ELTE | IK

# PROGRAMOZÁS

## Gyakori programozási minta változatok

Horváth Győző

# 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. és, vagy, ->, ∀, ∃
      iv. nevezetes minták
   e) Kiírás
3. Kód

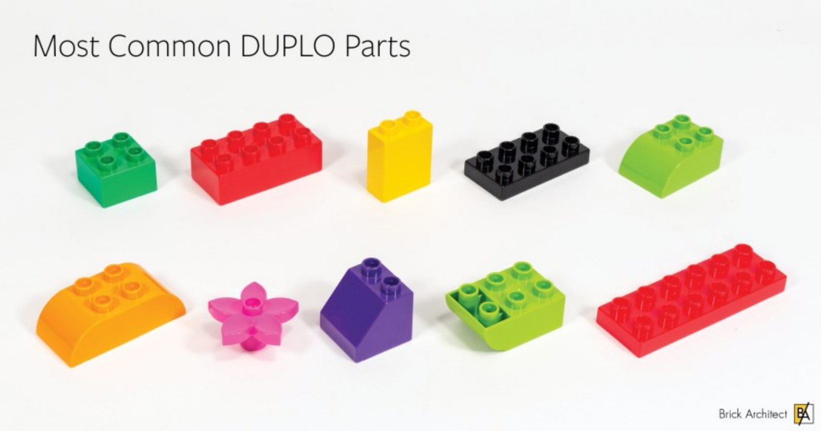
# Megfeleltetések

| 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:S x jegy: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*

8. Másolás
9. Kiválogatás

*szummás, mindenes feladat*
↓
*számlálós ciklus*

# Programozási minták
## intervallumon értelmezett függvényekre

### Összegzés

| i | | f(i) |
|---|---|---|
| e | → | f(e) |
| e+1 | → | f(e+1) |
| e+2 | → | f(e+2) |
| … | → | … |
| u-2 | → | f(u-2) |
| u-1 | → | f(u-1) |
| u | → | f(u) |
| | | = |
| | | s |

### Megszámolás

| i | | T(i) | érték |
|---|---|---|---|
| e | → | IGAZ | 1 |
| e+1 | → | HAMIS | 0 |
| e+2 | → | HAMIS | 0 |
| … | → | … | … |
| u-2 | → | IGAZ | 1 |
| u-1 | → | IGAZ | 1 |
| u | → | HAMIS | 0 |
| | | | = |
| | | | db |

### Maximum kiválasztás

| i | | f(i) |
|---|---|---|
| e | → | f(e) |
| e+1 | → | f(e+1) |
| e+2 | → | f(e+2) |
| … | → | … |
| u-2 | → | f(u-2) |
| u-1 | → | f(u-1) |
| u | → | f(u) |

maxind, maxért

### Feltételes maximumkeresés

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

van, maxind, maxért

# Programozási minták
## intervallumon értelmezett függvényekre

Keresés

| **i** | | **T(i)** |
|---|---|---|
| e | → | HAMIS |
| e+1 | → | HAMIS |
| e+2 | → | IGAZ |
| … | → | … |
| u-2 | → | IGAZ |
| u-1 | → | IGAZ |
| u | → | HAMIS |

van, ind

Eldöntés

| **i** | | **T(i)** |
|---|---|---|
| e | → | HAMIS |
| e+1 | → | HAMIS |
| e+2 | → | IGAZ |
| … | → | … |
| u-2 | → | IGAZ |
| u-1 | → | IGAZ |
| u | → | HAMIS |

van

Kiválasztás

| **i** | | **T(i)** |
|---|---|---|
| e | → | HAMIS |
| e+1 | → | HAMIS |
| e+2 | → | IGAZ |
| … | → | … |
| u-2 | → | IGAZ |
| u-1 | → | IGAZ |
| u | → | HAMIS |

ind

# Programozási minták
## intervallumon értelmezett függvényekre

### Másolás

| i | | | f(i) |
|---|---|---|---|
| e | → | 1 | f(e) |
| e+1 | → | 2 | f(e+1) |
| e+2 | → | 3 | f(e+2) |
| … | → | … | … |
| u-2 | → | u-e-1 | f(u-2) |
| u-1 | → | u-e | f(u-1) |
| u | → | u-e+1 | f(u) |

y

### Kiválogatás

| i | | T(i) | f(i) | | | y |
|---|---|---|---|---|---|---|
| e | → | HAMIS | f(e) | | 1 | f(e+1) |
| e+1 | → | IGAZ | f(e+1) | | 2 | f(e+2) |
| e+2 | → | IGAZ | f(e+2) | db= | 3 | f(u-1) |
| … | → | | … | | | |
| u-2 | → | HAMIS | f(u-2) | | | |
| u-1 | → | IGAZ | f(u-1) | | | |
| u | → | HAMIS | f(u) | | | |

db, y

# Minimumkiválasztás

# Minimumkiválasztás
## példa – specifikáció

minimumkiválasztás

**Feladat:**

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

**Specifikáció:**

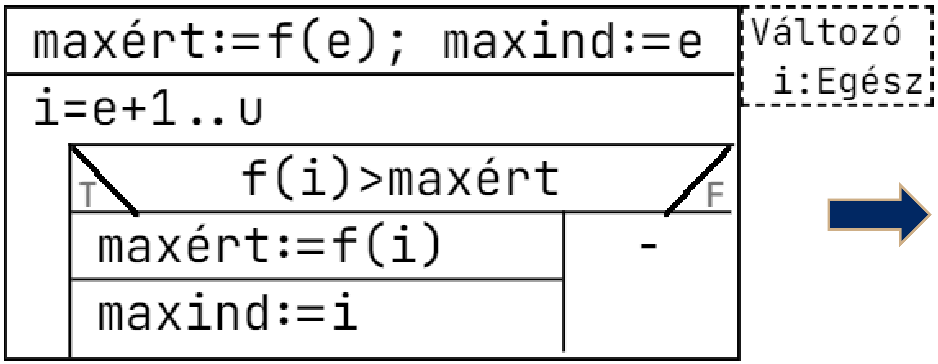
```
Be: e∈Z, u∈Z
Ki: minind∈Z, minért∈H
Ef: e<=u
Uf: minind∈[e..u] és
    ∀i∈[e..u]:(f(minind)<=f(i)) és
    minért=f(minind)
Uf: (minind,minért)=MIN(i=e..u,f(i))
```

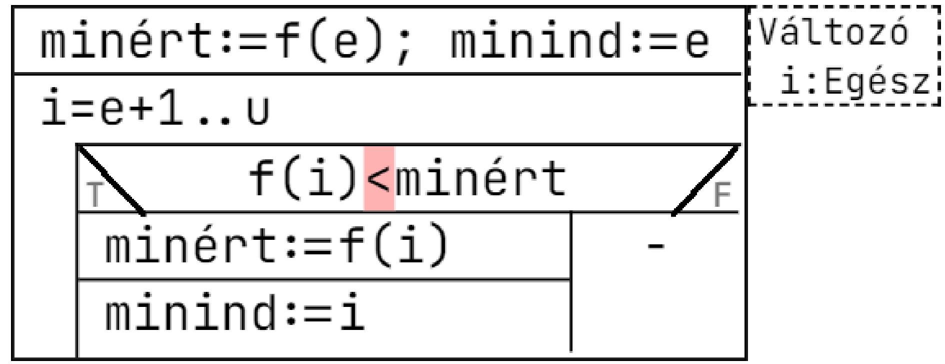
egyetlen különbség a maximumkiválasztáshoz képest az elnevezések mellett

# Minimumkiválasztás
## algoritmus – analóg algoritmikus gondolkodással

**Algoritmus:**

másik reláció használata

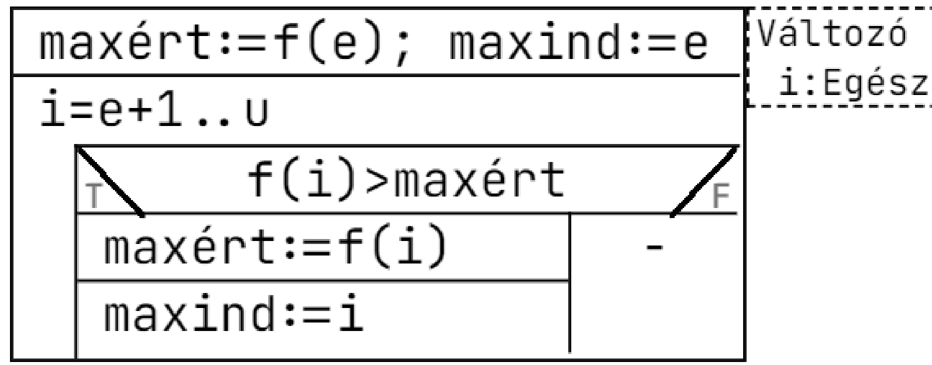


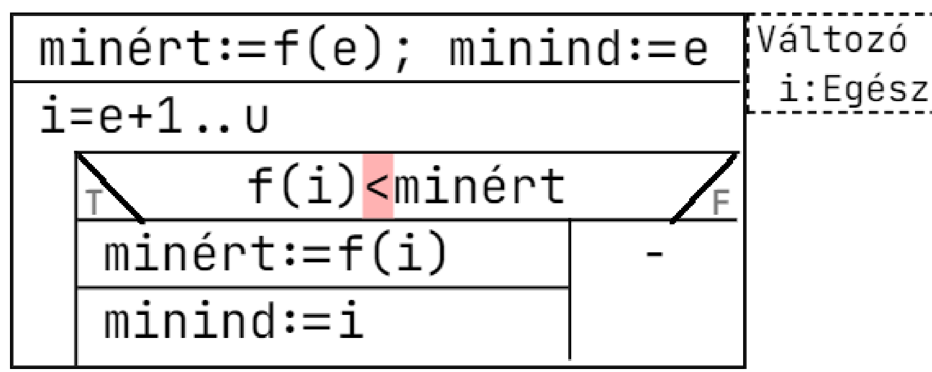



# Minimumkiválasztás
## algoritmus – visszavezetéssel

**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




a≤b felírható függvényszerűen: ≤(a,b)
Az operátor jelentését definiálhatjuk felül

# 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∈H
Ef: e<=u
Uf: (minind,-minért)=
                    MAX(i=e..u,-f(i))
```

ötlet!

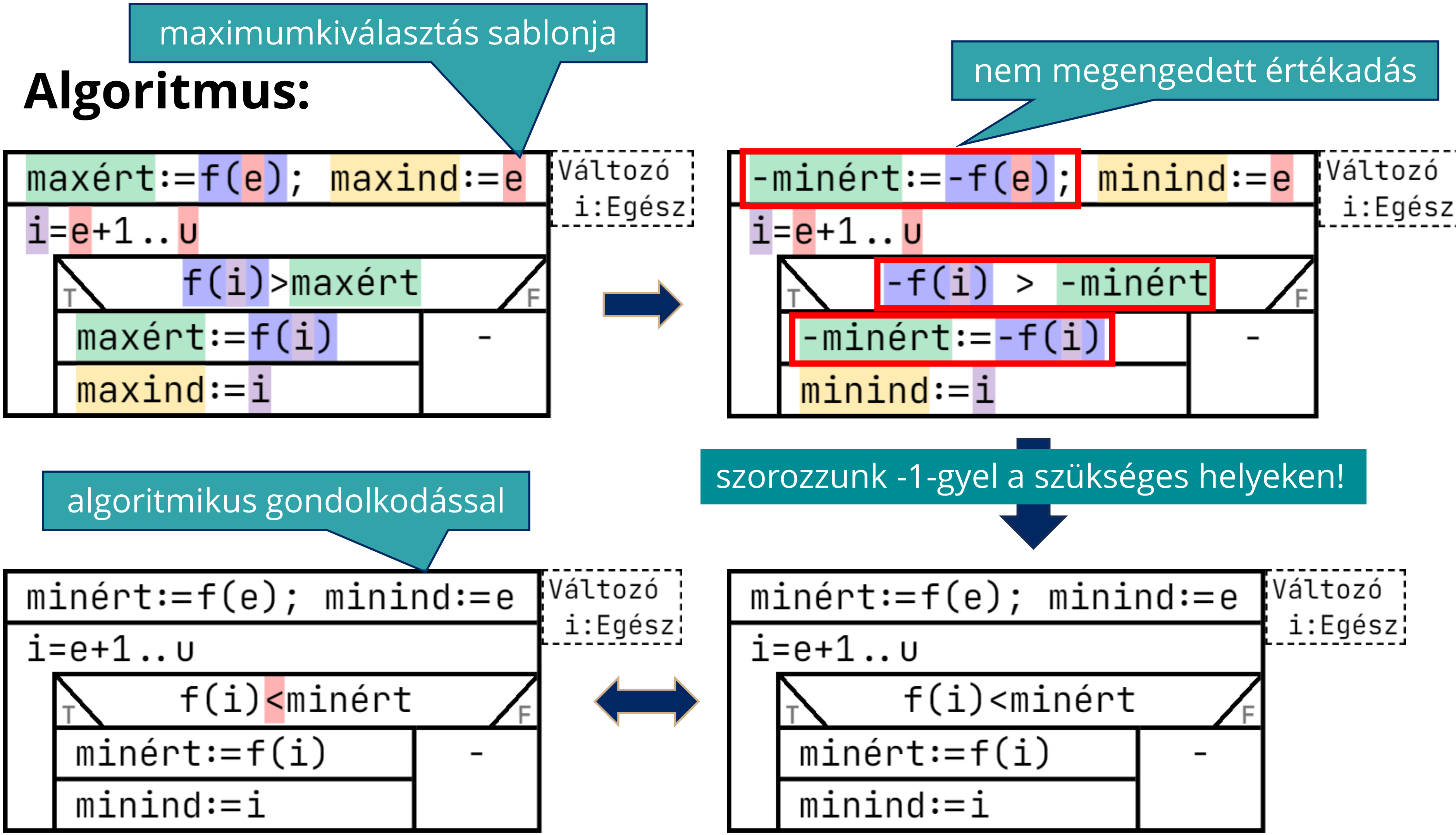
## Visszavezetés:

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

# Minimumkiválasztás algoritmus – visszavezetéssel

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

## Algoritmus:



# 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!

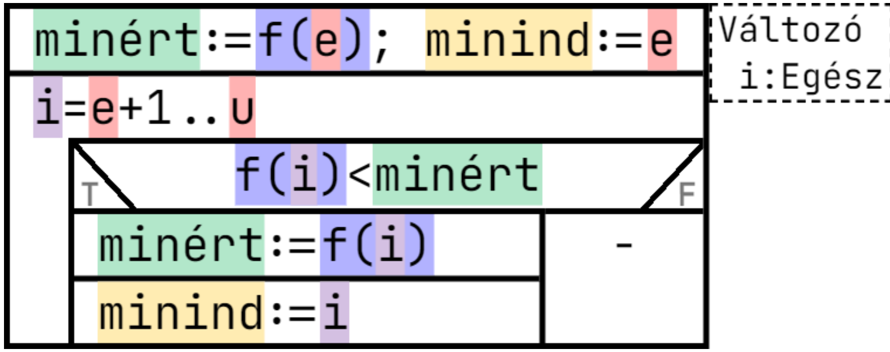
### Specifikáció

```
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

# 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))
```

ötlet!

### Visszavezetés:

| | | |
|---|---|---|
| ind | ~ | -ind |
| e..u | ~ | -u..-e |
| T(i) | ~ | T(-i) |

# 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
T(i) ~ T(-i)

nem megengedett értékadás

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

ind ~ -ind
e..u ~ -u..-e
T(i) ~ T(-i)

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

# Mind eldöntés
## példa

**Feladat:**

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

**Specifikáció:**

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

eldöntésben: ∃i∈[e..u]:(T(i))

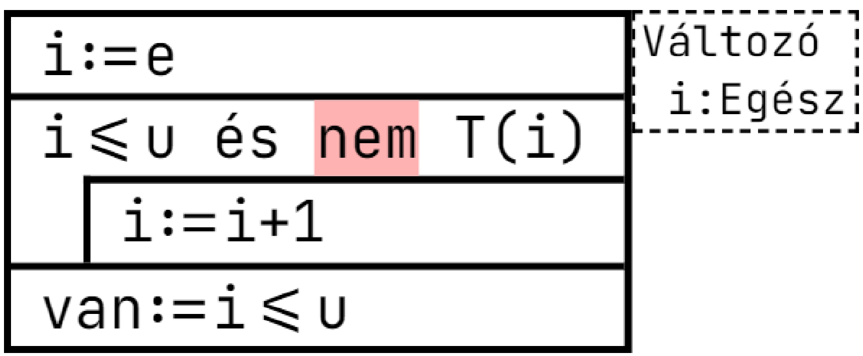
Rövidítve:

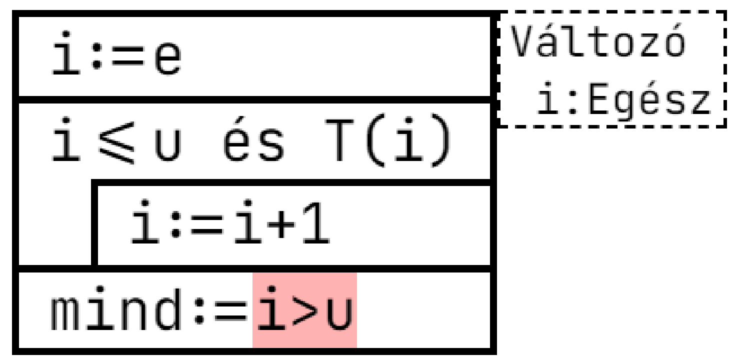
Uf: mind=**MIND**(i=e..u,T(i))

# Mind eldöntés
## algoritmus – algoritmikus gondolkodással

**Algoritmus:**

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



# 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))
```

ötlet!

## 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
```

Változó
i:Egész

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

Változó
i:Egész

nem megengedett értékadás

negálunk a szükséges helyeken

algoritmikus gondolkodással

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

Változó
i:Egész

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

Változó
i:Egész

# 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!

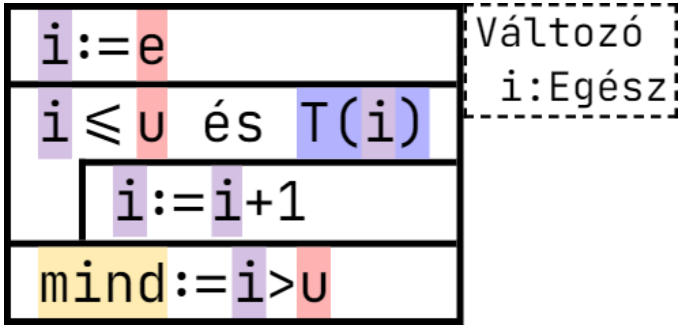
## Specifikáció

```
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



# Kiegészítés:
# 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
  T                    F
  maxért:=f(i)       | -
  maxind:=i          |
```

Változó
i:Egész

```
mh:=hőm[1*7]; lmv:=1
v=1+1..n div 7
  hőm[v*7]>mh
  T                    F
  mh:=hőm[v*7]       | -
  lmv:=v             |
```

Vá
i

| | 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