ELTE | IK

PROGRAMOZÁS

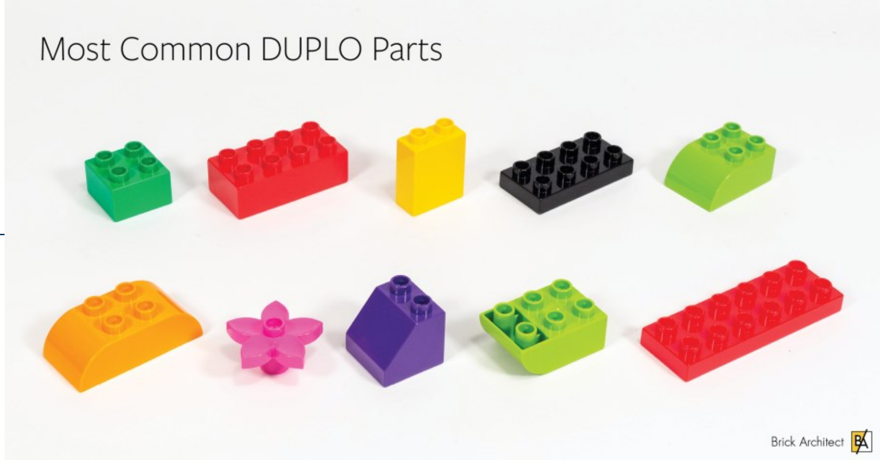
Programtranszformációk

Horváth Győző, 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



Több programozási minta használata

egymás után

Több minta alkalmazása

- **Összetettebb feladatok** nem vezethetők vissza csupán egyetlen programozási mintára
- **Több programozási minta** együttes használata szükséges!
- Egyelőre foglalkozzunk olyan feladatokkal, ahol a mintákat **egymás után** kell alkalmazni!

Maximumkiválasztás+kiválogatás

Feladat: Adott számok sorozata. Add meg az összes maximális elemet.

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

1. lépés: maximális érték meghatározása

A két lépés közötti kapcsolatot egy közbülső segédadat biztosítja

2. minta: maximális elemek indexeinek kiválogatása

Maximumkiválasztás+kiválogatás

Algoritmus:

maxért:=x[1]; maxind:=1
i=2..n

x[i]>maxért (I)	(N)
maxért:=x[i]	—
maxind:=i	

db:=0
i=1..n

x[i]=maxért (I)	(N)
db:=db+1	—
maxI[db]:=i	

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

Észrevétel:
Az eredmény helyes, de bántóan nem hatékony.

Próbáljuk hatékonyabbra írni a kapott algoritmust!

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

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

Bevezető példa

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

Melyik az origótól legmesszebb levő mP pont (p∈Pont[1..n], Pont=X×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]:(
    $\sqrt{p[mP].x^2 + p[mP].y^2}$ >=
    $\sqrt{p[i].x^2 + p[i].y^2}$) 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=$\sqrt{p[mP].x^2 + p[mP].y^2}$

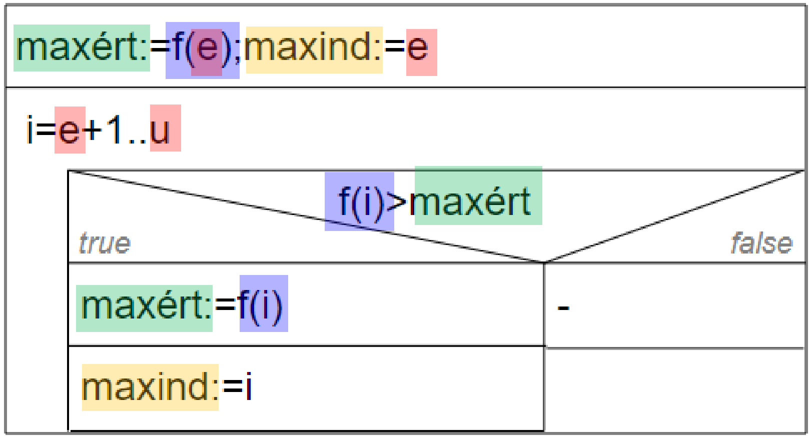
Rövidítve:

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)	~	$\sqrt{p[i].x + p[i].y^2}$

Bevezető példa

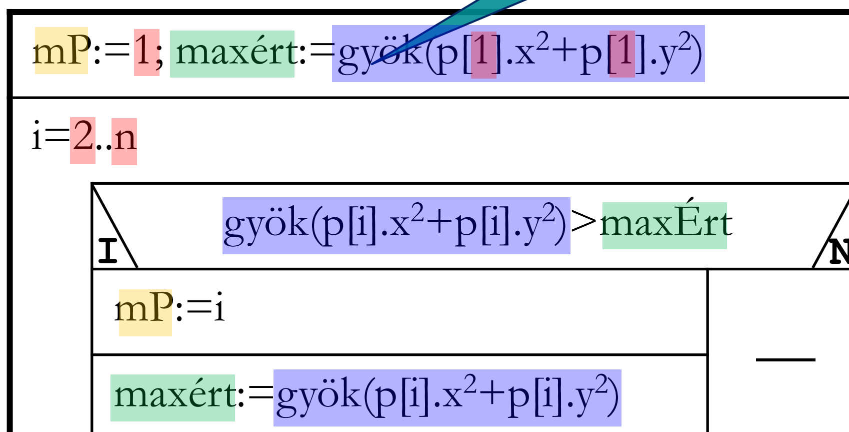


maxind, maxért ~ mP, maxért

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

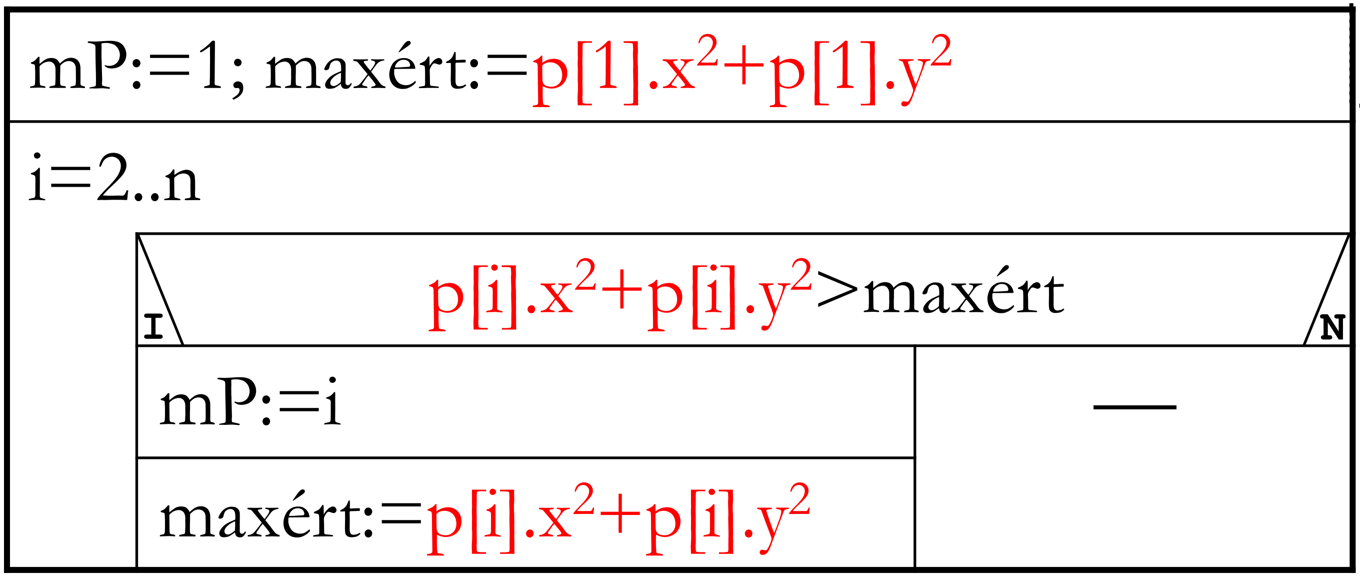
i=2..n

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

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

Bevezető példa

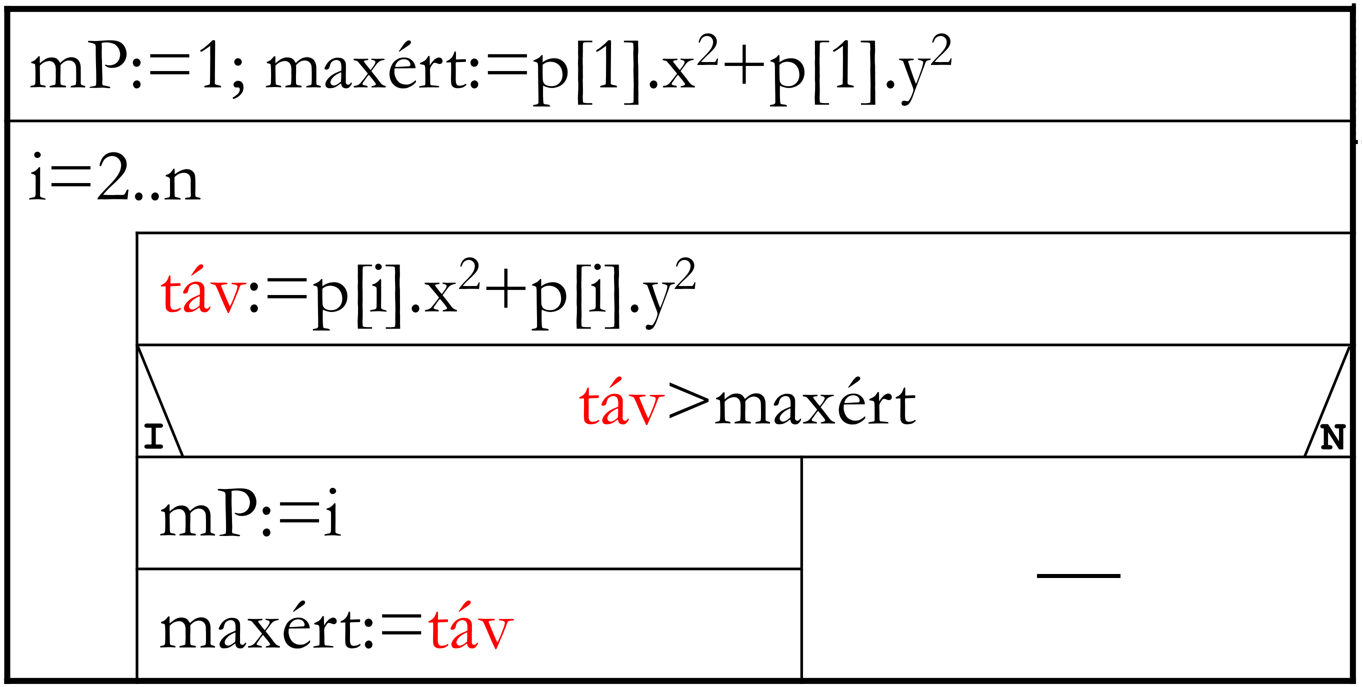
Változó
i:**Egész**
maxért:**Valós**



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:

Változó
segéd:TH

segéd:=a; a:=b; b:=c; c:=segéd

Nevezetes programtranszformációk

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

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

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:

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

Ha

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

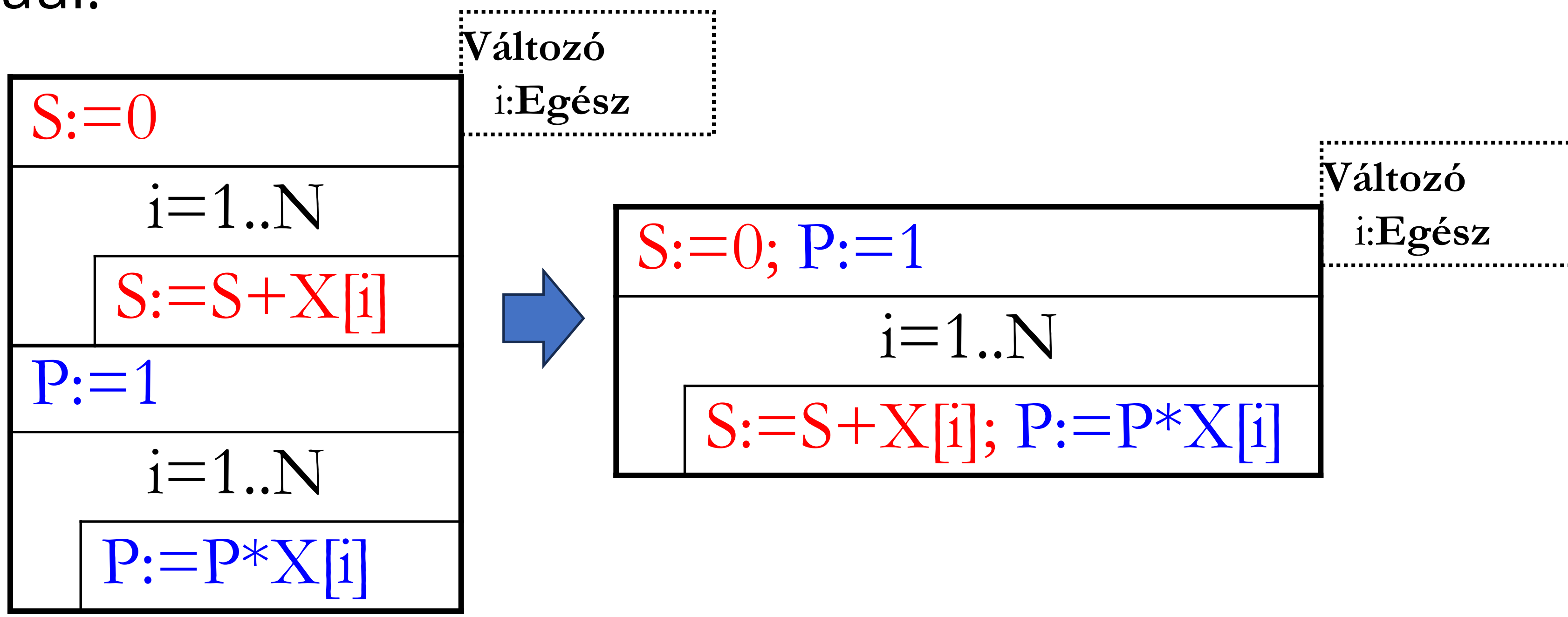
akkor

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

Nevezetes programtranszformációk

Ciklusok összevonása:
Azonos lépésszámú ciklusok összevonhatóak, ha függetlenek egymástól.

Például:

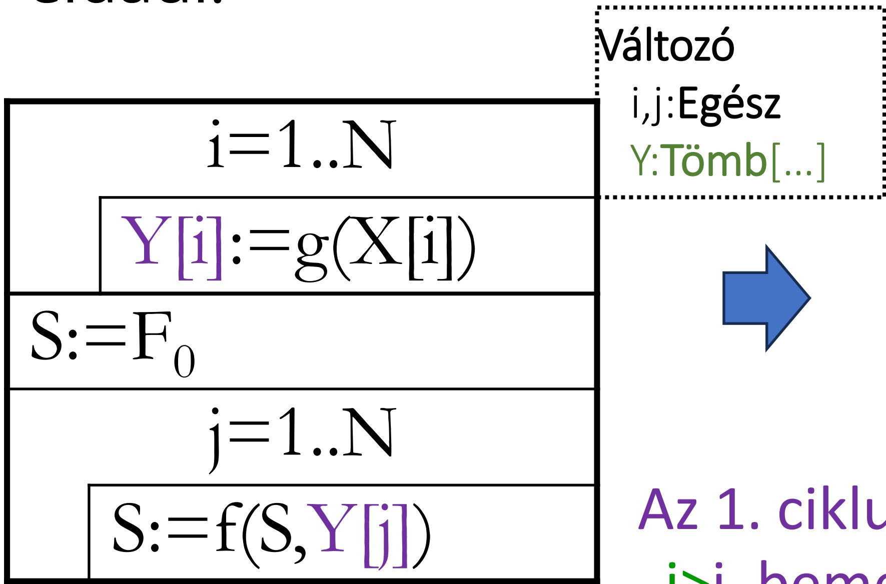


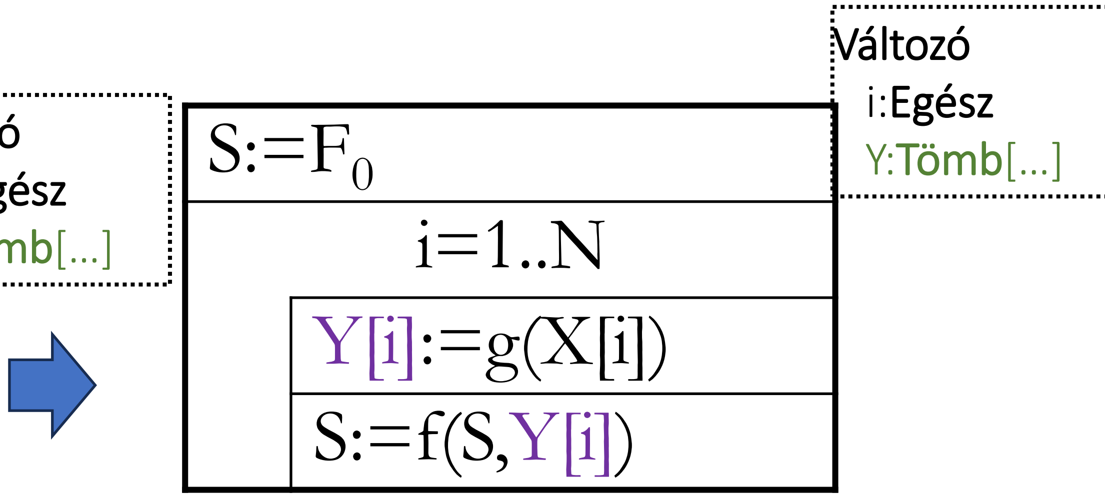
Nevezetes programtranszformációk

Ciklusok összevonása (gyenge függés):

„Gyenge” függés megengedhető.

Például:

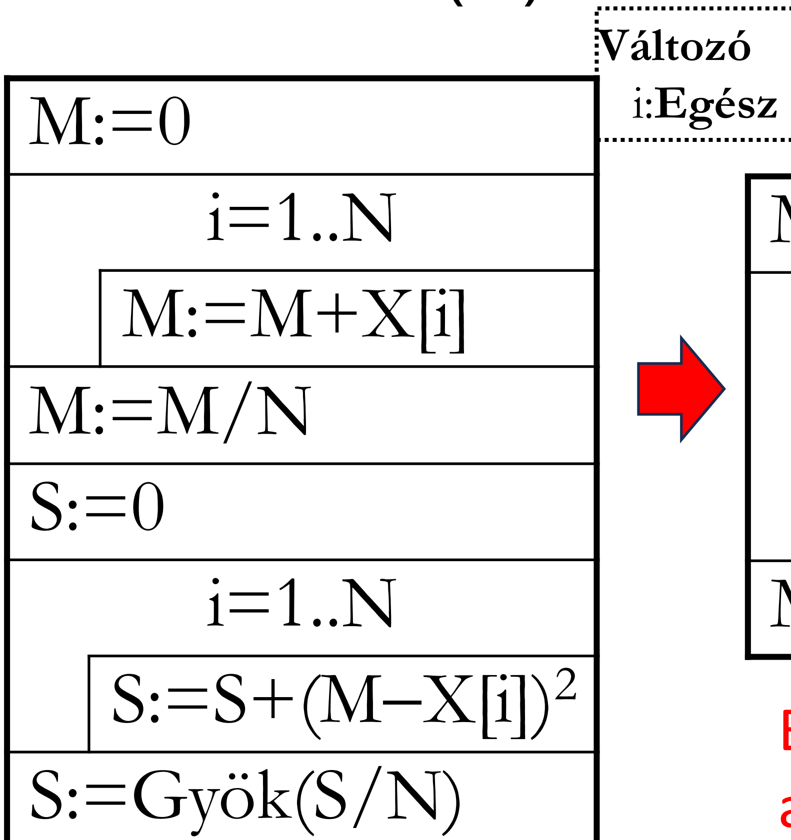


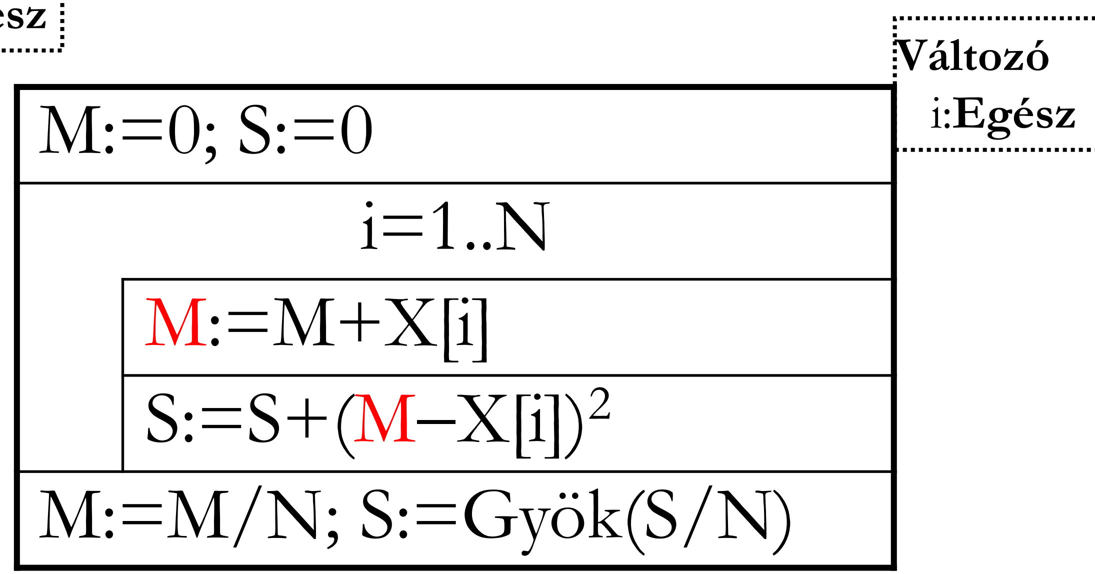


Az 1. ciklusmag „i. kimenete” lehet a 2. ciklus „j>i. bemenete”, de a 2. ciklus „i. kimenete” nem lehet az 1. ciklus „j>i bemenete”.

Nevezetes programtranszformációk

Ciklusok összevonása (ellenpélda):
A várhatóérték (M) és szórás kiszámolása (S):



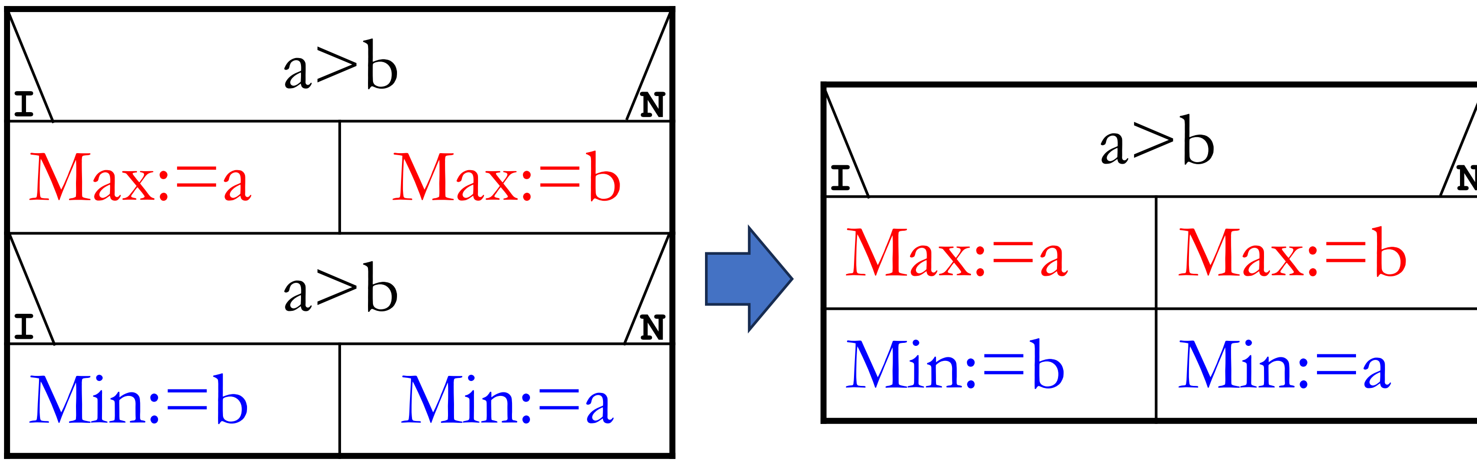


Baj van: M még nem „kész”, amikor felhasználásra kerül.

Nevezetes programtranszformációk

Elágazások összevonása:

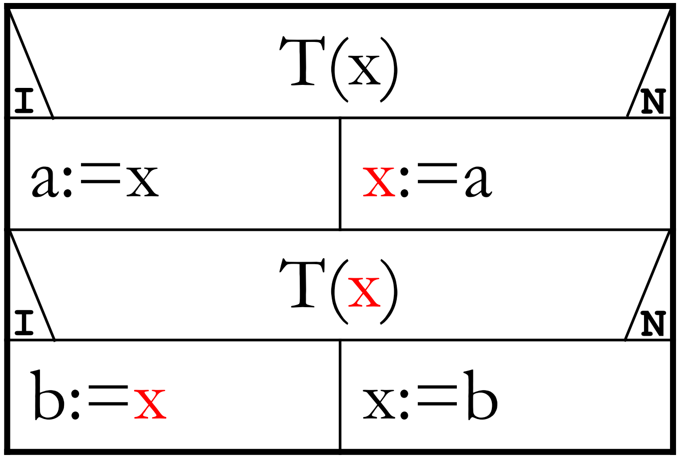
Azonos feltételű elágazások összevonhatóak, ha függetlenek egymástól.

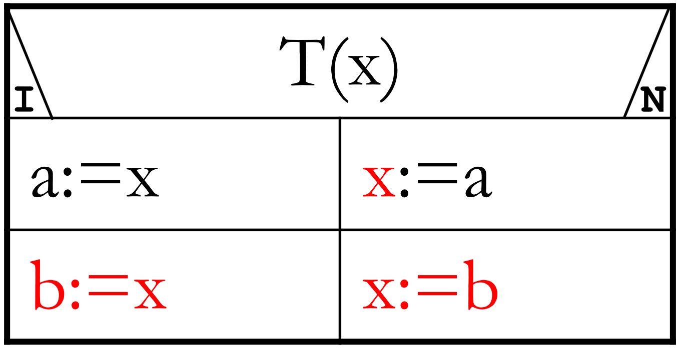


Függetlenek, ha az 1. feltétel egyik ágán sem változik meg sem az ‚a’ , sem a ‚b’ változó (kifejezés). Gondolja meg: mikor nem független a két elágazás, ha ‚feltétel(a,b)’ függvény a közös feltétel?

Nevezetes programtranszformációk

Elágazások összevonása (ellenpélda):



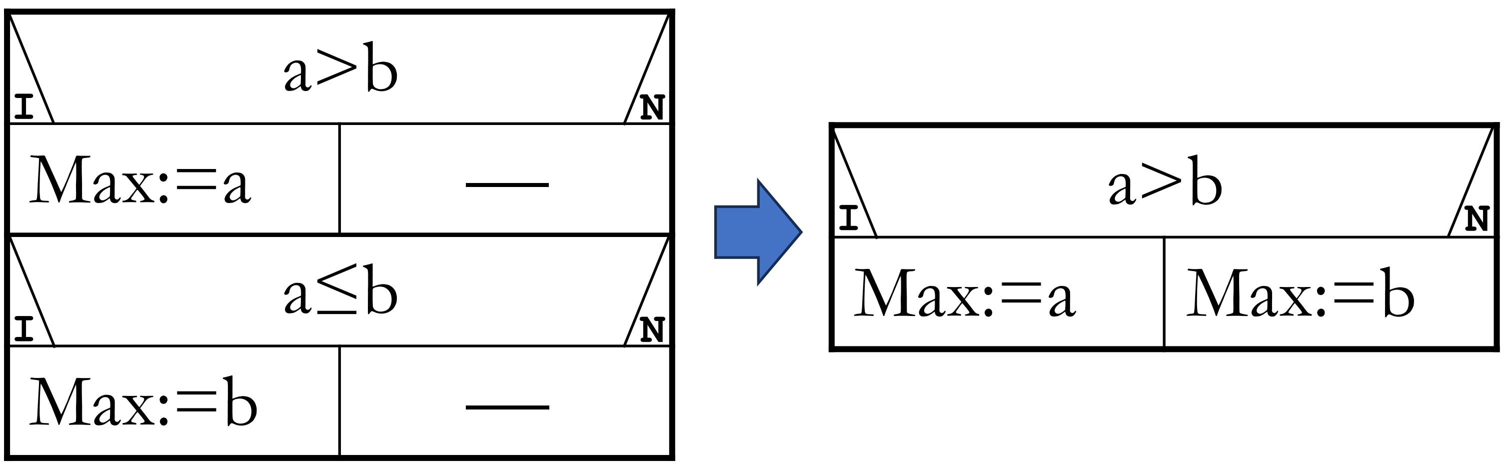


Baj van: x megváltozhat
a második elágazásig.

Nevezetes programtranszformációk

Elágazások összevonása:

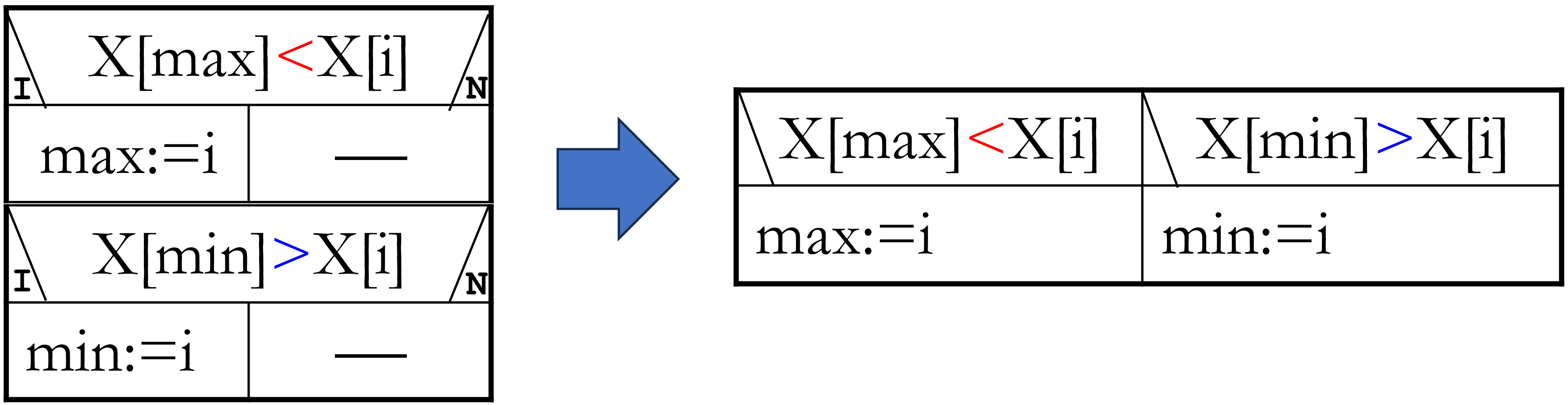
Kizáró feltételű, teljes (egyágú) elágazások is összevonhatók, ha függetlenek egymástól.



Nevezetes programtranszformációk

Elágazások összevonása:

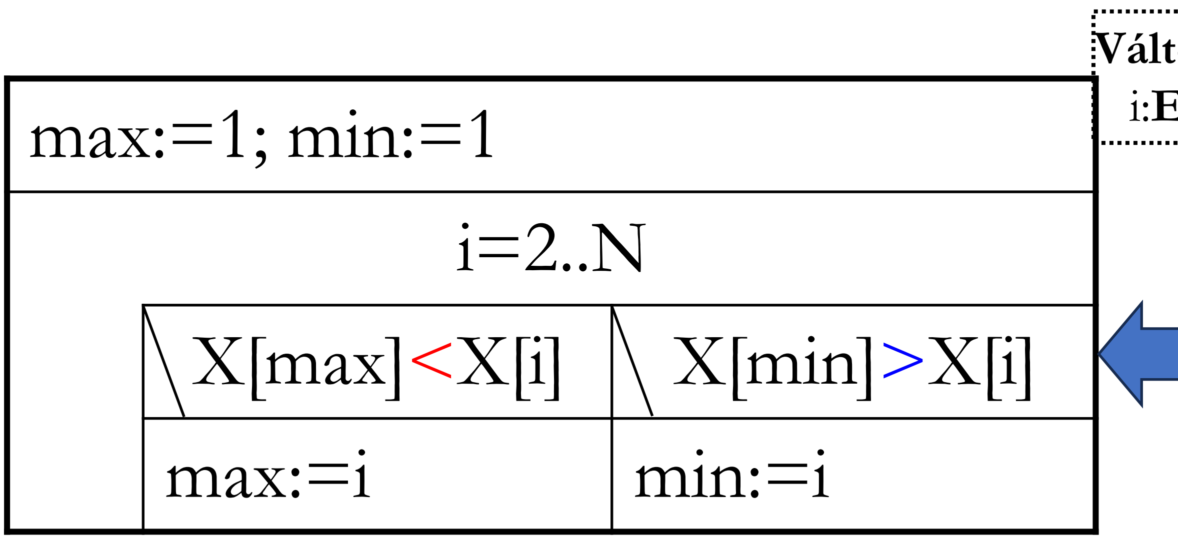
Kizáró feltételű elágazásokkal is összevonhatók, ha függetlenek egymástól.

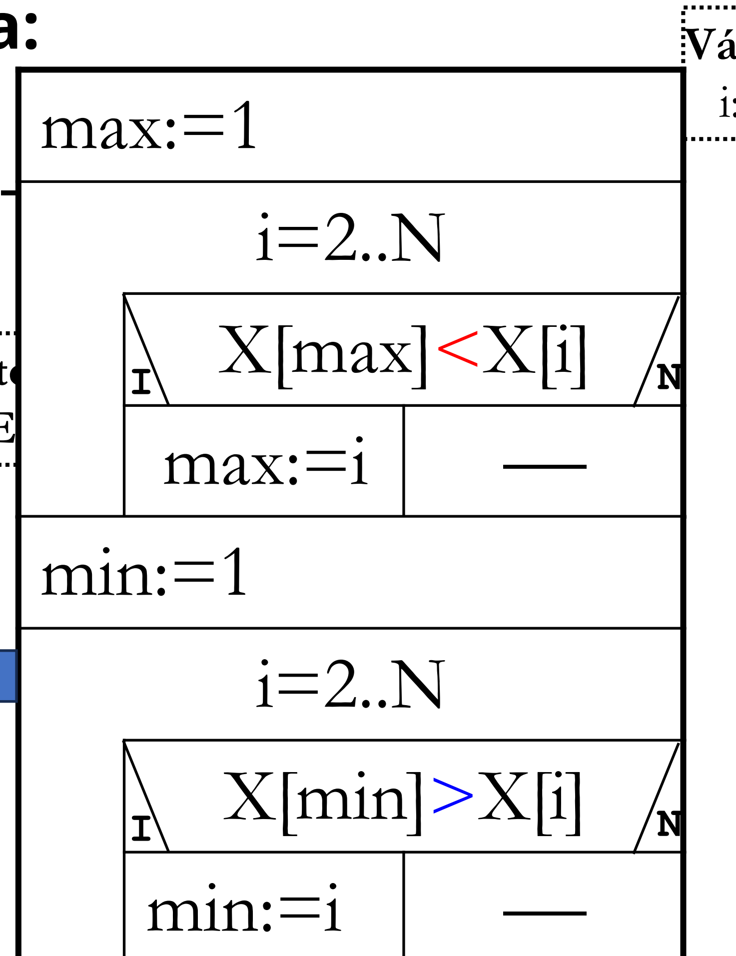


Nevezetes programtranszformációk

Ciklusok és elágazások összevonása:

Azonos lépésszámú ciklusok, bennük kizáró feltételű elágazásokkal is össze-vonhatók, ha függetlenek egymástól.

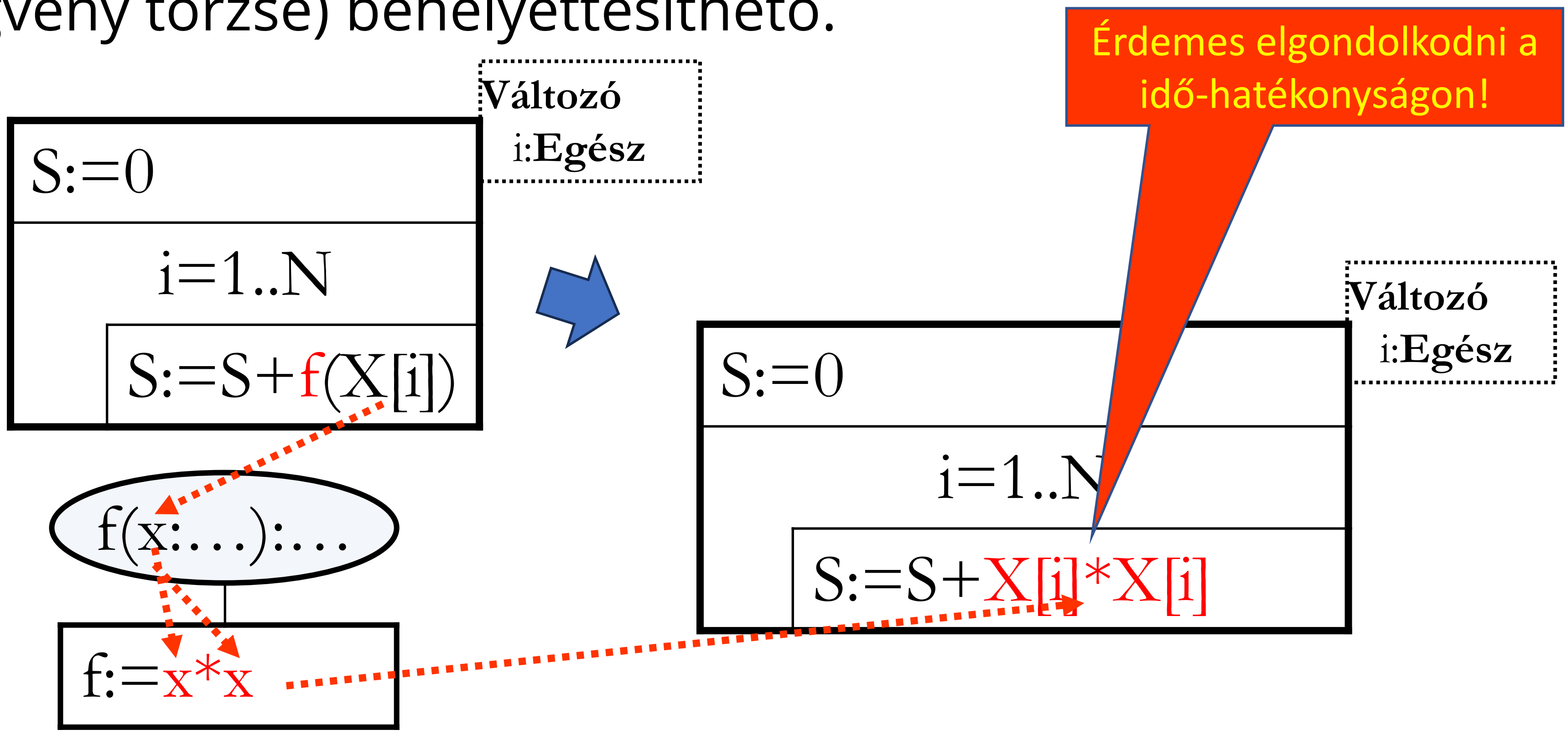




Nevezetes programtranszformációk

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!

Változó
i:Egész

i:=1
i≤N és nem T(X[i])
i:=i+1
Van:=i≤N
…

Változó
i:Egész

i:=1; X[N+1]:=Telem
nem T(X[i])
i:=i+1
Van:=i≤N
…

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)* $\longrightarrow$ *kimenet* *(értékkészlet)*

A bemenet minden tétel esetében legalább 1 sorozat.

Tételek összeépítése elé…

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

- T_0 kimenete (legalább 1) <u>sorozat</u>, 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,
  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. A 2. ciklus **i.** lépésben csak az y[i] kell.

másolás+összegzés

1. programtranszformáció: (gyenge függésű) ciklusok öszevonása

Ekvivalens átalakítás

Változó
i:**Egész**
y:**Tömb**[...]

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

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

másolás+összegzés

2. programtranszformáció: függvénykompozíció

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

Észrevétel:

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

$\leftrightarrow$

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

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:

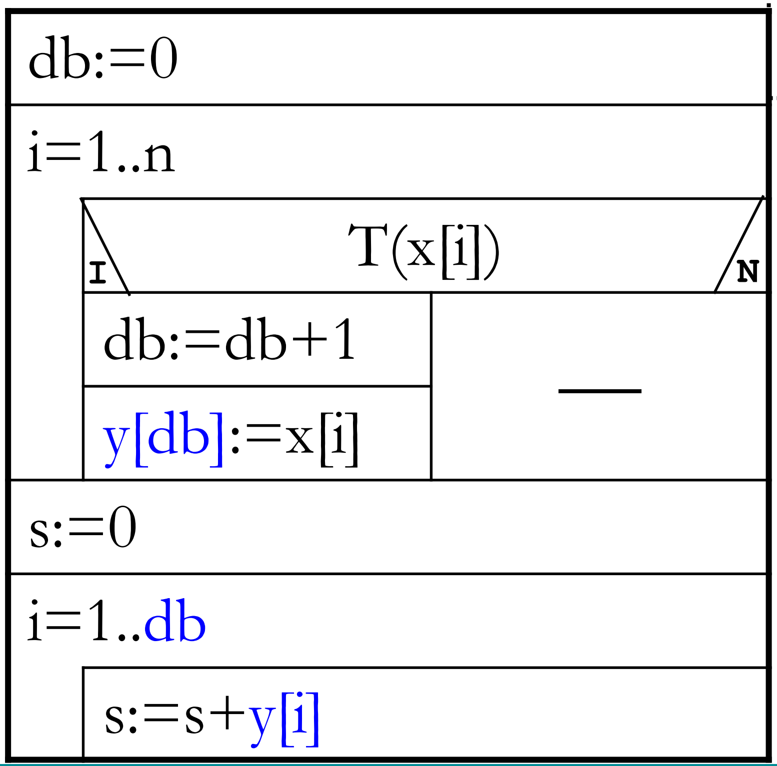
```
Be: n∈N, x∈H[1..n]
Sa: db∈N, y∈H[1..n]
Ki: s∈H2
Ef: -
Uf: (db,y)=KIVÁLOGAT(i=1..n,T(x[i]),x[i]) és
    s=SZUMMA(i=1..db,y[i])
```

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.

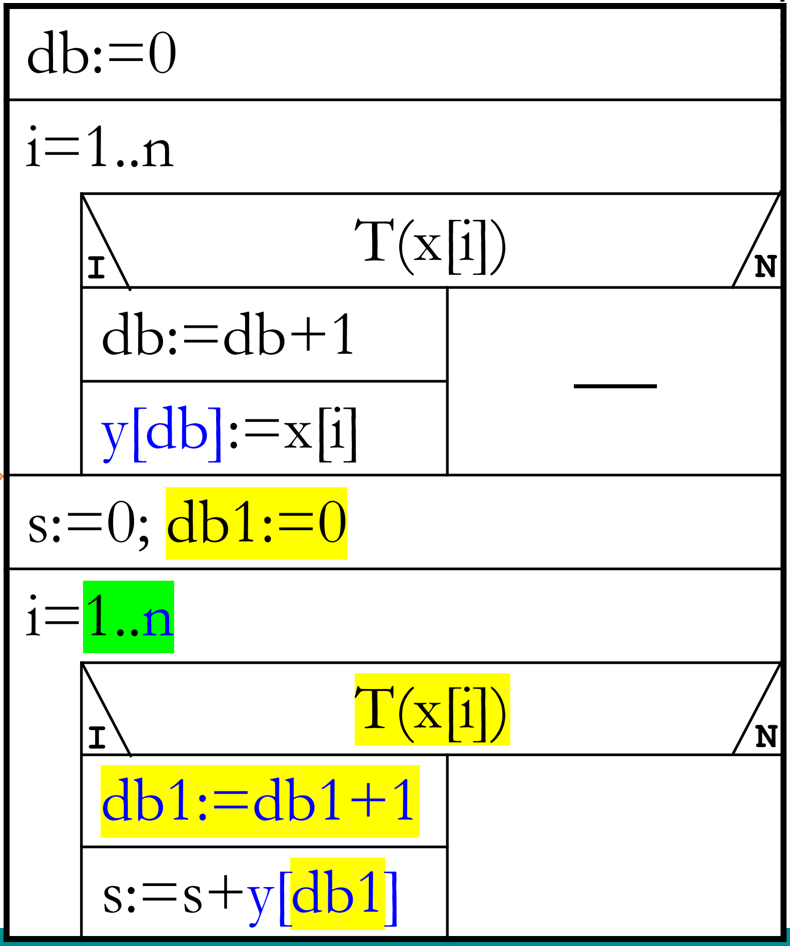
kiválogatás+összegzés

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.

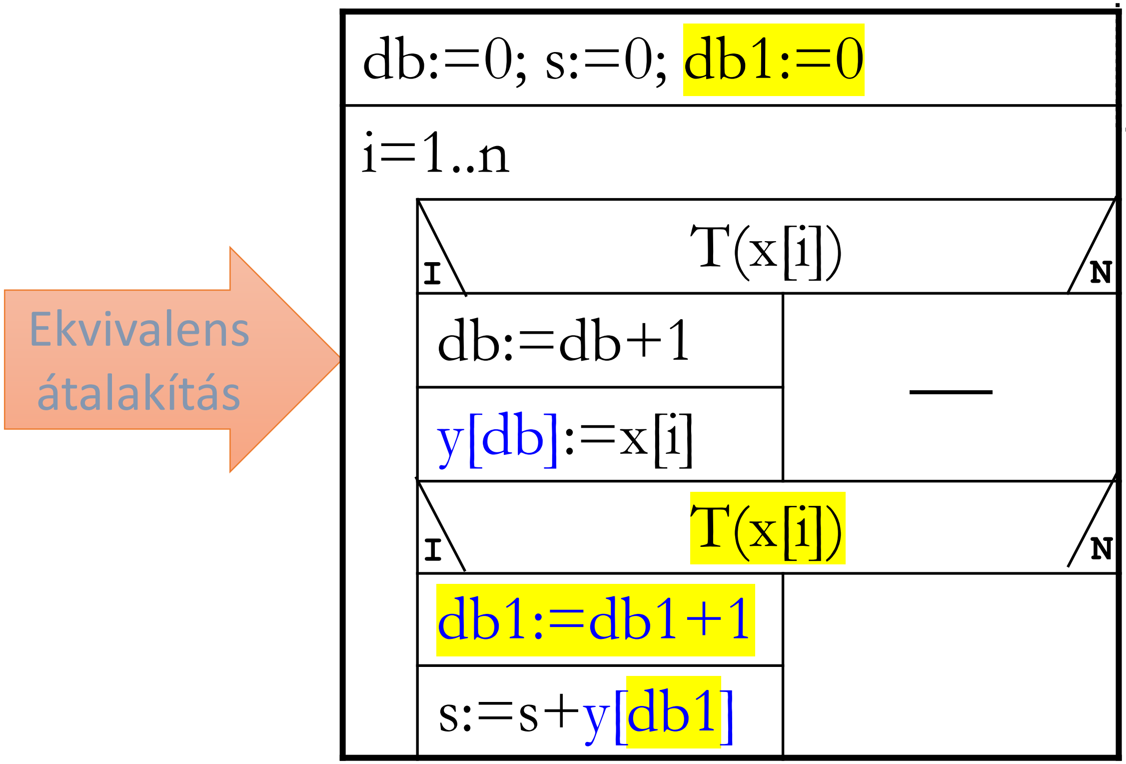
Ekvivalens átalakítás



kiválogatás+összegzés

Programtranszformáció: (gyengén függő) ciklusok összevonása

Változó
i,db,
db1:**Egész**
y:Tömb[…]



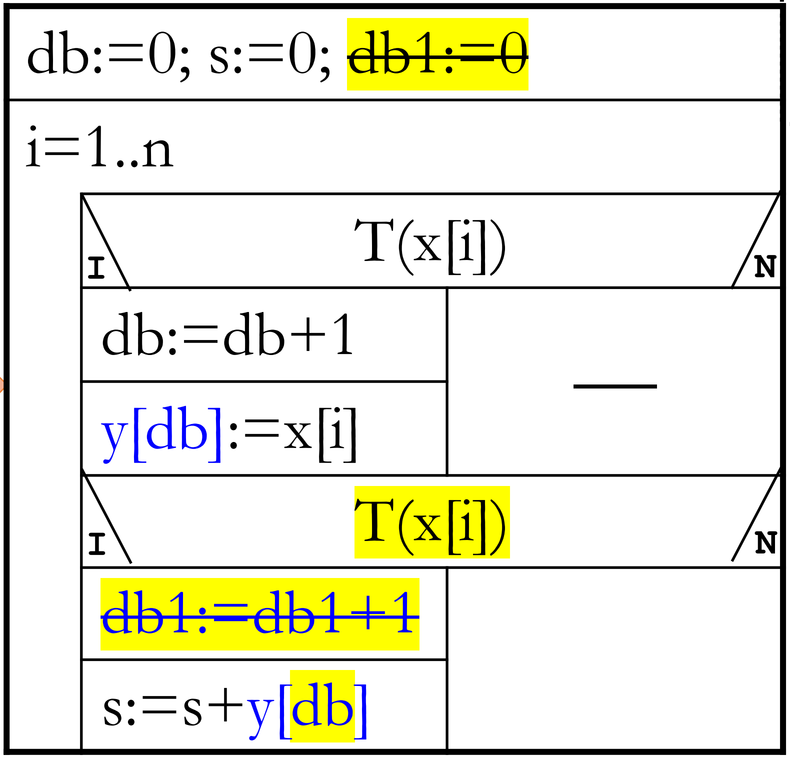
kiválogatás+összegzés

Változó
i,db,
~~db1~~:**Egész**
y:Tömb[…]

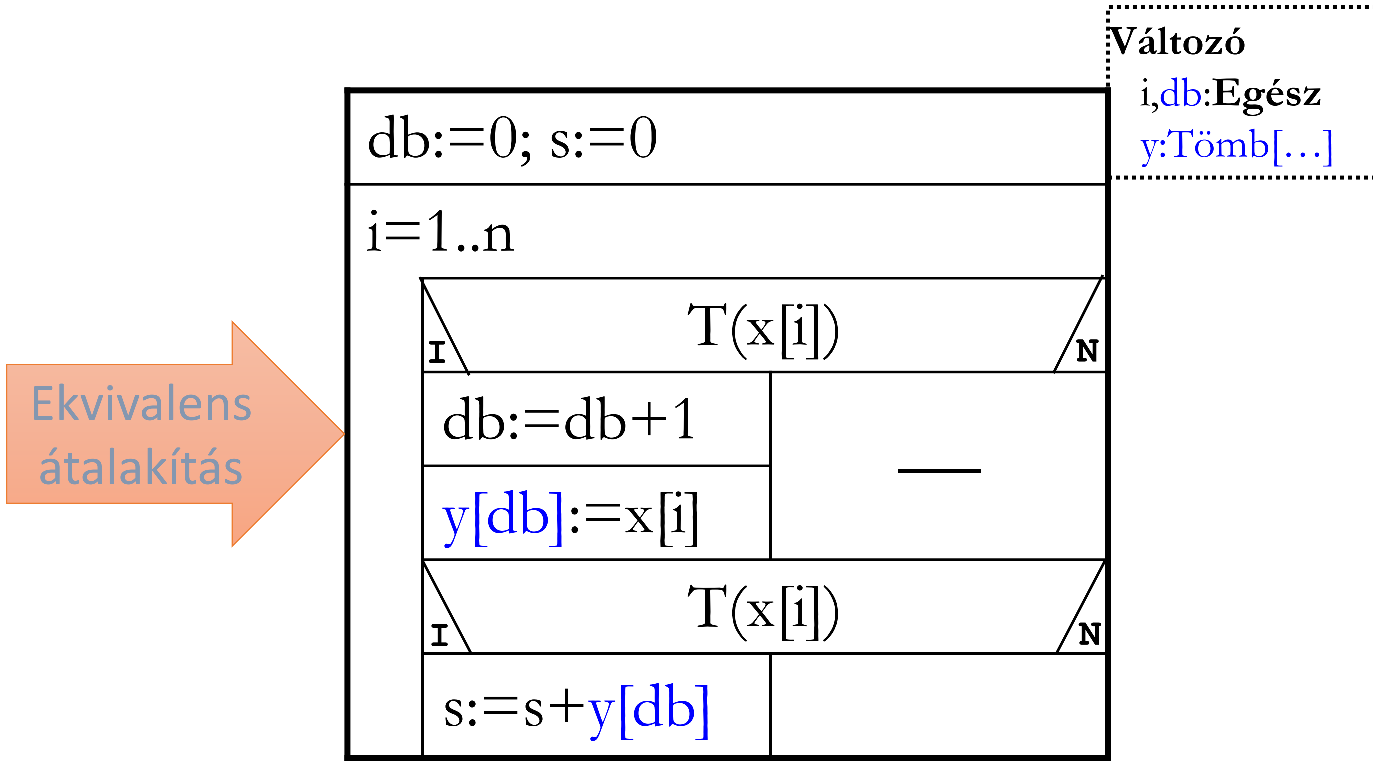
Észrevétel:

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



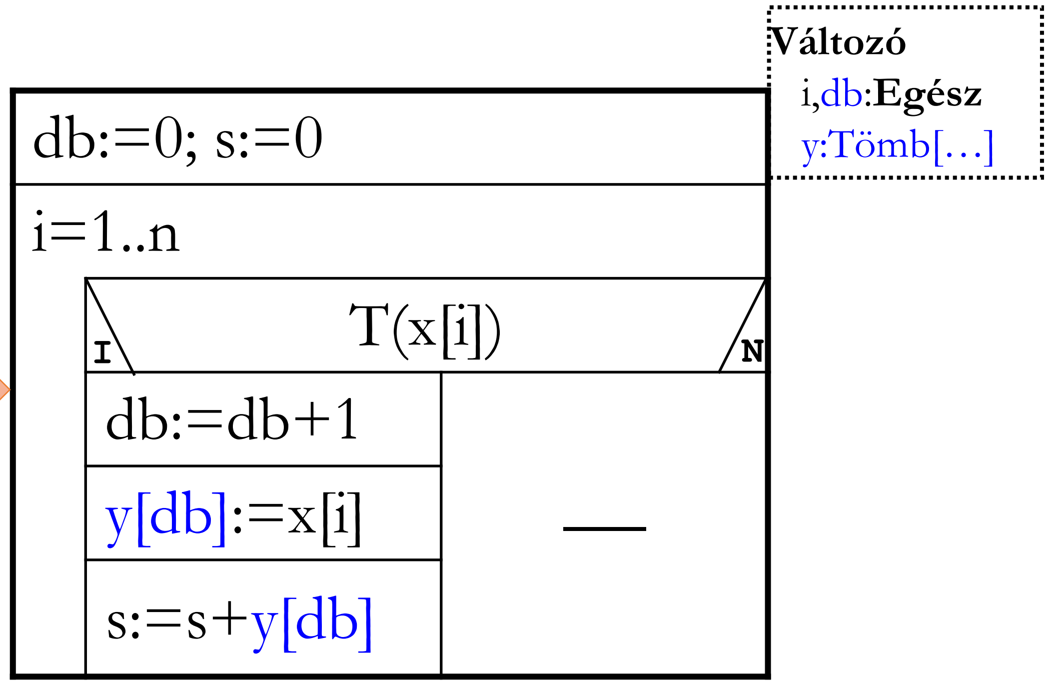
kiválogatás+összegzés

Programtranszformáció: elágazások összevonása

Észrevétel:

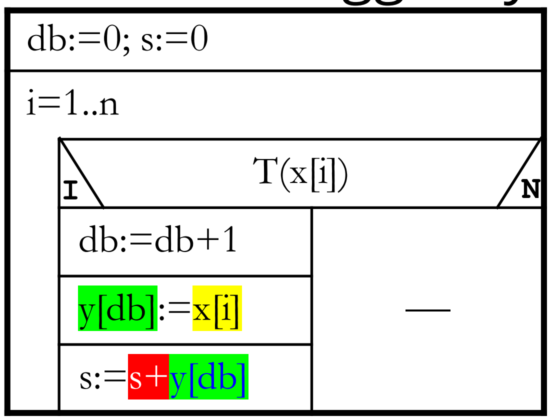
2, azonos feltételű elágazás összevonható (mivel a feltétel paramétere az elágazásban nem változik meg)

Ekvivalens átalakítás



kiválogatás+összegzés

Programtranszformáció: függvénykompozíció



Észrevétel:

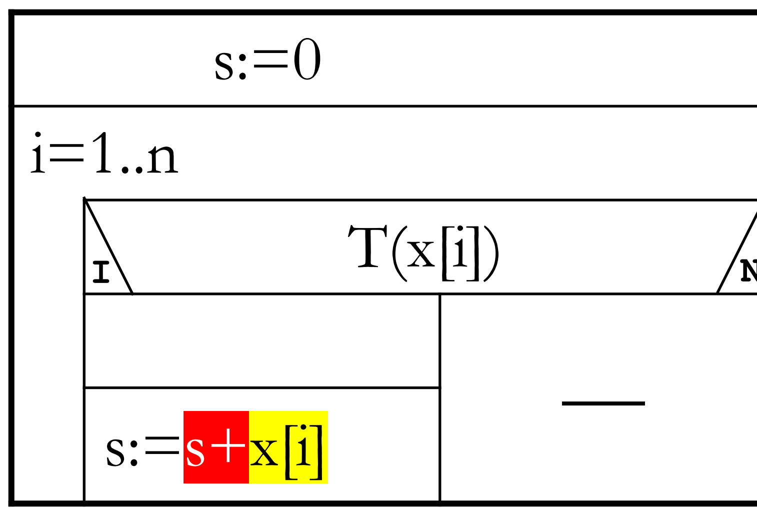
B:=*g*(A); C:=*f*(B)

↔

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

→

y és db elhagyható

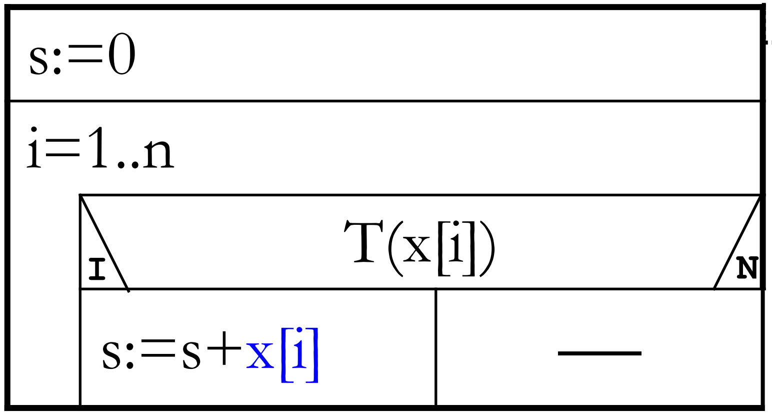


Változó
i :**Egész**

kiválogatás+összegzés

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

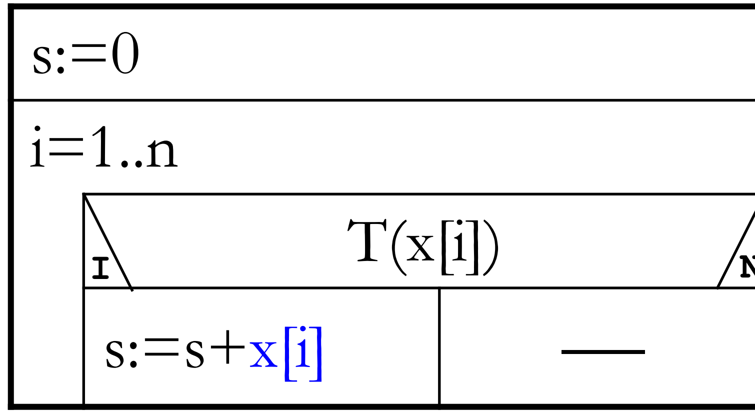
Változó
i:Egész



kiválogatás+összegzés helyességbizonyítással

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

Változó
i:**Egész**

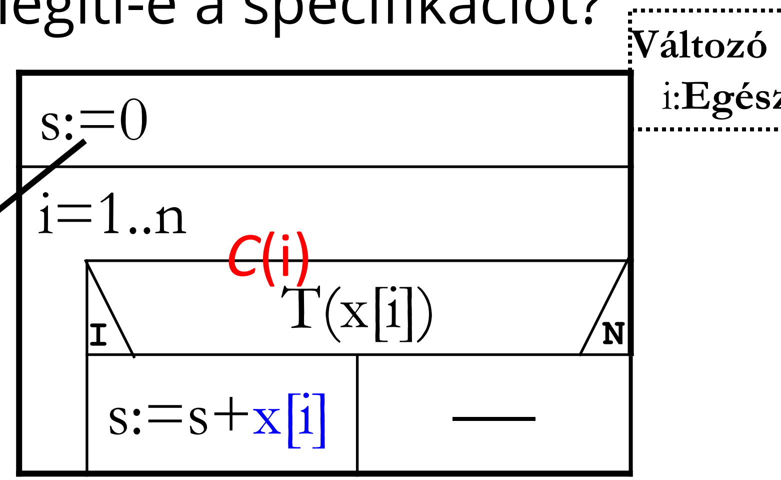


Ez esetben bizonyítanunk kell a helyességet!

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 ciklusmagba belépéskor kiértékelendő:

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

Indukciós bizonyítás:

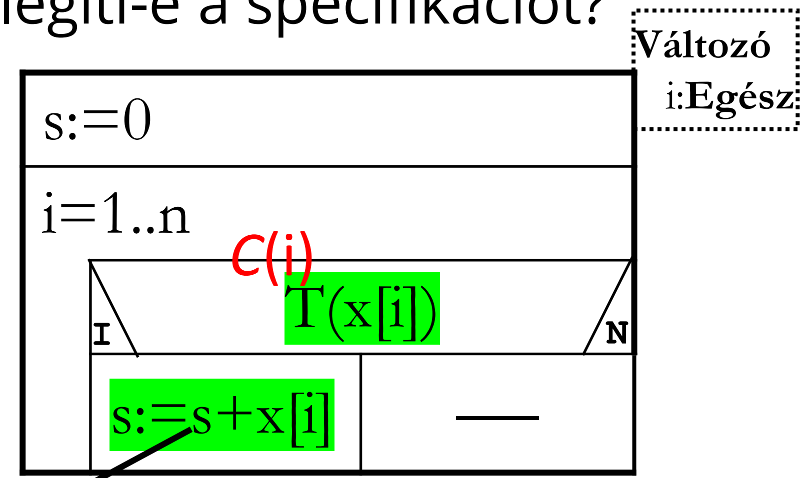
Ciklusba belépéskor (`i`=1):

```
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 ciklus-magba belépéskor kiértékelendő:

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

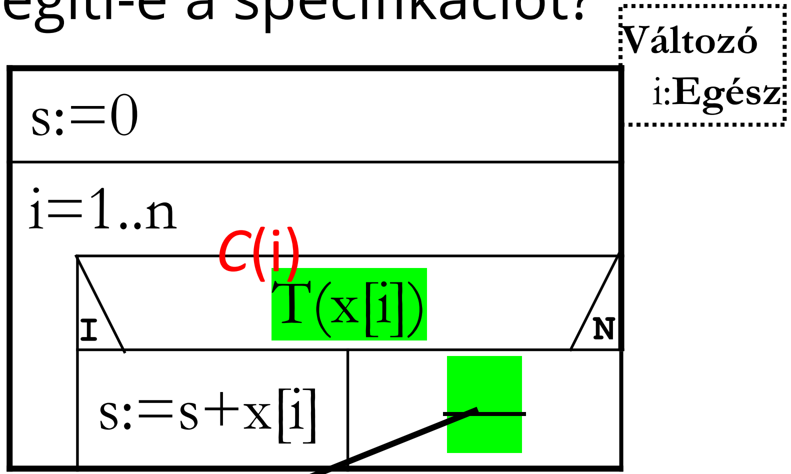
Indukciós lépés:

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 ciklus-magba belépéskor kiértékelendő:

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

Indukciós lépés:

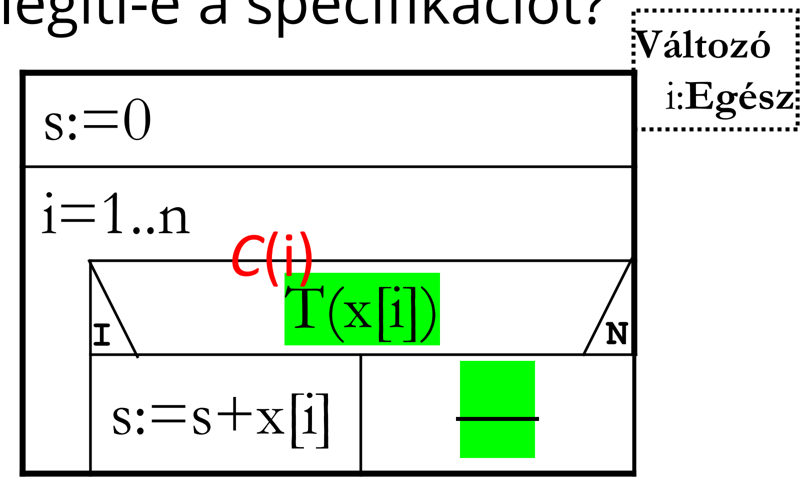
Ciklusmag egyszeri végrehajtása után (`i`→`i+1`):

```
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 ciklus-magba belépéskor kiértékelendő:

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

Ciklusból kilépéskor(i→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.

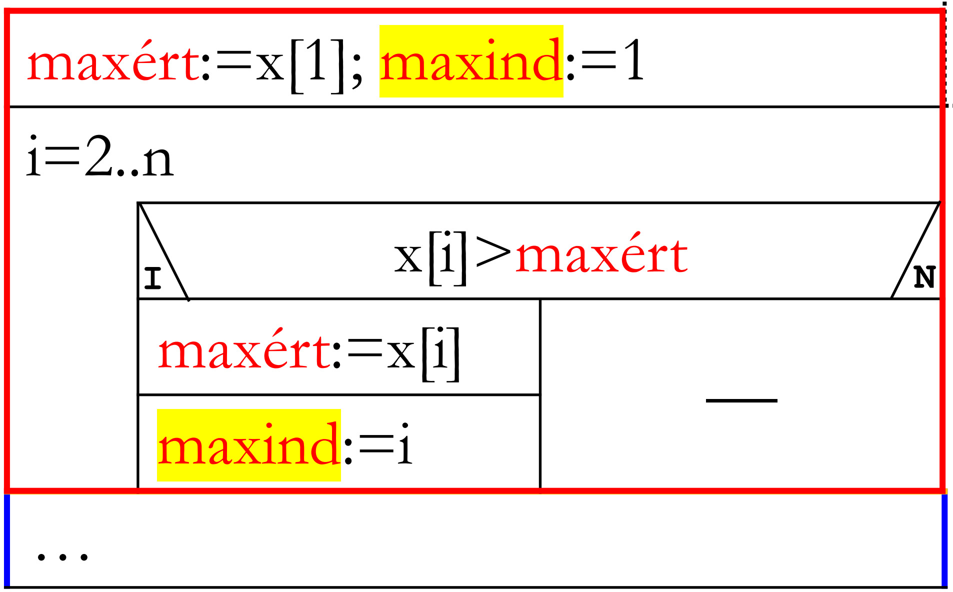
```
Be: n∈N, x∈H[1..n]
Ki: db∈N, maxI∈N[1..n]
Sa: maxért∈H
Fv: legnagyobb:H->L, legnagyobb(h)=h=maxért
Ef: n>0
Uf: (,maxért)=MAX(i=1..n,x[i]) és
    (db,maxI)=KIVÁLOGAT(i=1..n,legnagyobb(x[i]),i)
```

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**

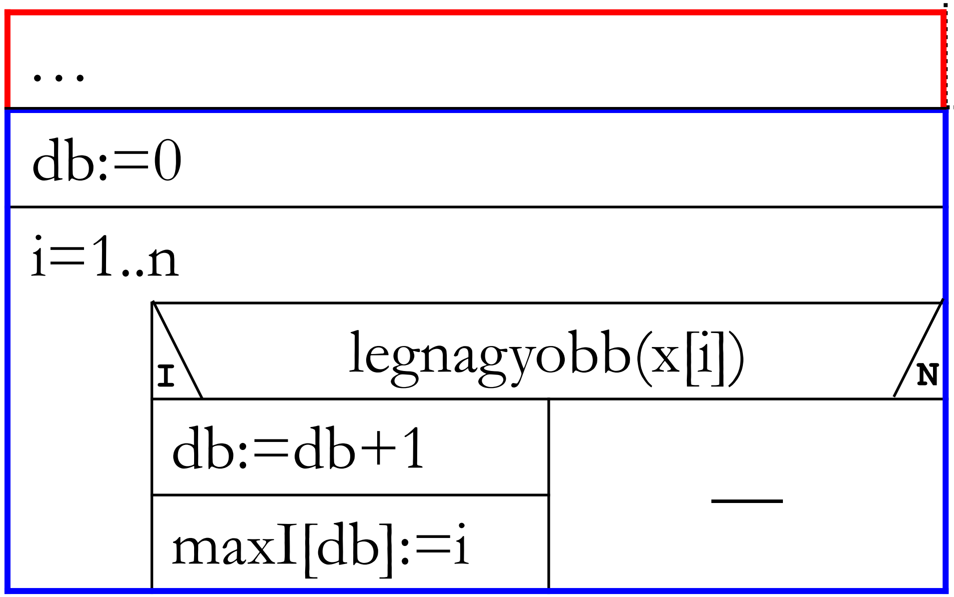


Maximumkiválasztás+kiválogatás

Algoritmus:

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

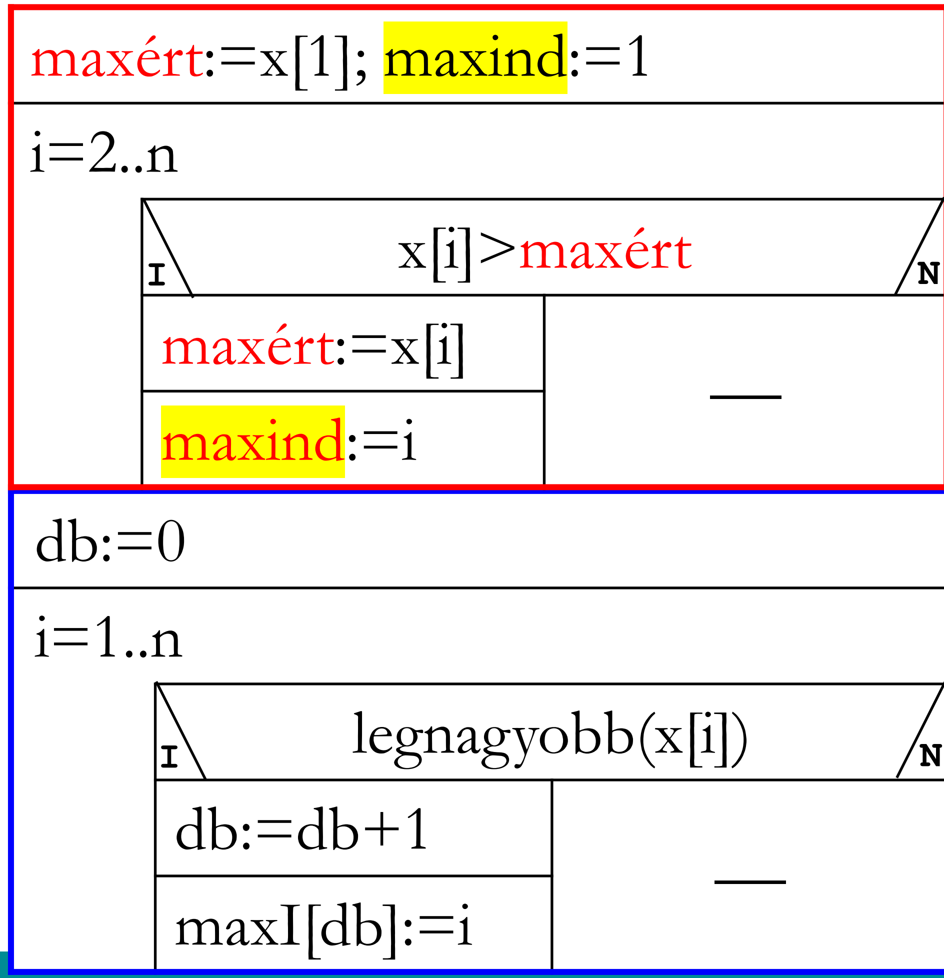
Változó
  i,maxind:**Egész**
  maxért:**TH**



Maximumkiválasztás+kiválogatás

Algoritmus:

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



Észrevétel:
Az eredmény helyes, de bántóan nem hatékony.

Maximumkiválasztás+kiválogatás

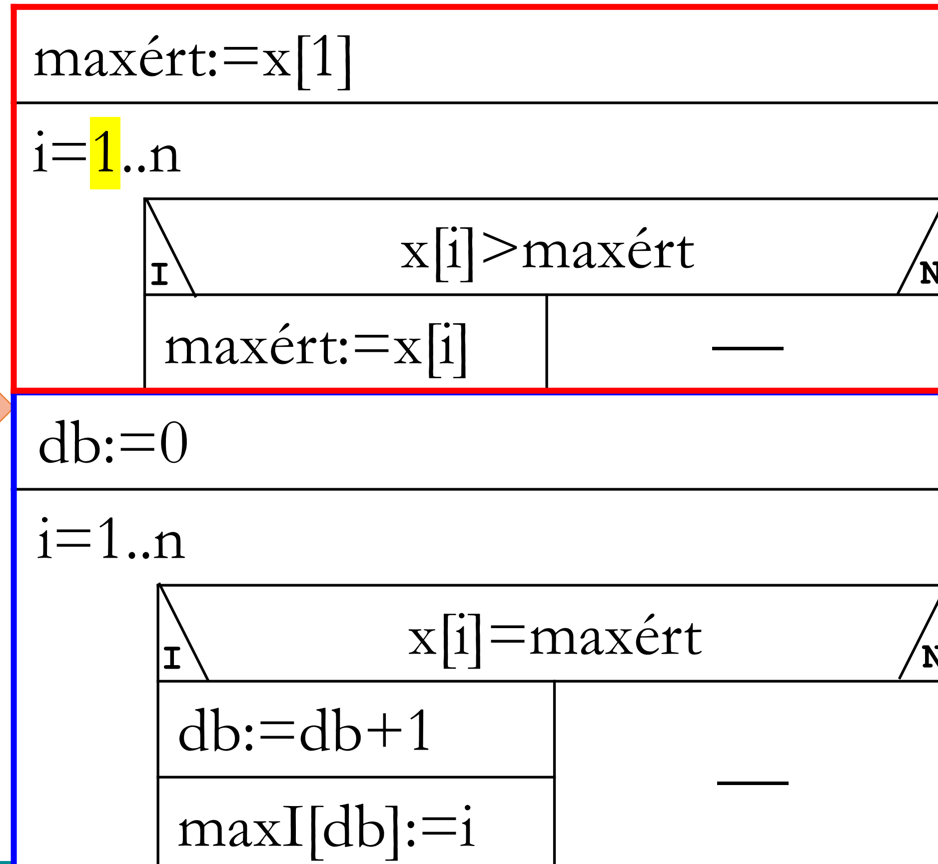
Az algoritmust programtranszformációkkal alakítsuk át!

Változó
i:**Egész**
maxért:**TH**

Észrevétel:

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

Ekvivalens átalakítás



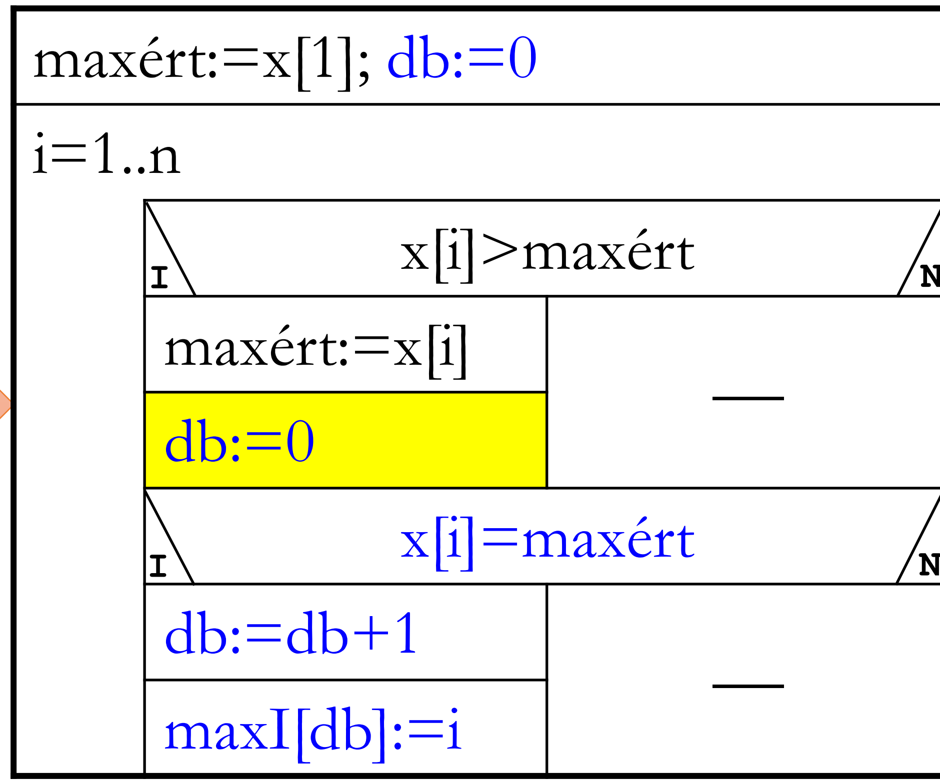
Maximumkiválasztás+kiválogatás

Programtranszformáció: ciklusok és elágazások összevonása

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



Változó
i:**Egész**
maxért:**TH**

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.

Ekvivalens átalakítás

<table>
<tr><td colspan="3">maxért:=x[1]; db:=1; maxI[db]:=1</td></tr>
<tr><td colspan="3">i=2..n</td></tr>
<tr><td rowspan="4"></td><td>x[i]>maxért</td><td>x[i]=maxért</td></tr>
<tr><td>maxért:=x[i]</td><td>db:=db+1</td></tr>
<tr><td>db:=1</td><td rowspan="2">maxI[db]:=i</td></tr>
<tr><td>maxI[db]:=i</td></tr>
</table>

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

megszámolás

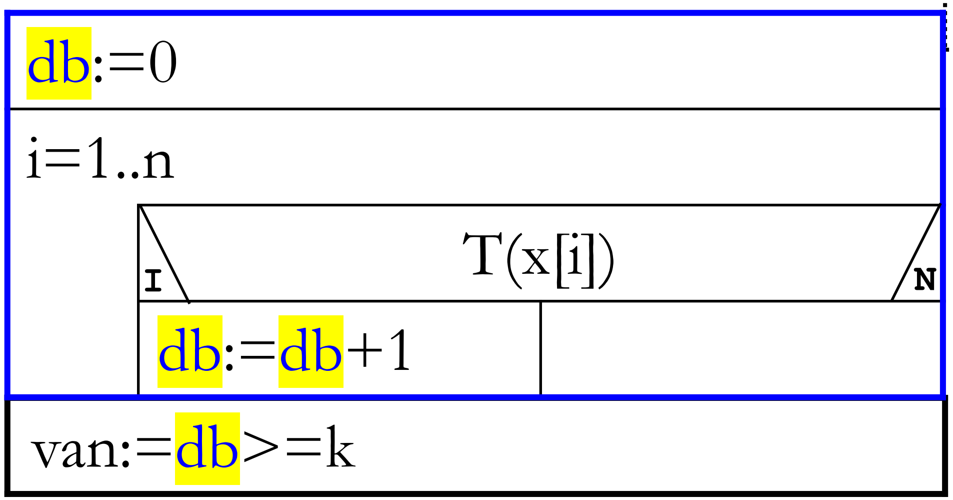
~~eldöntés~~

Eldöntés+megszámolás

Algoritmus:

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

Változó
i,db:Egész



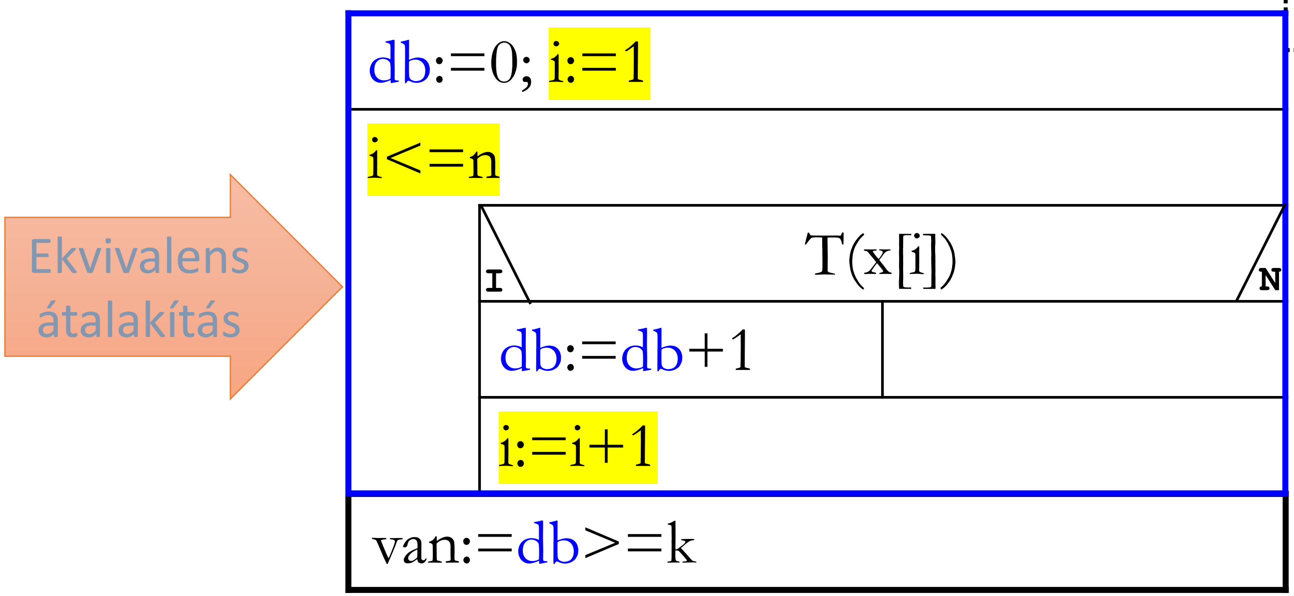
Észrevétel:
Helyes, de nem hatékony meg-oldás!
Ha már találtunk K darab adott tulajdonságút, akkor ne nézzük tovább!

Eldöntés+megszámolás

Az algoritmust programtranszformációkkal alakítsuk át!

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

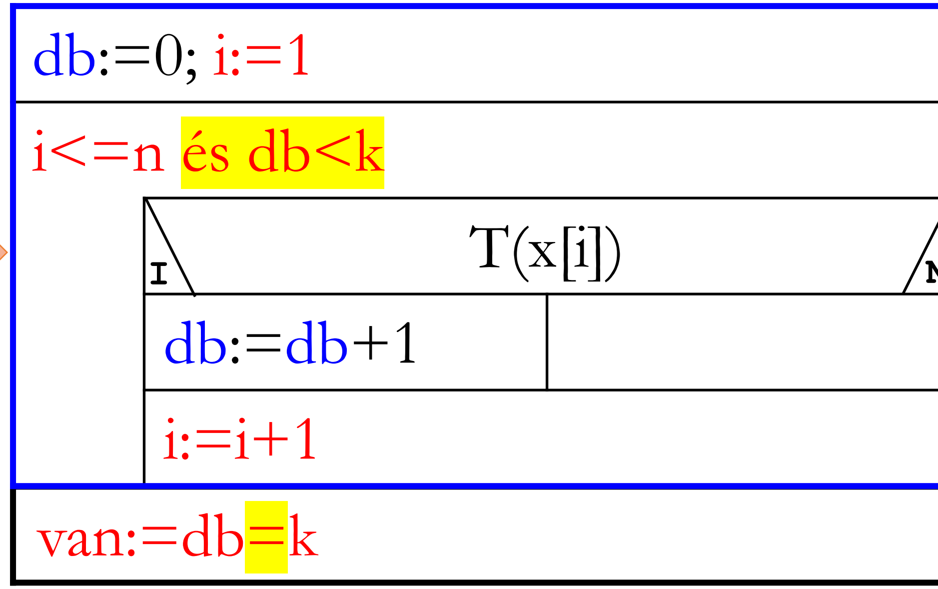
Változó
i,db:**Egész**



Eldöntés+megszámolás

Észrevétel:
Bővítsük a ciklusfeltételt a kívánttal!

Ekvivalens átalakítás



Változó
i,db:**Egész**

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.

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