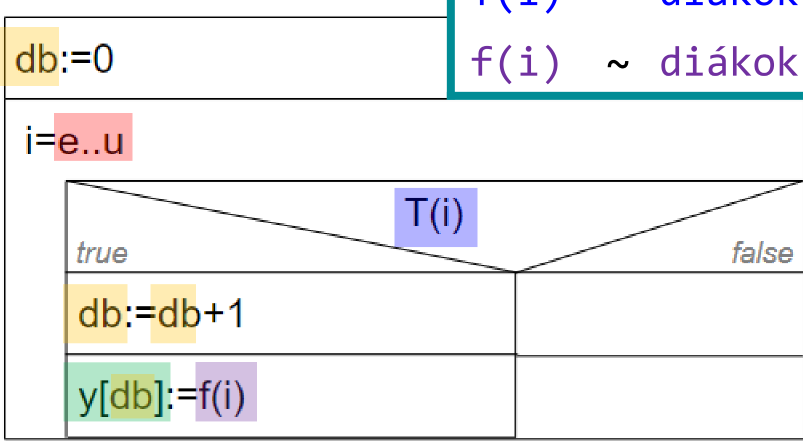
Feladatsablon

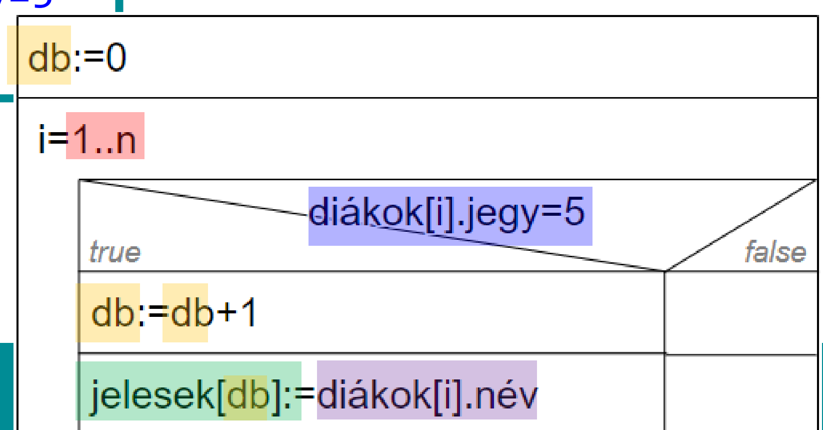
```
Be: e∈Z, u∈Z

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





Statikus tömb kiválogatás

```
struct Diak {
  public string nev;
  public int jegy;
}
static void Main(string[] args) {
  // dekl: spec be + ki
  int n; Diak[] diakok;
  int db; string[] jelesek;
  // beolvasás: spec be
  Console.Write("n = ");
  int.TryParse(Console.ReadLine(), out n);
  diakok = new Diak[n];
  jelesek = new string[n];
  for (int i = 1; i <= n; i++) {
    Console.Write("{0}. nev = ", i);
    diakok[i - 1].nev = Console.ReadLine();
    Console.Write("{0}. jegy = ", i);
    int.TryParse(Console.ReadLine(), out diakok[i - 1].jegy);
  }
```

jelesek db-ig lesz feltöltve, de n elem van lefoglalva, az esetleges maximum

```
  // feldolgozás: stuki
  db = 0;
  for (int i = 1; i <= n; i++) {
    if (diakok[i - 1].jegy == 5) {
      db = db + 1;
      jelesek[db - 1] = diakok[i - 1].nev;
    }
  }
  // kiírás
  Console.WriteLine("{0} db jeles tanuló:", db);
  for (int i = 1; i <= db; i++) {
    Console.WriteLine(jelesek[i - 1]);
  }
}
```

DEMO vagy házi feladat: függvényekre átírni!

Dinamikus tömb

- A dinamikus tömb változó elemszámú sorozatok ábrázolására szolgál
  - Mérete futás közben igény szerint változtatható.
  - A kiválogatásnál éppen erre van szükségünk
- Specifikáció

  Ki: ~~db∈N,~~ y∈N[1..~~db~~]
  Uf: hossz(y)=… és ∀i∈[1..hossz(y)]:(T(y[i]))
- Algoritmus

  Változó y:Tömb[1..~~db~~: Egész]

  Jelentése:
  - kezdőcímet kap, és 0 mérettel rendelkezik.
  - Új műveletek: hossz, Végére.
- Kód

```
List<int> y = new List<int>();
y.Add(1);
y.Add(2);
Console.WriteLine(y[0]);     // 1
Console.WriteLine(y.Count); // 2
```

Példa – kitűnő tanulók
visszavezetés

Adjuk meg egy osztály kitűnő tanulóit!

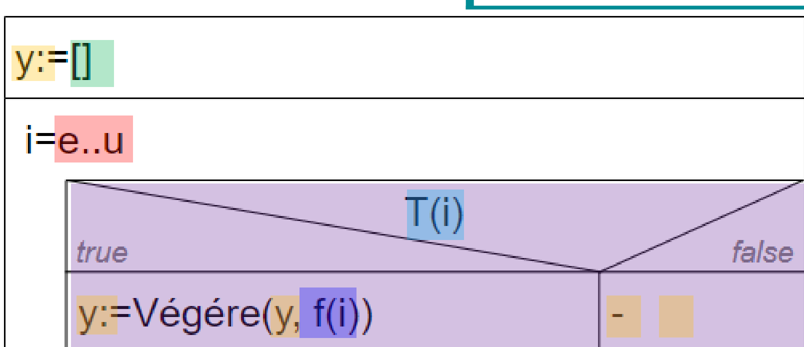
Feladatsablon

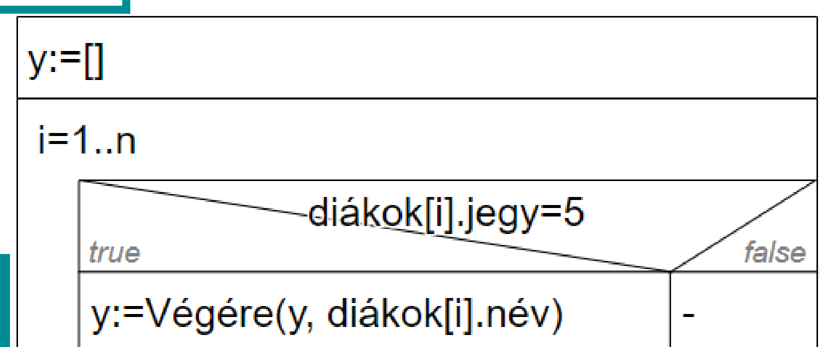
```
Be: e∈Z, u∈Z

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





Dinamikus tömb kiválogatás

DEMO vagy házi feladat: függvényekre átírni!

```
// dekl: spec be + ki
Diak[] diakok;
List<string> jelesek = new List<string>();
// beolvasás: spec be
Console.Write("n = ");
int.TryParse(Console.ReadLine(), out int n);
diakok = new Diak[n];
for (int i = 1; i <= n; i++) {
  Console.Write("{0}. nev = ", i);
  diakok[i - 1].nev = Console.ReadLine();
  Console.Write("{0}. jegy = ", i);
  int.TryParse(Console.ReadLine(), out diakok[i - 1].jegy);
}
// feldolgozás: stuki
jelesek.Clear(); // db = 0;
for (int i = 1; i <= diakok.Length; i++) {
  if (diakok[i - 1].jegy == 5) {
    jelesek.Add(diakok[i - 1].nev);
  }
}
// kiírás
Console.WriteLine("{0} db jeles tanuló van:", jelesek.Count);
for (int i = 1; i <= jelesek.Count; i++) {
  Console.WriteLine(jelesek[i - 1]);
}
```

jelesek igény szerint lesz feltöltve

Mátrixok

egyelőre trükkösen

Mátrix

- **Tömb:** azonos funkciójú elemek *egyirányú* sorozata
  - egy index egy elem kiválasztásához, pl. `x[i]`

x	
1	-4
2	2
3	5

- **Mátrix:** azonos funkciójú elemek *kétirányú* sorozata
  - két index egy elem kiválasztásához, pl. `x[i,j]`
  - specifikáció: `n∈N, m∈N, x∈Z[1..n,1..m]`
  - algoritmus: `x:Tömb[1..n,1..m:Egész]`
  - kód: `int[,] x = new int[n, m];`

x	1	2	3
1	-4	3	2
2	2	10	11
3	5	4	-5

Példa

Példa:

3			8		1			2
2		1		3		6		4
			2		4			
8		9				1		6
	6						5	
7		2				4		9
			5		9			
9		4		8		7		5
6			1		7			3

→ db=3

Feladat:

Hány 5-öst írtunk már be egy sudoku táblázatba?

Olyan feladat, amelyben egy sémát kell alkalmazni egy mátrixra.

Specifikáció:

```
Be: s∈N[1..9,1..9]
Ki: db∈N
Ef: ∀i∈[1..9]:(∀j∈[1..9]:(0<=s[i,j]<=9))
```

~~Uf: db=**DARAB**(i=1..9,j=1..9,s[i,j]=5)~~

Nincs ilyen rövidítésünk.

```
Uf: db=DARAB(i=??..??,s[??]=5)
```

Egy futóindex egydimenziós adatszerkezetet kíván. Alakítsuk át a mátrixot sima tömbbé!

2D → 1D trükk

- Ábrázoljuk a kétdimenziós négyzetes mátrixot egydimenzióban, pl. sorfolytonosan!

$$\begin{array}{c|ccc|}
 & 1 & 2 & 3 \\
\hline
1 & 1 & 2 & 3 \\
2 & 4 & 5 & 6 \\
3 & 7 & 8 & 9 \\
\hline
\end{array}$$

1 2 3 4 5 6 7 8 9

```
i=(k-1) div 3 + 1
j=(k-1) mod 3 + 1
```

```
k=(i-1)*3+j
```

Pl. k=7 → i=(7-1) div 3 + 1=3, j=(7-1) mod 3 + 1=1

Pl. i=2, j=3 → k=(2-1)*3+3=6

Példa

Példa:

3			8		1			2
2		1		3		6		4
			2		4			
8		9				1		6
	6						5	
7		2				4		9
			5		9			
9		4		8		7		5
6			1		7			3

→ db=3

Feladat:

Hány 5-öst írtunk már be egy sudoku táblázatba?

Specifikáció:

```
Be: s∈N[1..9,1..9]
Ki: db∈N
Ef: ∀i∈[1..9]:(∀j∈[1..9]:(0<=s[i,j]<=9))
Uf: db=DARAB(k=1..9*9,s[(k-1) div 9 + 1,
                         (k-1) mod 9 + 1]=5)
```

Mátrix

Hány 5-öst írtunk már be egy sudoku táblázatba?

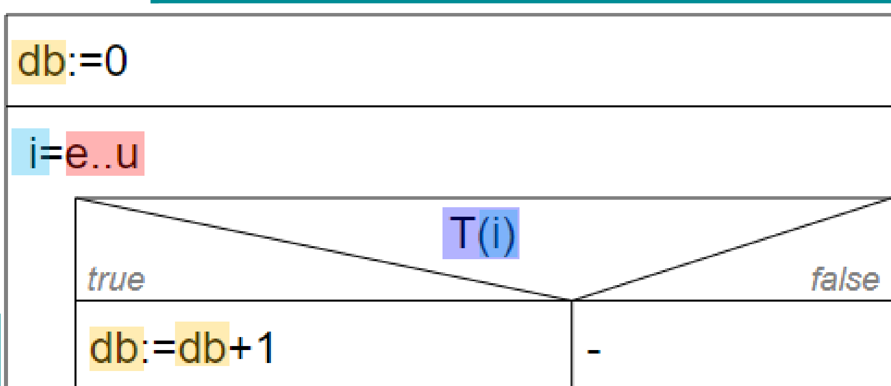
Feladatsablon

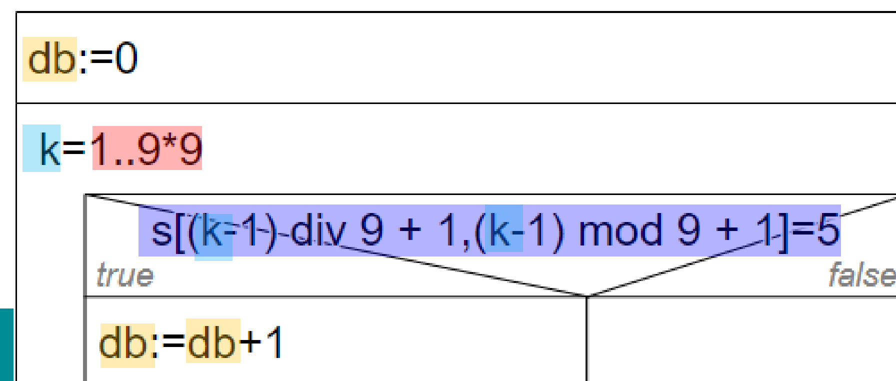
```
Be: e∈Z, u∈Z
Ki: db∈N
Ef: -

Uf: db=DARAB(i=e..u,
             T(i))
```

Sudoku

```
Be: s∈N[1..9,1..9]
Ki: db∈N
Ef: ∀i∈[1..9]:(∀j∈[1..9]:(
                  0<=s[i,j]<=9))
Uf: db=DARAB(k=1..9*9,
             s[(k-1) div 9 + 1,
               (k-1) mod 9 + 1]=5)
```

i	~	k
e..u	~	1..9*9
T(i)	~	s[(k-1) div 9 + 1,(k-1) mod 9 + 1]=5





Mátrix

DEMO vagy házi feladat: függvényekre átírni!

```
// deklaráció
int[,] s = new int[9, 9] {
  {3, 0, 0, 8, 0, 1, 0, 0, 2 },
  {2, 0, 1, 0, 3, 0, 6, 0, 4 },
  {0, 0, 0, 2, 0, 4, 0, 0, 0 },
  {8, 0, 9, 0, 0, 0, 1, 0, 6 },
  {0, 6, 0, 0, 0, 0, 0, 5, 0 },
  {7, 0, 2, 0, 0, 0, 4, 0, 9 },
  {0, 0, 0, 5, 0, 9, 0, 0, 0 },
  {9, 0, 4, 0, 8, 0, 7, 0, 5 },
  {6, 0, 0, 1, 0, 7, 0, 0, 3 }
};
int db;
// feldolgozás
db = 0;
for (int k = 1; k <= 81; k++) {
  if (s[(k - 1) / 9 + 1 - 1, (k - 1) % 9 + 1 - 1] == 5) {
    db = db + 1;
  }
}
// kiírás
Console.WriteLine("{0} db 5-ös van", db);
```

Mátrix

DEMO vagy házi feladat: függvényekre átírni!

```
// deklaráció
int[,] s = new int[9, 9];
int db;
// beolvasás
for (int i = 1; i <= 9; i++) {
  string[] sortomb = Console.ReadLine().Split(" ");
  for (int j = 1; j <= 9; j++) {
    int.TryParse(sortomb[j - 1], out s[i - 1, j - 1]);
  }
}
// feldolgozás
db = 0;
for (int k = 1; k <= 81; k++) {
  if (s[(k - 1) / 9 + 1 - 1, (k - 1) % 9 + 1 - 1] == 5) {
    db = db + 1;
  }
}
// kiírás
Console.WriteLine("{0} db 5-ös van", db);
```

Összefoglalás

Összefoglalás

- Több programozási minta használata
  - bizonyos feladatok megoldásához több programozási minta használata szükséges
  - ezek egy része, amikor a mintákat egymás után használjuk
  - közbülső segédadatok
  - hatékonyság programtranszformációkkal
- Dinamikus tömb
- Mátrix 2D→1D trükk (nem sokáig)

Ellenőrző kérdések

Ellenőrző kérdések

1. Programtranszformációkkal lássa be a `MAX(i=1..n,x[i])` → → `MAX(i=1..n,f(x[i]))` átalakítás alkalmazhatóságát!
2. Az eldöntés utófeltételeként írhatjuk: $\mathrm{van} = \bigvee_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{T}(\mathrm{x}[\mathrm{i}])$. Ennek algoritmusa számlálós ciklust tartalmaz:

van:=hamis [kezdetben még nem találtunk megfelelőt]	
i=1..n	
	van:=van vagy T(x[i]) [i.-ig találtunk-e]

amely nyilvánvalóan nem hatékony.
Vezesse le programtranszformációkkal valamelyik korábban tanult algoritmust!