ELTE | IK

PROGRAMOZÁS

3. előadás

Horváth Győző, Horváth Gyula, Szlávi Péter

Ismétlés

Feladatmegoldás lépései

- Specifikáció
  - mi a feladat?
  - adatok, megszorítások, összefüggések
- Algoritmus
  - hogyan oldjuk meg a feladatot?
  - milyen lépésekre bontjuk?
  - szekvencia (utasítások egymás után)
  - elágazás (utasítások feltételes végrehajtása)
- Kód
  - megvalósítás a gép számára érthető módon
  - adatok deklarálása, beolvasás, feldolgozás, kiírás

Feladatmegoldás lépései

- Adat
  - egy-szerű: elemi
  - több különböző: rekord
  - több egyforma: tömb
- Vezérlési szerkezetek
  - Szekvencia: és
  - Elágazás: ->
  - Ciklus: ∀, ∃, ∑

- Feladatmegoldás
  1. Példa
  2. Specifikáció (← példa)
     1. Adatok (Be, Ki)
     2. Megszorítás (Ef → Be)
     3. Összefüggés (Uf)
  3. Adat → Változó
  4. Algoritmus (← Uf)
  5. Kód (← Spec + Alg)

Tömb

Sok adat

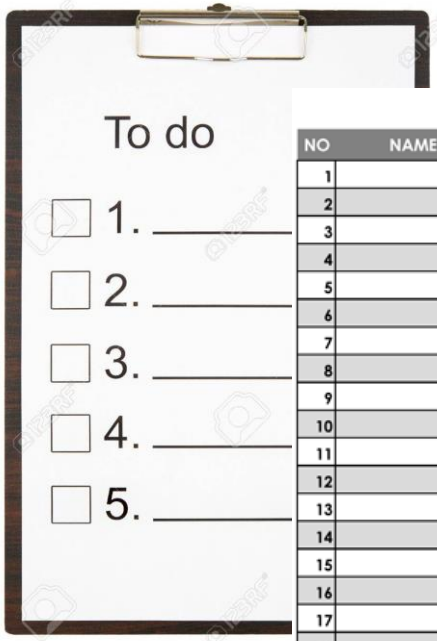
- Vannak olyan feladatok, amelyekben az adatleírás nehezen vagy egyáltalán nem végezhető el elemi típusokkal vagy rekorddal.
- Ezek, amikor **sok egyforma adat**tal dolgozunk
- Példa: melyik a legnagyobb?
  - `Be: a∈Z, b∈Z -> Ki: max∈Z`
  - `Be: a∈Z, b∈Z, c∈Z -> Ki: max∈Z`
  - `Be: a∈Z, b∈Z, c∈Z, d∈Z -> Ki: max∈Z`
- Gond:
  - ez a megoldás nem skálázódik jól a számossággal
  - az utófeltétel egyre bonyolultabb
  - változik az egész megoldás a számossággal (=új spec+alg+kód)

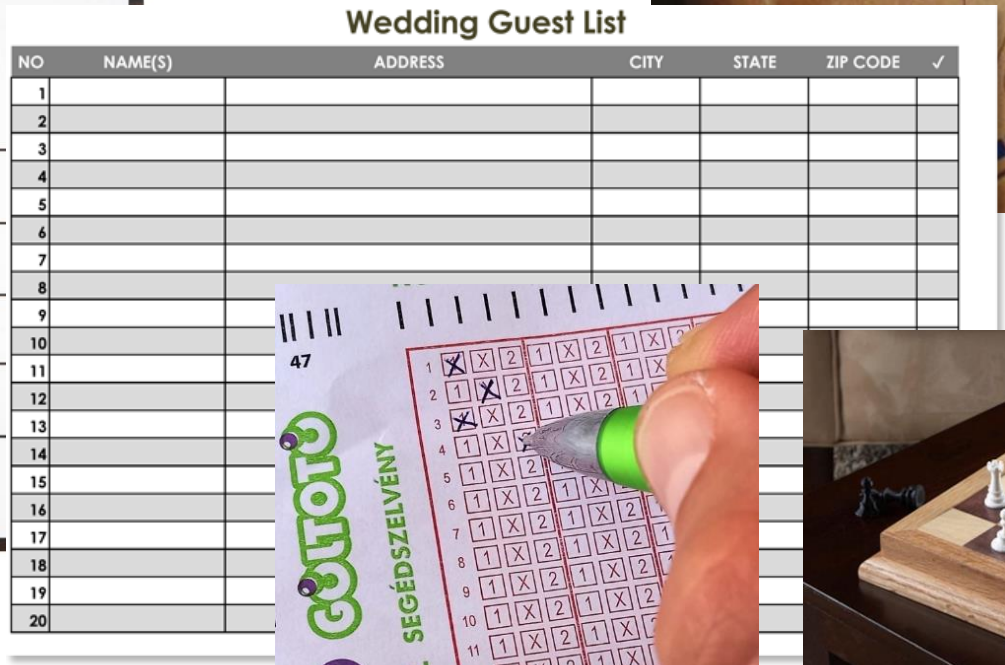
Tömb

- Szükségünk van egy olyan adatszerkezetre, amely
  - sok adat kezelésére alkalmas,
  - jól kezeli a bemenet változó számosságát, és
  - könnyű vele dolgozni (spec, alg, kód)
- **Tömb:**
  - *ugyanolyan funkciójú adatok sokasága*
  - szemantikus egység létrehozása
  - „funkció”: mit *jelent* az adat

Hétköznapi példák

- Amikor sok dolog van…





Tömb vs táblázatkezelő oszlopa

Excel

	A
1	zöld
2	piros
3	sárga
4	fehér
5	fekete
6	
7	

hivatkozás:

A2 → piros

Tömb

	szín
0	zöld
1	piros
2	sárga
3	fehér
4	fekete

hivatkozás:

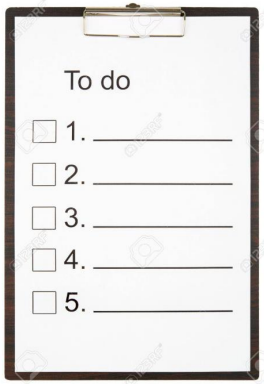
szín[2] → sárga

Számozott felsorolás,
hozzáférés a sorszámon keresztül

Tömb – specifikáció

Azonos funkció → azonos halmazbeli

- **Sorozat:** azonos funkciójú elemek egymásutánja, az elemei sorszámozhatók.
- Példa a definiálásra (`Be, Ki`):
  - `teendők∈S[1..5]`
  - `totó∈K[1..14]`
  - `n∈N, vendégek∈S[1..n]`
- Példa a használatra (=hivatkozás egy elemre, `Ef, Uf`)
  - `teendők[1]`
  - `vendégek[n-2]`



Az összes olyan véges 5 hosszú sorozat halmaza, amely a szöveg alaphalmaz elemeiből áll.

Tömb – algoritmus

- **Tömb:** véges hosszúságú *sorozat algoritmikus párja*, amelynek i-edik tagjával végezhetünk műveleteket.
- Példa a definiálásra:
  - `teendők:Tömb[1..5:Szöveg]`
  - `vendégek:Tömb[1..n:Szöveg]`
  - `Konst MAXN=100, Változó n:Egész`
    `vendégek:Tömb[1..MAXN:Szöveg]`

Specifikáció:

```
teendők∈S[1..5]
totó∈K[1..14]
n∈N, vendégek∈S[1..n]
```

- Példa a használatra:
  - hivatkozás: `teendők[2]`
  - értékadás: `vendégek[1]:="Miklós atya"`

Tömb – C# kód

Algoritmus:
- teendők:Tömb[1..5:Szöveg]

- Statikus tömb – ismert méret
  - `string[] teendok=new string[5];`
  - `char[] toto=new char[14];`

Algoritmus:
- Konst MAXN=100, Változó n:Egész vendégek:Tömb[1..MAXN:Szöveg]

- Statikus tömb – max.méret
  - ```
    int n;
    const int MAXN = 100;
    string[] vendegek = new string[MAXN];
    ```

	vendégek
1	Miklós atya
2	Józsi bácsi
3	Ili néni
n= 4	Gábor barátom
5	
6	
7	
8	
9	
MAXN= 10	

- Statikus tömb – igény szerinti méret
  - ```
    int n;
    int.TryParse(Console.ReadLine(), out n);
    string[] vendegek = new string[n];
    ```

	vendégek
1	Miklós atya
2	Józsi bácsi
3	Ili néni
n= 4	Gábor barátom

Algoritmus:
- vendégek:Tömb[1..n:Szöveg]

Tömb – algoritmus → kód

C#-ban a tömbök 0-tól indexelődnek.

1. ötlet: ne használjuk a 0. elemet!

Algoritmus		Kód	
		0	?
1	a	1	a
2	b	2	b
3	c	3	c

Deklarációs példa:

```
x:Tömb[1..n:Valós]
```

C# kód:

```
float[] x=new float[n+1];
```

Algoritmus:

```
i=1..n
  x[i]:=i
```

C# kód:

```
for(int i=1;i<=n;++i){
  x[i]=i;
}
```

Tömb – algoritmus → kód

C#-ban a tömbök 0-tól indexelődnek.

2. ötlet: indexeltolás!

Algoritmus		Kód	
1	a	0	a
2	b	1	b
3	c	2	c

Deklarációs példa:

```
x:Tömb[1..n:Valós]
```

C# kód:

```
float[] x=new float[n];
```

Algoritmus:

```
i=1..n
  x[i]:=i
```

C# kód:

```
for(int i=1;i<=n;++i){
  x[i-1]=i;
}
//------- vagy -------
for(int i=1-1;i<=n-1;++i){
  x[i]=i+1;
}
```

Tömb – algoritmus → kód

C#-ban a tömbök 0-tól indexelődnek.

3. ötlet: általános esetben!

Algoritmus		Kód	
e	a	0	a
e+1	b	1	b
u	c	2	c

Deklarációs példa:

```
x:Tömb[e..u:Valós]
```

C# kód:

```
float[] x=new float[u-e+1];
```

Algoritmus:

```
i=e..u
  x[i]:=i
```

C# kód:

```
for(int i=e;i<=u;++i){
  x[i-e]=i;
}
//------- vagy -------
for(int i=e-e;i<=u-e;++i){
  x[i]=i+e;
}
```

Tömb – algoritmus → kód

C#-ban a tömbök 0-tól indexelődnek.

3. ötlet: példa

Algoritmus		Kód	
-1	4	0	4
0	5	1	5
1	6	2	6

Deklarációs példa:

```
x:Tömb[-1..10:Valós]
```

C# kód:

```
float[] x=new float[12];
```

Algoritmus:

```
i=-1..10
  x[i]:=i+5
```

C# kód:

```
for(int i=-1;i<=10;++i){
  x[i+1]=i+5;
}
//------- vagy -------
for(int i=0;i<=11;++i){
  x[i]=i+(-1)+5;
}
```

Konstans tömb

- **Specifikáció:**
  - SZÍNEK∈S[0..4]=
    ("zöld","piros","sárga","fehér","fekete")
- **Algoritmus:**
  - Konstans SZÍNEK:Tömb[0..4:Szöveg]=
    ("zöld","piros","sárga","fehér","fekete")
- **Kód:**

```
string[] SZINEK = new string[5]
  { "zöld", "piros", "sárga", "fehér", "fekete" };
// vagy
string[] SZINEK =
  { "zöld", "piros", "sárga", "fehér", "fekete" };
```

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

Analóg programozás

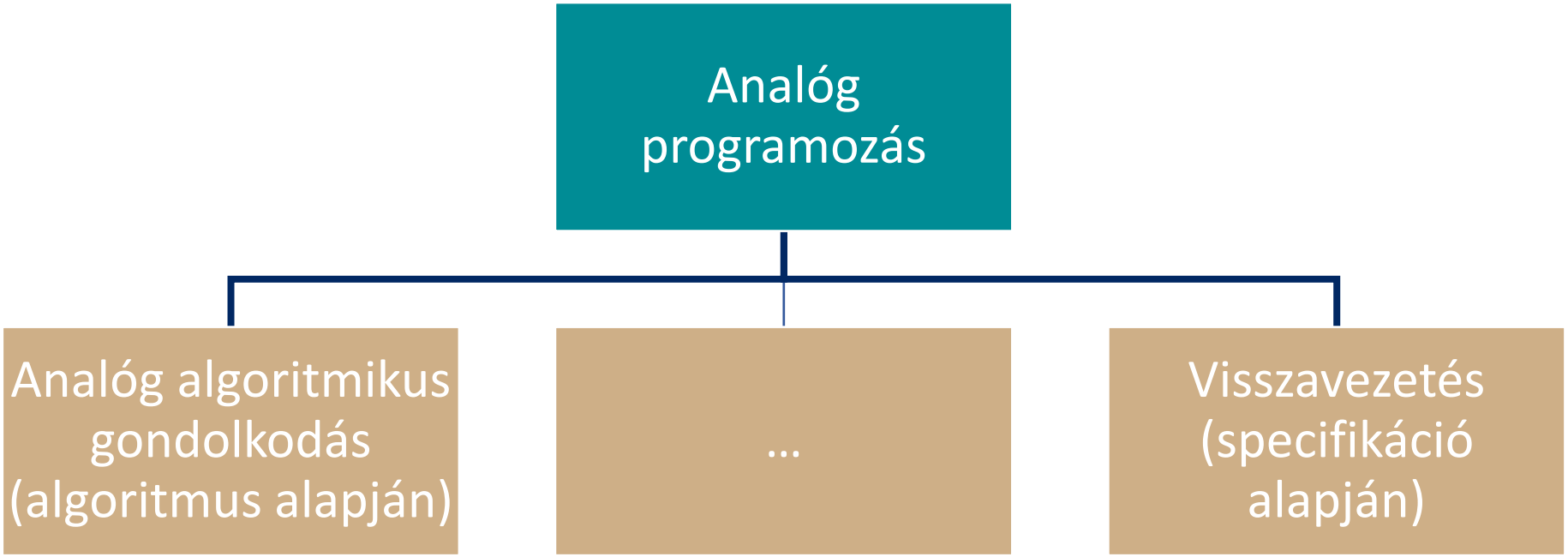
Analóg problémamegoldás

- Amikor egy olyan feladatot kell megoldani, amelyhez **hasonló feladat**ot már korábban megoldottunk, akkor a megoldó programot a **korábbi feladat megoldó programja alapján** állítjuk elő.
- Általános problémamegoldó módszer
  - gyakorlati tapasztalat, megtanult ismeret
  - élet minden területén
  - gyakorló programozó eszköztárában is

Analóg programozás

- Cél: egy feladat megoldására program készítése
- Minden egyes alkalommal újra és újra kitalálhatjuk a megoldást
- Vagy egy korábbi, hasonló feladat megoldását vehetjük alapul
- **Analóg programozás**: a konkrét feladatot egy korábbi feladat megoldása alapján állítjuk elő.

Analóg programozás



Analóg programozás – analóg algoritmikus gondolkodás

Mintafeladat
(feladatosztály sablonja)

Konkrét feladat

Specifikáció ⟷ Specifikáció

Algoritmus → Algoritmus

↓

Kód

A mintafeladathoz hasonló feladatokat a mintafeladatnál alkalmazott gondolatsorral, annak ötleteit kölcsönözve, lemásolva, analóg algoritmikus gondolkodással oldjuk meg.

Analóg programozás – visszavezetés

Mintafeladat
(feladatosztály sablonja)

Konkrét feladat

Specifikáció $\xleftrightarrow{\Delta}$ Specifikáció

Algoritmus $\xrightarrow{\Delta}$ Algoritmus

↓

Kód

A **visszavezetés** során a *mintaprogramot sablonként használva* állítjuk elő a kitűzött feladat megoldó programját úgy, hogy figyelembe vesszük a kitűzött- és a mintafeladat specifikációkban felfedett *eltéréseket*.

Visszavezetés

- A visszavezetés során a mintafeladat megoldó programját (a mintaprogramot) **sablon**ként használva állítjuk elő a kitűzött feladat megoldó programját úgy, hogy figyelembe vesszük a kitűzött- és a mintafeladat specifikációkban felfedett **eltérések**et.
- Megtehetjük, mivel ismerjük a mintafeladat és a kitűzött feladat precíz leírását.

Példa – „mintafeladat”

Feladat:

Egy középiskolai osztályban papírgyűjtést szerveznek. Minden diákról tudjuk, hány kg papírt hozott be. Határozd meg, hogy mennyi gyűlt összesen össze?

Példa:
[1, 3, 2, 0, 5, 2] → 13

Példa:
n=6, kg=[1, 3, 2, 0, 5, 2] → össz=13

	kg
1	1
2	3
3	2
4	0
5	5
n=6	2

össz= 13

Példa – „mintafeladat”

Példa:
n=6, kg=[1, 3, 2, 0, 5, 2] → össz=13

Feladat:

Határozd meg egy n elemű tömb elemeinek összegét!

Specifikáció:

```
Be: n∈N, kg∈N[1..n]
Ki: össz∈N
Ef: -
Uf: össz = SZUMMA(i=1..n, kg[i])
```

$$\text{össz} = \sum_{i=1}^{n} kg[i]$$

	kg
1	1
2	3
3	2
4	0
5	5
n=6	2

össz= 13

Algoritmus:

```
össz:=0
i=1..n
    össz:=össz+kg[i]
```

Példa – „mintafeladat”

Példa:
n=6, kg=[1, 3, 2, 0, 5, 2] → össz=13

Feladat:

Határozd meg egy n elemű tömb elemeinek összegét!

Algoritmus:

```
össz:=0
i=1..n
    össz:=össz+kg[i]
```

	kg
1	1
2	3
3	2
4	0
5	5
n=6	2

össz= 13

Leírás:

A megoldás során a kívánt összeget fokozatosan állítjuk elő. Ehhez végig kell vezetnünk egy i egész típusú változót a tömb indexein, és minden lépésben hozzá kell adni a kezdetben nullára beállított össz változóhoz a kg[i] értéket. A megoldó program tehát a kezdeti értékadás után (ahol az össz változó értékét nullára állítjuk) egy ciklus, amely az i változót egyesével növeli egészen n-ig.

Példa – skaláris szorzat

Feladat:

Határozzuk meg két azonos dimenziójú vektor skaláris szorzatát!

Példa:
n=3,
a=[1, -3, 2], → s = -9
b=[4, 5, 1]

	a	b	a[i]*b[i]
1	1	4	4
2	-3	5	-15
n=3	2	1	2
		s=	-9

Példa – skaláris szorzat

Példa:
n=3,
a=[1, -3, 2], → s = -9
b=[4, 5, 1]

Feladat:

Határozd meg egy n elemű tömb elemeinek összegét!

Specifikáció:

```
Be: n∈N, a∈Z[1..n], b∈Z[1..n]
Ki: s∈Z
Ef: -
Uf: s = SZUMMA(i=1..n, a[i]*b[i])
```

	a	b	a[i]*b[i]
1	1	4	4
2	-3	5	-15
n=3	2	1	2
		s=	-9

$$s = \sum_{i=1}^{n} a[i] * b[i$$

Példa – skaláris szorzat

Papírgyűjtés

```
Be: n∈N, kg∈N[1..n]
Ki: össz∈N
Ef: -
Uf: össz=SZUMMA(i=1..n,kg[i])
```

Skaláris szorzat

```
Be: n∈N, a∈Z[1..n], b∈Z[1..n]
Ki: s∈Z
Ef: -
Uf: s=SZUMMA(i=1..n,a[i]*b[i])
```

A specifikációk összevetéséből is látszik, hogy a két feladat hasonlít egymásra.

Állítsuk elő az algoritmust

- **analóg algoritmikus gondolkodással**, majd
- **visszavezetéssel**!

Példa – skaláris szorzat
analóg algoritmikus gondolkodás

	a	b	a[i]*b[i]
1	1	4	4
2	-3	5	-15
n=3	2	1	2
		s=	-9

Algoritmus:

```
össz:=0
i=1..n
    össz:=össz+kg[i]
```

```
s:=0
i=1..n
    s:=s+a[i]*b[i]
```

Leírás:

A megoldás során a kívánt összeget fokozatosan állítjuk elő. Ehhez végig kell vezetnünk egy i egész típusú változót a tömb indexein, és minden lépésben hozzá kell adni a kezdetben nullára beállított össz változóhoz a kg[i] értéket. A megoldó program tehát a kezdeti értékadás után (ahol az össz változó értékét nullára állítjuk) egy ciklus, amely az i változót egyesével növeli egészen n-ig.

A megoldás során a kívánt összeget fokozatosan állítjuk elő. Ehhez végig kell vezetnünk egy i egész típusú változót a **tömbök** indexein, és minden lépésben hozzá kell adni a kezdetben nullára beállított **s** változóhoz az **a[i]*b[i]** értéket. A megoldó program tehát a kezdeti értékadás után (ahol az **s** változó értékét nullára állítjuk) egy ciklus, amely az i változót egyesével növeli egészen n-ig.

Példa – skaláris szorzat visszavezetéssel

Vegyük számba a két feladat közötti különbségeket, a mintaprogram algoritmusában pedig cseréljük le a megfelelő részeket!

Papírgyűjtés (mintafeladat)

```
Be: n∈N, kg∈N[1..n]
Ki: össz∈N
Ef: -
Uf: össz=SZUMMA(i=1..n,kg[i])
```

Skaláris szorzat (konkrét feladat)

```
Be: n∈N, a∈Z[1..n], b∈Z[1..n]
Ki: s∈Z
Ef: -
Uf: s=SZUMMA(i=1..n,a[i]*b[i])
```

```
össz   ~ s
kg[i]  ~ a[i]*b[i]
```

Algoritmus:

```
össz:=0
i=1..n
    össz:=össz+kg[i]
```

```
s:=0
i=1..n
    s:=s+a[i]*b[i]
```

Példa – számok összege

Feladat:

Határozzuk meg az a és b egész számok (a≤b) közé eső egész számok összegét (a határokat is beleértve)!

Példa:
a=1, b=5 → össz=15
a=-2, b=2 → össz=0

a=	1
	2
	3
	4
b=	5
össz=	15

a=	-2
	-1
	0
	1
b=	2
össz=	0

Példa – számok összege

Példa:
a=1, b=5 → össz=15
a=-2, b=2 → össz=0

Feladat:

Határozzuk meg az a és b egész számok (a≤b) közé eső egész számok összegét (a határokat is beleértve)!

Specifikáció:

```
Be: a∈Z, b∈Z
Ki: össz∈Z
Ef: a<=b
Uf: össz=SZUMMA(i=a..b, i)
```

$$\text{össz} = \sum_{i=1}^{n} i$$

a=	-2
	-1
	0
	1
b=	2
össz=	0

Példa – számok összege

Papírgyűjtés
(mintafeladat)

```
Be: n∈N, kg∈N[1..n]
Ki: össz∈N
Ef: -
Uf: össz=SZUMMA(i=1..n,kg[i])
```

Számok összege
(konkrét feladat)

```
Be: a∈Z, b∈Z
Ki: össz∈Z
Ef: a<=b
Uf: össz=SZUMMA(i=a..b, i)
```

A specifikációk összevetéséből is látszik, hogy a két feladat hasonlít egymásra.

Állítsuk elő az algoritmust **visszavezetéssel**!

Példa – számok összege visszavezetéssel

Vegyük számba a két feladat közötti különbségeket, a mintaprogram algoritmusában pedig cseréljük le a megfelelő részeket!

Papírgyűjtés (mintafeladat)

```
Be: n∈N, kg∈N[1..n]
Ki: össz∈N
Ef: -
Uf: össz=SZUMMA(i=1..n,kg[i])
```

Számok összege (konkrét feladat)

```
Be: a∈Z, b∈Z
Ki: össz∈Z
Ef: a<=b
Uf: össz=SZUMMA(i=a..b, i)
```

```
1..n   ~ a..b
kg[i]  ~ i
```

Algoritmus:

```
össz:=0
i=1..n
    össz:=össz+kg[i]
```

```
össz:=0
i=a..b
    össz:=össz+i
```

Tanulságok

- Eltérések ellenére ezek a feladatok annyira hasonlóak, hogy ugyanazon sablon alapján megoldhatók.
- Mi alapján sorolhatók ugyanazon sablon alá?
- Hogyan lehetne általánosan megfogalmazni az ilyen feladatokat?

Közös tulajdonságok
egész számok zárt intervalluma

Papírgyűjtés

```
össz=SZUMMA(i=1..n, kg[i])
```

	kg		i		kg[i]
1	1	e=	1	→	1
2	3		2	→	3
3	2		3	→	2
4	0		4	→	0
5	5		5	→	5
6	2	u=	6	→	2

Tömb indextartománya

Skalárszorzat

```
s=SZUMMA(i=1..n, a[i]*b[i])
```

	a	b		i		a[i]*b[i]
1	1	4	e=	1	→	4
2	-3	5		2	→	-15
3	2	1	u=	3	→	2

Ciklusváltozó futási tartományát határozza meg

Tömb indextartománya

Számok összege

```
össz=SZUMMA(i=a..b, i)
```

	i		i
e=	-2	→	-2
	-1	→	-1
	0	→	0
	1	→	1
u=	2	→	2

Zárt intervallum

Közös tulajdonságok
intervallum elemeihez rendelt érték

Papírgyűjtés

```
össz=SZUMMA(i=1..n, kg[i])
```

	kg		i		kg[i]
1	1	e=	1	→	1
2	3		2	→	3
3	2		3	→	2
4	0		4	→	0
5	5		5	→	5
6	2	u=	6	→	2

Tömb eleme

Skalárszorzat

```
s=SZUMMA(i=1..n, a[i]*b[i])
```

	a	b		i		a[i]*b[i]
1	1	4	e=	1	→	4
2	-3	5		2	→	-15
3	2	1	u=	3	→	2

Tömbök elemértékének szorzata

Leképezés (függvény), amely az intervallum elemeihez rendel valamilyen értéket: i → f(i)

Számok összege

```
össz=SZUMMA(i=a..b, i)
```

	i		i
e=	-2	→	-2
	-1	→	-1
	0	→	0
	1	→	1
u=	2	→	2

Az intervallum i eleme

Közös tulajdonságok
összegzés

Papírgyűjtés

```
össz=SZUMMA(i=1..n, kg[i])
```

	kg		i		kg[i]
1	1	e=	1	→	1
2	3		2	→	3
3	2		3	→	2
4	0		4	→	0
5	5		5	→	5
6	2	u=	6	→	2

Skalárszorzat

```
s=SZUMMA(i=1..n, a[i]*b[i])
```

	a	b		i		a[i]*b[i]
1	1	4	e=	1	→	4
2	-3	5		2	→	-15
3	2	1	u=	3	→	2

A leképezett értékeket 0-tól kezdve össze kellett adni: s=0+f(e)+f(e+1)+…+f(u)

Számok összege

```
össz=SZUMMA(i=a..b, i)
```

	i		i
e=	-2	→	-2
	-1	→	-1
	0	→	0
	1	→	1
u=	2	→	2

össz s össz

Általános feladat

Feladat:

Legyen adott az egész számok egy [e..u] intervallumán értelmezett f:[e..u]→H függvény (H olyan halmaz, amelyen értelmezett az összeadás művelete). Határozzuk meg az f függvény [e..u] intervallumon felvett értékeinek az összegét, azaz az f(e)+f(e+1)+...+f(u) kifejezés értékét!

i		f(i)
e	→	f(e)
		+
e+1	→	f(e+1)
		+
...	→	...
		+
u-1	→	f(u-1)
		+
u	→	f(u)

Általános feladat

Feladat:

Legyen adott az egész számok egy [e..u] intervallumán értelmezett f:[e..u]→H függvény (H olyan halmaz, amelyen értelmezett az összeadás művelete). Határozzuk meg az f függvény [e..u] intervallumon felvett értékeinek az összegét, azaz az f(e)+f(e+1)+...+f(u) kifejezés értékét!

i		f(i)
e	→	1
		+
e+1	→	3
		+
…	→	2
		+
u-1	→	0
		+
u	→	5

Specifikáció:

```
Be: e∈Z, u∈Z
Ki: s∈H
Ef: -
Uf: s=SZUMMA(i=e..u, f(i))
```

Algoritmus:

```
s:=0
i=e..u
    s:=s+f(i)
```

A megoldás során a kívánt összeget fokozatosan állítjuk elő. Ehhez végig kell vezetnünk egy i egész típusú változót az intervallum elemein, és minden lépésben hozzá kell adni a kezdetben nullára beállított s változóhoz az f(i) értéket. A megoldó program tehát a kezdeti értékadás után (ahol az s változó értékét nullára állítjuk) egy ciklus, amely az i változót egyesével növeli egészen n-ig.

Példa – papírgyűjtés

Példa:
n=6, kg=[1, 3, 2, 0, 5, 2] → össz=13

Feladat:

Határozd meg egy n elemű tömb elemeinek összegét!

Specifikáció:

```
Be: n∈N, kg∈N[1..n]
Ki: össz∈N
Ef: -
Uf: össz=SZUMMA(i=1..n, kg[i])
```

	kg
1	1
2	3
3	2
4	0
5	5
n=6	2

össz= 13

Példa – papírgyűjtés visszavezetés

Vegyük számba a két feladat közötti különbségeket, a mintaprogram algoritmusában pedig cseréljük le a megfelelő részeket!

Feladatsablon

(mintafeladat)

```
Be: e∈Z, u∈Z
Ki: s∈H
Ef: -
Uf: s=SZUMMA(i=e..u, f(i))
```

Papírgyűjtés

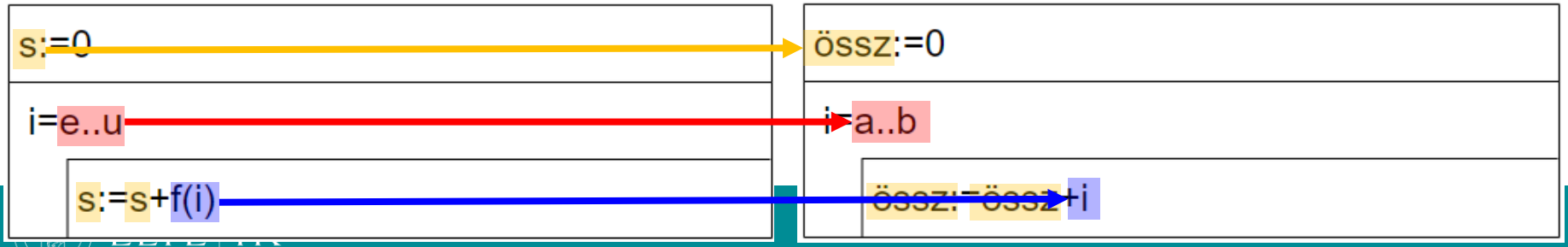
(konkrét feladat)

```
Be: n∈N, kg∈N[1..n]
Ki: össz∈N
Ef: -
Uf: össz=SZUMMA(i=1..n, kg[i])
```

Visszavezetés:

s	~	össz
e..u	~	1..n
f(i)	~	kg[i]

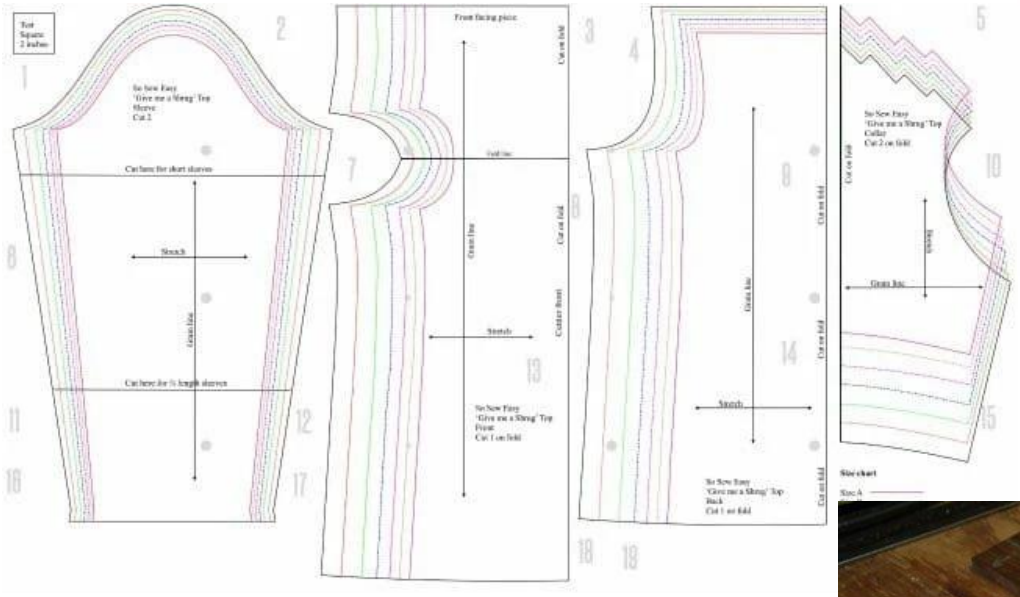
Algoritmus:

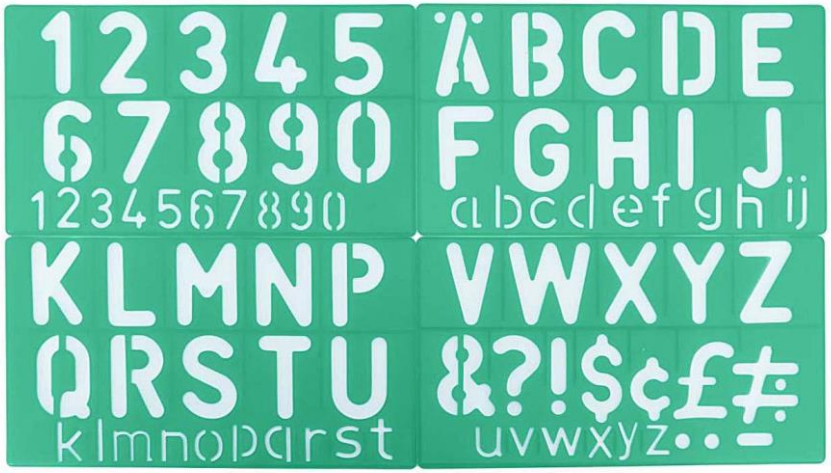


Nevezetes programozási minták

Programozási tételek

Hétköznapi példa





Analóg programozás

- Cél: egy feladat megoldására program készítése
- Minden egyes alkalommal újra és újra kitalálhatjuk a megoldást
- Vagy egy korábbi, hasonló feladat megoldását vehetjük alapul
- **Analóg programozás**: a konkrét feladatot egy korábbi feladat megoldása alapján állítjuk elő **visszavezetéssel**.

Feladattípusok

- A programok ad hoc módon is előállhatnak.
- De: felismerhetjük, hogy vannak megoldások, amelyek többé-kevésbé ugyanazt a feladatot oldják meg csak más és más köntösben
- Feladattípusok
- Ezekhez készíthetünk/létezik bizonyítottan *helyes megoldás*
- Analóg programozás során ezeket használjuk **mintá**nak.

Programozási minta
Programozási tétel

Célja:

Bizonyíthatóan **helyes sablon**, amelyre magasabb szinten lehet építeni a megoldást. (A fejlesztés gyorsabb és biztonságosabb.)

Szerkezete:

1. absztrakt feladat specifikáció
2. absztrakt algoritmus

Egy fontos előzetes **megjegyzés**:

A bemenet legalább egy sorozat…

Programozási minta

Felhasználásának menete:

1. a konkrét feladat specifikálása
2. a specifikációban a programozási minta (PrM) megsejtése
3. a konkrét feladat és az absztrakt feladat paramétereinek egymáshoz rendelése
4. a konkrét algoritmus „generálása” a megsejtett PrM-ek absztrakt algoritmusok alapján, 3. szerint átparaméterezve
5. „hatékonyítás” programtranszformációkkal

Összegzés

Összegzés

Feladatok:

1. Ismerjük egy ember havi bevételeit és kiadásait. Adjuk meg, hogy év végére **mennyivel** nőtt a vagyona!
2. Ismerjük egy autóversenyző körönkénti idejét. Adjuk meg az **átlag**körének idejét!
3. Adjuk meg az N számhoz az N **faktoriális** értékét!
4. Ismerjük egy iskola szakkö szakkörönként. Adjuk meg
5. Ismerünk N szót. Adjuk meg a belőlük összeállított mondatot!

Mi bennük a közös?
n szám összegét kell kiszámolni!

Összegzés
sablon

Feladat

Adott az egész számok egy [e..u] intervalluma és egy f:[e..u]→H függvény. A H halmaz elemein értelmezett az összeadás művelet. Határozzuk meg az f függvény [e..u] intervallumon felvett értékeinek az összegét, azaz a $\sum_{i=e}^{u} f(i)$ kifejezés értékét! (e>u esetén ennek az értéke definíció szerint a nulla elem)

Specifikáció

```
Be: e∈Z, u∈Z
Ki: s∈H
Ef: -
Uf: s=SZUMMA(i=e..u, f(i))
```

Algoritmus

```
s:=0
i=e..u
    s:=s+f(i)
```

Példa – jövedelmek
visszavezetés

Ismerjük egy ember havi bevételeit és kiadásait. Adjuk meg, hogy év végére **mennyivel** nőtt a vagyona!

Feladatsablon
(mintafeladat)

```
Be: e∈Z, u∈Z

Ki: s∈H
Ef: -
Uf: s=SZUMMA(i=e..u, f(i))
```

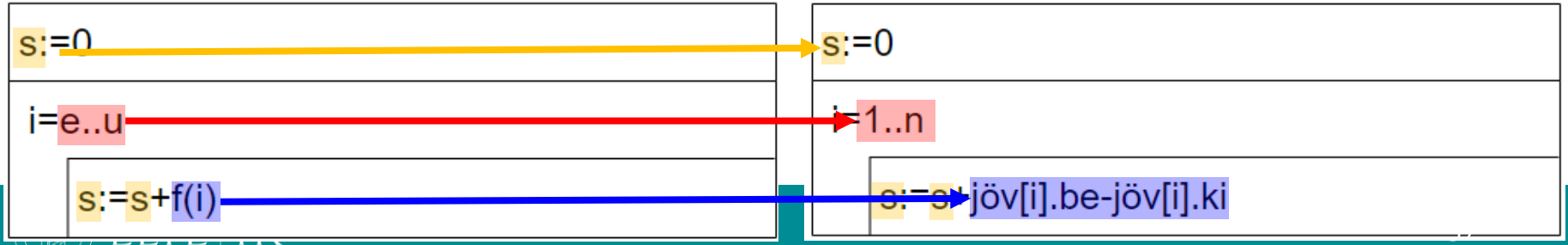
Jövedelmek
(konkrét feladat)

```
Be: n∈N, jöv∈Jövedelem[1..n],
    Jövedelem=Be x Ki, Be=N, Ki=N
Ki: s∈Z
Ef: -
Uf: s=SZUMMA(i=1..n,
                jöv[i].be-jöv[i].ki)
```

Visszavezetés:

e..u	~	1..n
f(i)	~	jöv[i].be-jöv[i].ki

Algoritmus:



Megszámolás

Megszámolás

Mi bennük a közös?

n darab „valamire” kell megadni, hogy hány adott tulajdonságú van közöttük!

Feladatok:

1. Ismerjük egy ember havi bevételeit és kiadásait. Adjunk meg, hogy **hány** hónapban nőtt a vagyona!
2. Adjuk meg egy természetes szám osztói **számá**t!
3. Adjuk meg egy ember nevében levő „a” betűk **számá**t!
4. Adjunk meg az éves statisztika alapján, hogy **hány** napon fagyott!
5. Adjuk meg n születési hónap alapján, hogy közöttük **hány**an születtek télen!

Példa – a betűk száma
analóg algoritmikus gondolkodással

i	név	név[i]="a"	érték
1	a	IGAZ	1
			+
2	l	HAMIS	0
			+
3	m	HAMIS	0
			+
4	a	IGAZ	1
			=
		db=	2

Feladat:

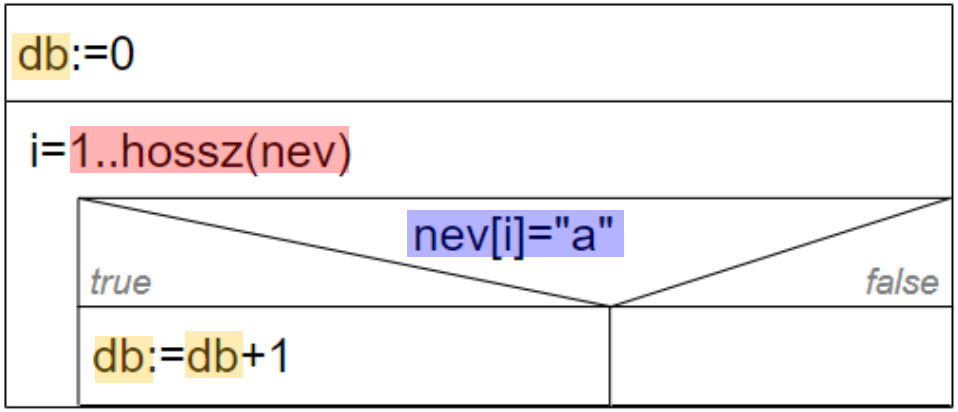
Adjuk meg egy ember nevében levő „a” betűk **számát**!

Specifikáció:

$$db = \sum_{\substack{i=1 \\ nev[i]="a"}}^{hossz(név)} 1$$

```
Be: név∈S
Ki: db∈N
Ef: -
Uf: db=SZUMMA(i=1..hossz(név), 1, név[i]="a")
```

Algoritmus:



Megszámolás
sablon

i	T(i)	érték
e	IGAZ	1
		+
e+1	HAMIS	0
		+
...	HAMIS	0
		+
u	IGAZ	1
		=
	db=	2

Feladat

Adott az egész számok egy [e..u] intervalluma és egy T:[e..u]→Logikai feltétel. Határozzuk meg, hogy az [e..u] intervallumon a T feltétel hányszor veszi fel az igaz értéket!

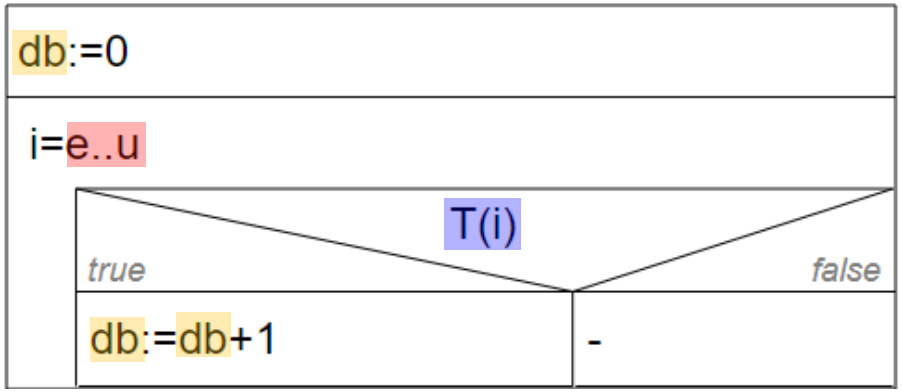
Specifikáció

```
Be: e∈Z, u∈Z
Ki: db∈N
Ef: -
Uf: db=SZUMMA(i=e..u, 1, T(i))
```

Rövidítve:

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

Algoritmus



Példa – téli születések
visszavezetés

Adjuk meg n születési hónap alapján, hogy közöttük hányan születtek télen!

Feladatsablon
(mintafeladat)

```
Be: e∈Z, u∈Z
Ki: db∈N
Ef: -
Uf: db=DARAB(i=e..u, T(i))
```

Téli születések száma
(konkrét feladat)

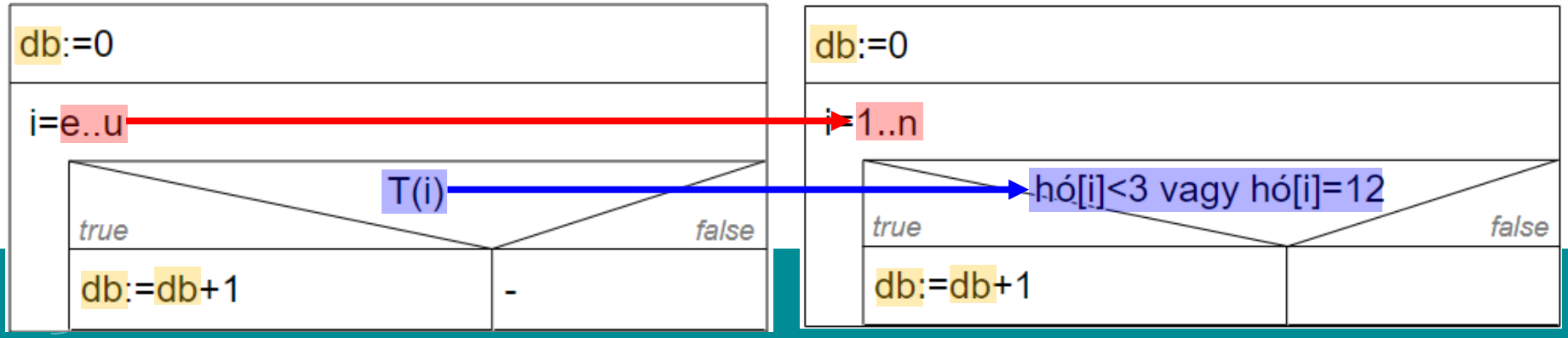
```
Be: n∈N, hó∈N[1..n]
Ki: db∈N
Ef: ∀i∈[1..n]:(1<=hó[i]<=12)
Uf: db=DARAB(i=1..n,
            hó[i]<3 vagy hó[i]=12)
```

Visszavezetés:

```
e..u  ~  1..n
T(i)  ~  hó[i]<3 vagy hó[i]=12
```

Megjegyzés: a konkrét feladat előfeltétele mindig lehet szigorúbb a tétel előfeltételénél!

Algoritmus:



Maximumkiválasztás

Maximumkiválasztás

Feladatok:

1. Ismerjük egy ember havi bevételeit és kiadásait. Adjunk meg, hogy melyik hónapban nőtt **leg**jobban a vagyona!
2. Adjuk meg n ember közül az ábécében **utolsó**t!
3. Adjuk meg n ember közül azt, aki a **leg**több ételt szereti!
4. Adjuk meg n napi statisztika alapján a **leg**melegebb napot!
5. Adjuk meg n születésnap alapján azt, akinek idén **először** van születésnapja!

Mi bennük a közös?

n darab „valami” közül kell megadni a legnagyobbat (vagy a legkisebbet)!

Fontos:

A „valamik” között értelmezhető egy **rendezési reláció**.

Ha **legalább 1** „valamink” van, akkor legnagyobb (legkisebb) is biztosan van közöttük!

Példa – legmelegebb nap
analóg algoritmikus gondolko

i	hőm	nap (i=1)	nap (i=2)	nap (i=3)	nap (i=4)
1	13,5	13,5	13,5	13,5	13,5
2	12,6	12,6	12,6	12,6	12,6
3	14,8	14,8	14,8	14,8	14,8
4	10,2	10,2	10,2	10,2	10,2

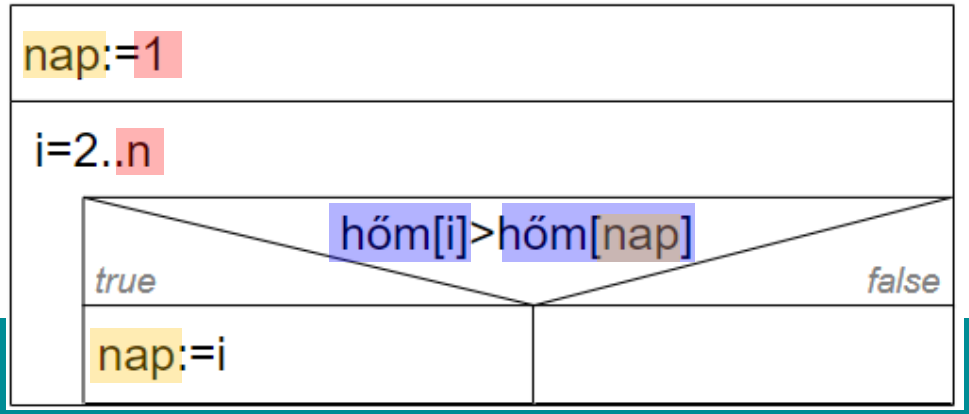
Feladat:

Adjuk meg n napi statisztika alapján a **leg**melegebb napot!

Specifikáció:

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

Algoritmus:



Maximumkiválasztás
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 legnagyobb értéket, és mondjuk meg, mekkora ez a maximális érték!

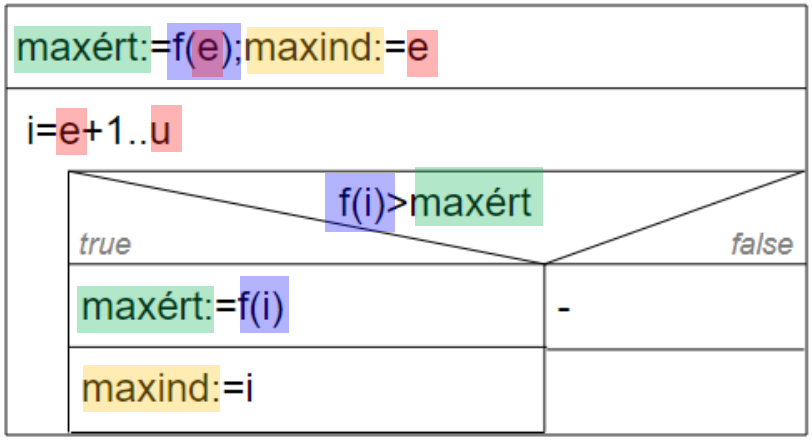
Specifikáció

```
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))
```

Algoritmus



Példa – ábécében utolsó
visszavezetés

Adjuk meg n ember közül az ábécében utolsót!

Feladatsablon

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

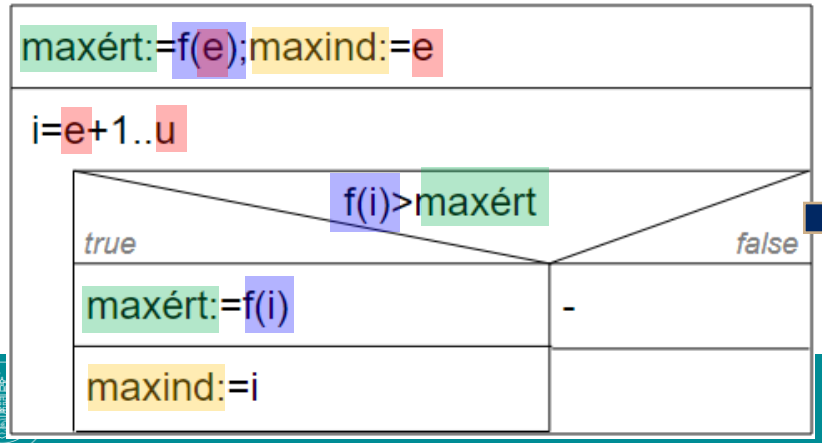
Ábécében utolsó név

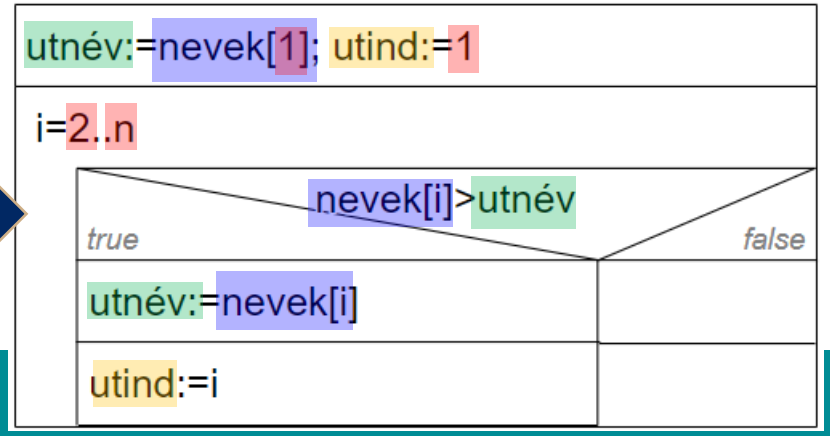
```
Be: n∈N, nevek∈S[1..n]
Ki: utind∈N, utnév∈S
Ef: n>0
Uf: (utind,utnév)=
                    MAX(i=1..n,nevek[i])
```

Visszavezetés:

maxind, maxért	~	utind, utnév
e..u	~	1..n
f(i)	~	nevek[i]

Algoritmus:





Feltételes maximumkeresés

Feltételes maximumkeresés

Feladatok:

1. Ismerjük egy ember havi bevételeit és kiadásait. Adjuk meg, hogy melyik hónapban volt a **leg**nagyobb **vesztesége**!
2. Adjuk meg n napi statisztika alapján a **leg**melegebb **fagyos** napot!
3. Egy hibakezelő szoftverben számokkal jelezzük a hibák súlyosságát. Adjuk meg a **leg**súlyosabb hibát (**ha van**)!

Mi bennük a közös?

n darab „valami” adott tulajdonságú elemei közül kell megadni a legnagyobbat (vagy a legkisebbet)!

Fontos:

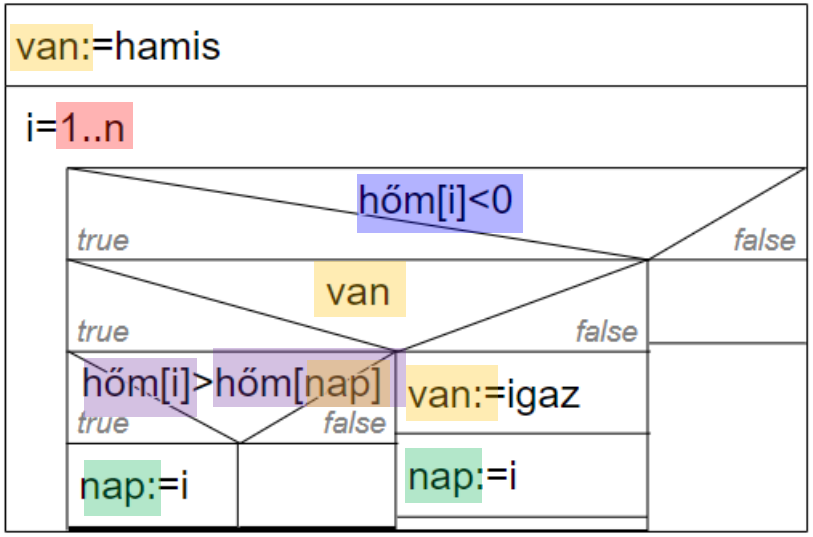
A „valamik” között értelmezhető egy **rendezési reláció**.
Nem biztos, hogy **van** adott tulajdonságú „valami”.
Nem biztos, hogy **egyáltalán van** „valami”-nk.

Példa – legmelegebb fagyos nap
analóg algoritmikus gondolkodással

Feladat: Adjuk meg n napi statisztika alapján a **leg**melegebb **fagyos** napot!

i	hőm	van?	maxért?
1	2,3	HAMIS	???
2	-1,3	IGAZ	-1,3
3	-0,5	IGAZ	-0,5
4	0,4	IGAZ	-0,5

i	hőm	van?	maxért?
1	2,3	HAMIS	???
2	1,3	HAMIS	???
3	0,5	HAMIS	???
4	0,4	HAMIS	???



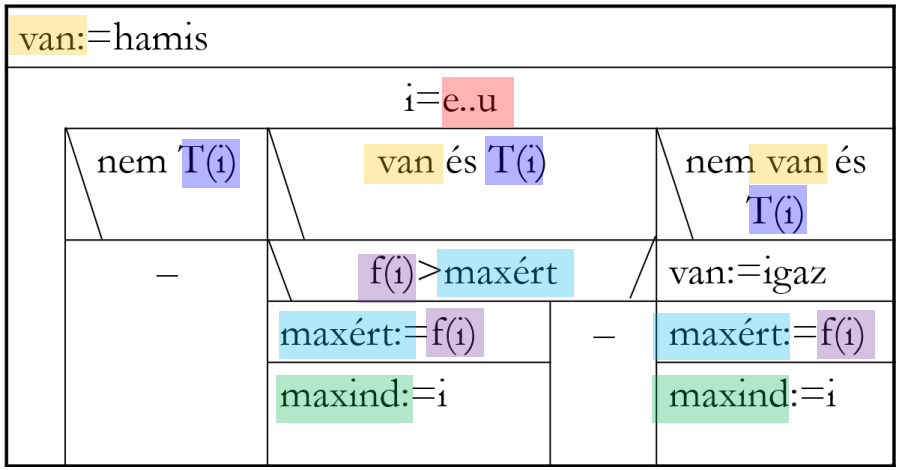
Specifikáció és algoritmus:

```
Be: n∈N, hőm∈R[1..n]
Ki: van∈L, nap∈N
Ef: ∀i∈[1..n]:(-100<=hőm[i]<=100)
Uf: van=∃i∈[1..n]:(hőm[i]<0) és
    van->(nap∈[1..n] és
          hőm[nap]<0 és
          ∀i∈[1..n]:(hőm[i]<0 -> hőm[nap]>=hőm[i]))
```

Feltételes maximumkeresés sablon

Feladat

Adott az egész számok egy [e..u] intervalluma, egy f:[e..u]→H függvény és egy T:[e..u]→Logikai feltétel. A H halmaz elemein értelmezett egy teljes rendezési reláció. Határozzuk meg, hogy az [e..u] intervallum T feltételt kielégítő elemei közül az f függvény hol veszi fel a legnagyobb értéket, és mondjuk meg, mekkora ez az érték!



Specifikáció és algoritmus:

```
Be: e∈Z, u∈Z
Ki: van∈L, maxind∈Z, maxért∈H
Ef: -
Uf: van = ∃i∈[e..u]:(T(i)) és
    van -> (maxind∈[e..u] és
            maxért=f(maxind) és T(maxind) és
            ∀i∈[e..u]:(T(i) -> maxért>=f(i)))
```

Rövidítve:

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

Példa – súlyos hiba visszavezetés

Egy hibakezelő szoftverben számokkal jelezzük a hibák súlyosságát. Adjuk meg a legsúlyosabb hibát (ha van)!

Feladatsablon

```
Be: e∈Z, u∈Z

Ki: van∈L, maxind∈Z, maxért∈H
Ef: -
Uf: (van,maxind,maxért)=
                MAX(i=e..u,f(i),T(i))
```

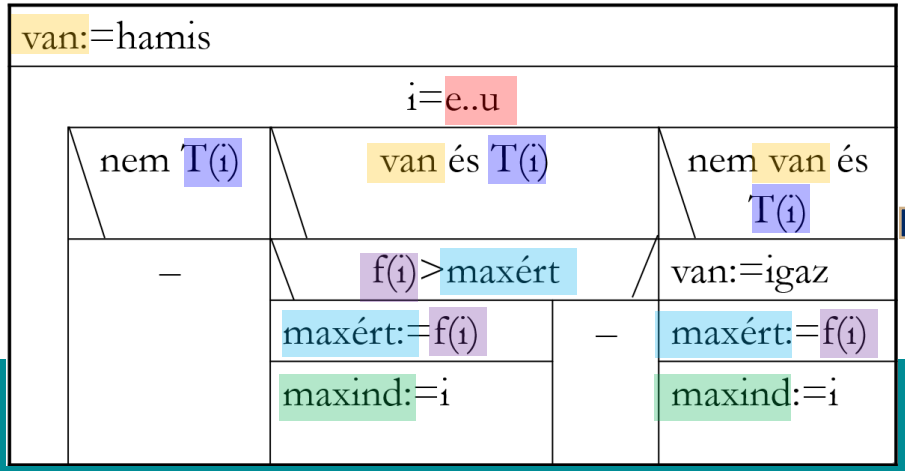
Legsúlyosabb hiba

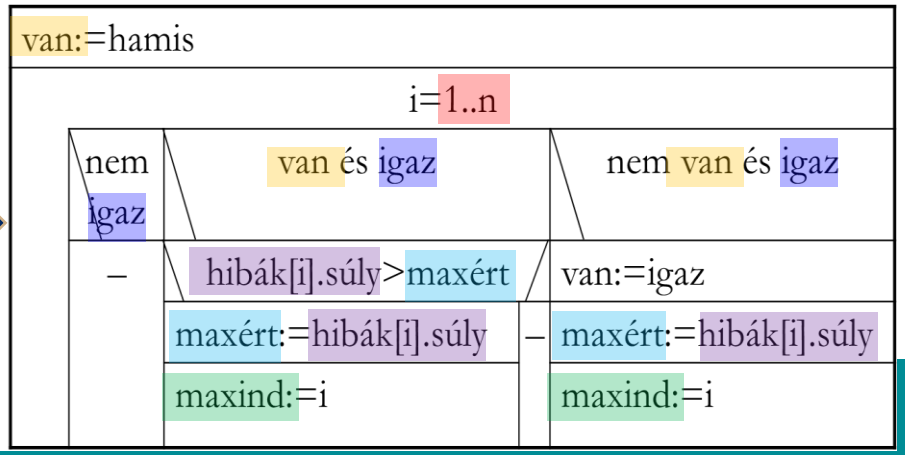
```
Be: n∈N, hibák∈Hiba[1..n],
    Hiba=Név x Súly, Név=S, Súly=N
Ki: van∈L, lsh∈S
Ef: ∀i∈[1..n]:(1<=hibák[i].súly<=5)
Uf: (van,maxind,maxért)=
    MAX(i=1..n,hibák[i].súly,igaz) és
van->lsh=hibák[maxind].név
```

Visszavezetés:

e..u	~	1..n
f(i)	~	hibák[i].súly
T(i)	~	igaz

Algoritmus:





Keresés

Keresés

Mi bennük a közös?

N darab „valami” közül kell megadni egy adott tulajdonságút, ha nem tudjuk, hogy ilyen elem van-e!

Feladatok:

1. Ismerjük egy ember havi bevételeit és kiadásait. Év végére nőtt a vagyona. **Adjunk meg egy** hónapot, amikor **nem** nőtt a vagyona!
2. **Adjuk meg egy** természetes szám egy 1-től és önmagától különböző legkisebb osztóját!
3. **Adjuk meg egy** ember nevében egy „**a**” betű helyét!
4. **Adjunk meg egy** tanulóra egy tárgyat, amiből megbukott!
5. **Adjuk meg egy** számsorozat olyan elemét, amely nagyobb az előzőnél!

Példa – legkisebb osztó
analóg algoritmikus gondolkodással

Feladat:

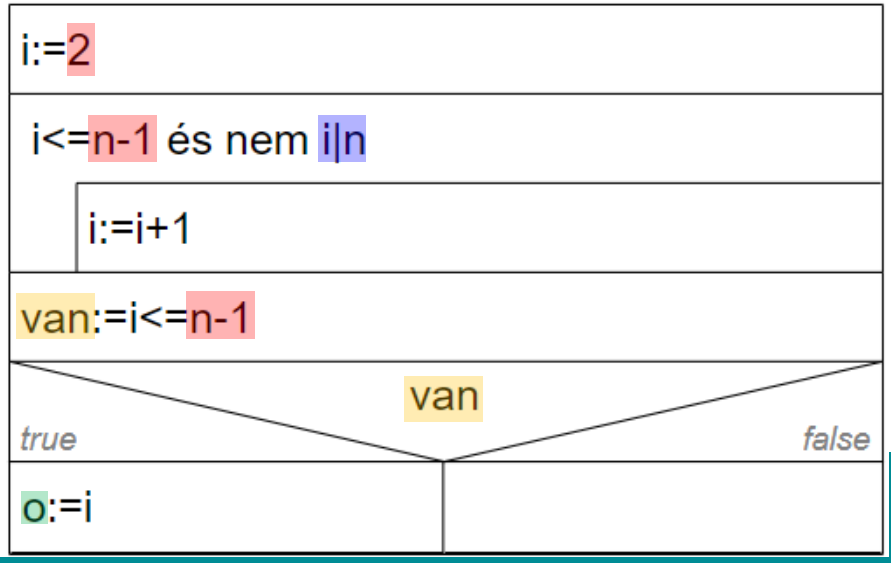
Határozzuk meg egy természetes szám (n>1) 1-től és önmagától különböző legkisebb osztóját!

Specifikáció és algoritmus:

```
Be: n∈N
Ki: o∈N, van∈L
Ef: n>1
Uf: van=∃i∈[2..n-1]:(i|n) és
    van->(2<=o<n és o|n és
          ∀i∈[2..o-1]:(i∤n))
```

	i	i<=8?	i ∤ n?
	1		
e=	2	IGAZ	IGAZ
	3	IGAZ	HAMIS
	4		
	5		
	6		
	7		
u=	8		
n=	9		

	i	i<=4?	i ∤ n?
	1		
e=	2	IGAZ	IGAZ
	3	IGAZ	IGAZ
u=	4	IGAZ	IGAZ
n=	5	HAMIS	



Keresés
sablon

Feladat

Adott az egész számok egy [e..u] intervalluma és egy T:[e..u]→Logikai feltétel. Határozzuk meg az [e..u] intervallumban balról az első olyan számot, ha van, amely kielégíti a T feltételt!

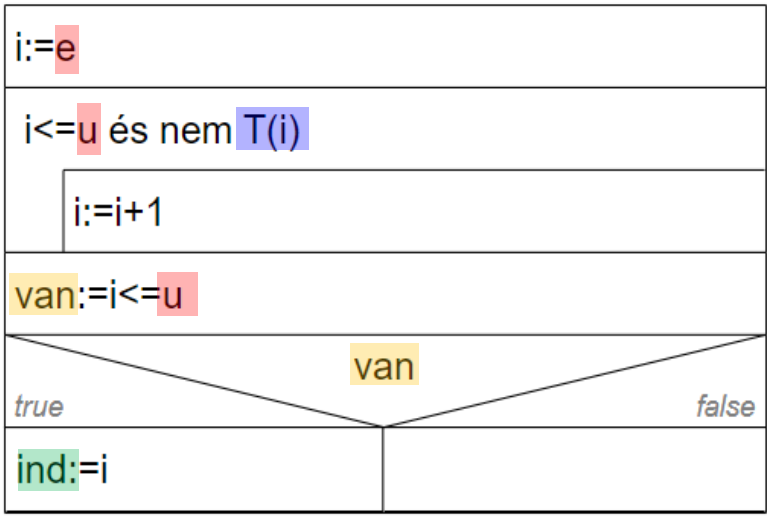
Specifikáció

```
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∈[e..ind-1]:(nem T(i)))
```

Rövidítve:

```
Uf: (van,ind)=KERES(i=e..u,T(i))
```

Algoritmus



Keresés
sablon

Feladat

Adott az egész számok egy [e..u] intervalluma és egy T:[e..u]→Logikai feltétel. Határozzuk meg az [e..u] intervallumban balról az első olyan számot, ha van, amely kielégíti a T feltételt!

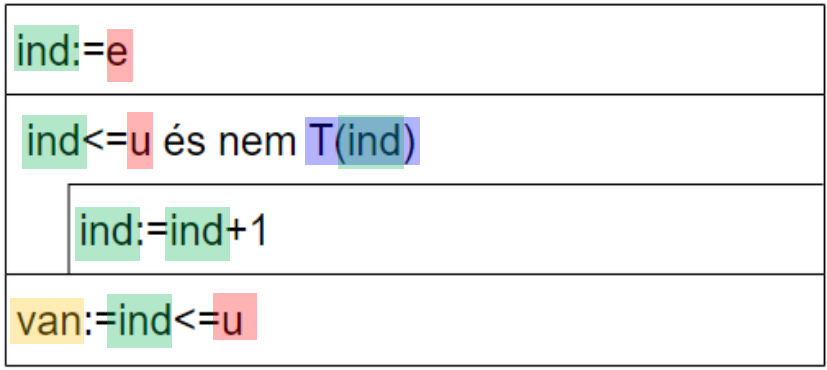
Specifikáció

```
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∈[e..ind-1]:(nem T(i)))
```

Rövidítve:

```
Uf: (van,ind)=KERES(i=e..u,T(i))
```

Algoritmus



Keresés
sablon

Feladat

Adott az egész számok egy [e..u] intervalluma és egy T:[e..u]→Logikai feltétel. Határozzuk meg az [e..u] intervallumban balról az első olyan számot, ha van, amely kielégíti a T feltételt!

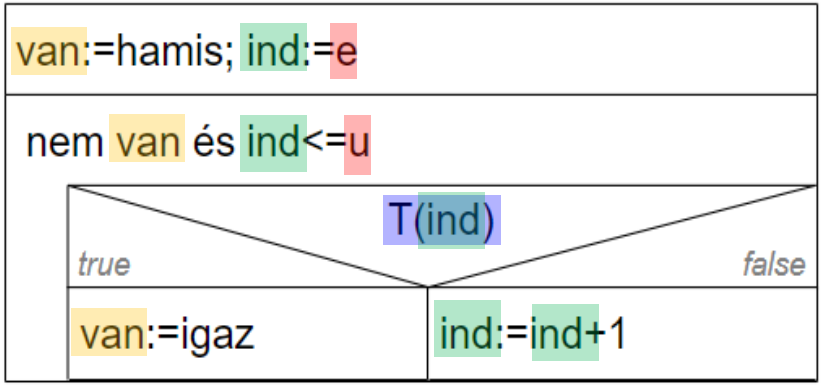
Specifikáció

```
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∈[e..ind-1]:(nem T(i)))
```

Rövidítve:

```
Uf: (van,ind)=KERES(i=e..u,T(i))
```

Algoritmus



Példa – legkisebb osztó visszavezetés

Határozzuk meg egy természetes szám (n>1) 1-től és önmagától különböző legkisebb osztóját!

Feladatsablon

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

Legkisebb osztó

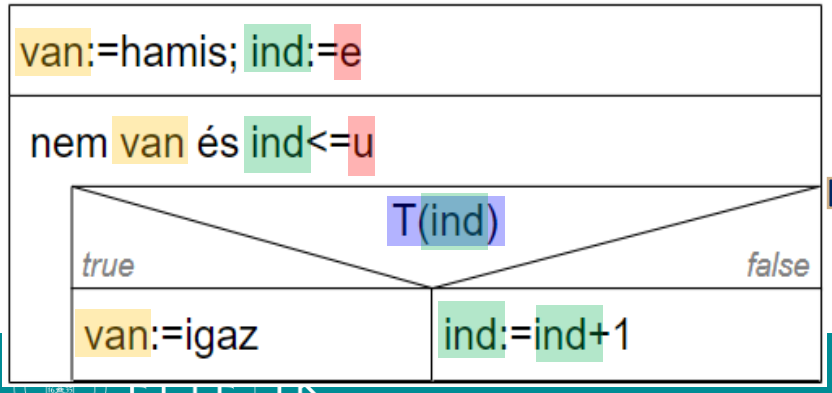
```
Be: n∈N
Ki: o∈N, van∈L
Ef: n>1
Uf: (van,o)=KERES(i=2..n-1,i|n)
```

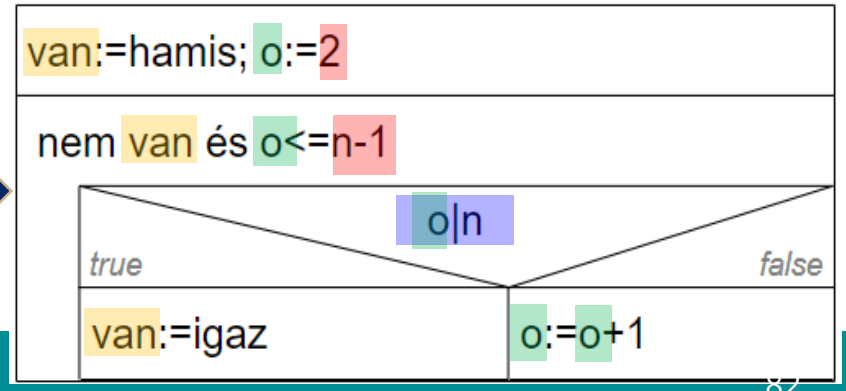
Visszavezetés:

ind	~	o
e..u	~	2..n-1
T(i)	~	i|n

Megjegyzés: a konkrét feladat előfeltétele mindig lehet szigorúbb a tétel előfeltételénél!

Algoritmus:





Eldöntés

Eldöntés

Mi bennük a közös?

Döntsük el, hogy N „valami” között van-e adott tulajdonsággal rendelkező elem!

Ez a keresés programozási tétel (kimenetének) szűkítése.

Feladatok:

1. Egy természetes számról **döntsük el**, hogy prímszám**-e**!
2. Egy szóról **mondjuk meg**, hogy egy hónapnak a neve**-e**!
3. Egy tanuló év végi osztályzatai alapján **állapítsuk meg**, hogy bukott**-e**!
4. Egy szóról **adjuk meg**, hogy van**-e** benne magánhangzó!
5. Egy számsorozatról **döntsük el**, hogy monoton növekvő**-e**!
6. Egy tanuló év végi jegyei alapján **adjuk meg**, hogy kitűnő**-e**!

Példa – hónapnév
analóg algoritmikus gondo

	HÓNÉV	i<=12 és nem egyezik április	i<=12 és nem egyezik tavasz
1	január	IGAZ	IGAZ
2	február	IGAZ	IGAZ
3	március	IGAZ	IGAZ
4	április	HAMIS	IGAZ
5	május		IGAZ
6	június		IGAZ
7	július		IGAZ
8	augusztus		IGAZ
9	szeptember		IGAZ
10	október		IGAZ
11	november		IGAZ
12	december		IGAZ
13			HAMIS

Feladat:

Egy szóról **mondjuk meg**, hogy egy hónapnak a neve-**e**!

Specifikáció:

```
Be: név∈S, HÓNÉV∈S[1..12]={"január","február",...}
Ki: hónapnéve∈L
Ef: -
Uf: hónapnéve=∃i∈[1..12]:(HÓNÉV[i]=név)
```

Algoritmus:

```
i:=1
i<=12 és nem HÓNÉV[i]=név
    i:=i+1
hónapnéve:=i<=12
```

Eldöntés
sablon

Feladat

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

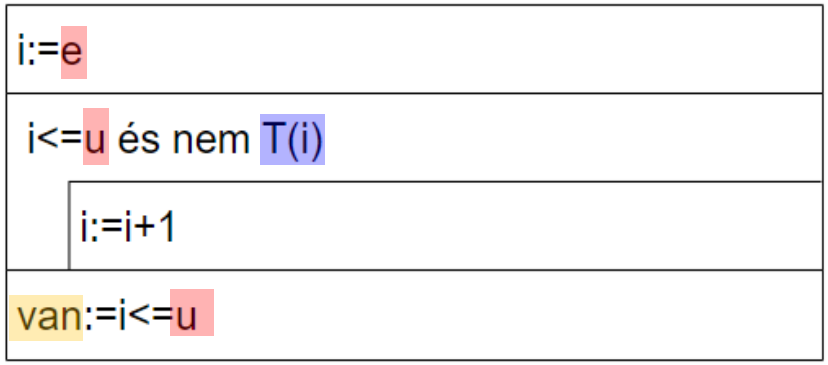
Specifikáció

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

Rövidítve:

```
Uf: van=VAN(i=e..u,T(i))
```

Algoritmus



Eldöntés
sablon

Feladat

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

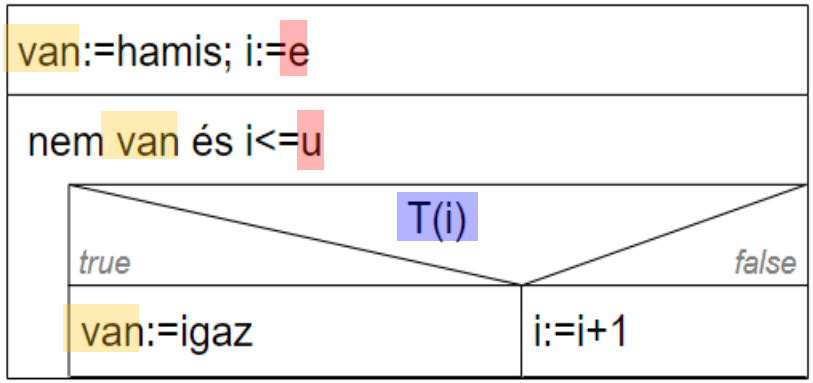
Specifikáció

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

Rövidítve:

```
Uf: van=VAN(i=e..u,T(i))
```

Algoritmus



Példa – bukott-e
visszavezetés

Egy tanuló év végi osztályzatai alapján állapítsuk meg, hogy bukott-e!

Feladatsablon

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

Bukott-e

```
Be: n∈N, jegyek∈N[1..n]
Ki: bukott∈L
Ef: ∀i∈[1..n]:(1<=jegyek[i]<=5)
Uf: bukott=VAN(i=1..n,jegyek[i]=1)
```

Visszavezetés:

van	~	bukott
e..u	~	1..n
T(i)	~	jegyek[i]=1

Algoritmus:

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

```
i:=1
i<=n és nem jegyek[i]=1
    i:=i+1
bukott:=i<=n
```

Kiválasztás

Kiválasztás

Feladatok:

1. Ismerjük egy ember havi bevételeit és kiadásait. Év végére nőtt a vagyona. **Adjunk meg egy** hónapot, amikor nőtt a vagyona!
2. **Adjuk meg egy** 1-nél nagyobb természetes szám egytől különböző legkisebb osztóját!
3. **Adjuk meg egy** magyar szó egy magánhangzóját!
4. **Adjuk meg egy** hónapnévről a sorszámát!

Mi bennük a közös?

N „valami” közül kell megadni egy adott tulajdonságút, ha tudjuk, hogy ilyen elem biztosan van.

Ez a keresés programozási tétel olyan változata, amelyben nem kell felkészülnünk arra, hogy a keresett elemet nem találjuk meg.

Példa – legkisebb osztó
analóg algoritmikus gondolkodással

Feladat:

Határozzuk meg egy természetes szám (n>1) 1-től különböző legkisebb osztóját!

Specifikáció és algoritmus:

```
Be: n∈N
Ki: o∈N
Ef: n>1
Uf: 1<o<=n és o|n és
    ∀i∈[2..o-1]:(i ∤ n)
```

	i	i<=9?	i ∤ n?
	1		
e=	2	IGAZ	IGAZ
	3	IGAZ	HAMIS
	4		
	5		
	6		
	7		
	8		
n=	9		

	i	i<=5?	i ∤ n?
	1		
e=	2	IGAZ	IGAZ
	3	IGAZ	IGAZ
	4	IGAZ	IGAZ
n=	5	IGAZ	HAMIS

```
i:=2
i ∤ n
    i:=i+1
o:=i
```

Kiválasztás
sablon

Feladat

Adott egy e egész szám és egy e-től jobbra értelmezett T:Egész→Logikai feltétel. Határozzuk meg az e-től jobbra eső első olyan számot, amely kielégíti a T feltételt, ha tudjuk, hogy ilyen szám biztosan van!

Specifikáció

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

Rövidítve:

```
Uf: ind=KIVÁLASZT(i>=e,T(i))
```

Algoritmus

```
i:=e
nem T(i)
    i:=i+1
ind:=i
```

```
ind:=e
nem T(ind)
    ind:=ind+1
```

Példa – magánhangzó-e
visszavezetés

Adjuk meg egy magyar szó egy magánhangzóját!

Feladatsablon

```
Be: e∈Z
Ki: ind∈Z

Ef: ∃i∈[e..∞]:(T(i))

Uf: ind=KIVÁLASZT(i>=e,T(i))
```

Magánhangzó

```
Be: szó∈S
Ki: mh∈N
Fv: magánhangzó:K->L,
    magánhangzó(k)=k="a" vagy ...
Ef: ∃i∈[1..hossz(szó)]:
            (magánhangzó(szó[i]))
Uf: mh=KIVÁLASZT(i>=1,
            magánhangzó(szó[i]))
```

T: tulajdonságfüggvény

Visszavezetés:

ind	~	mh
e	~	1
T(i)	~	magánhangzó(szó[i])

Algoritmus:

```
ind:=e
nem T(ind)
    ind:=ind+1
```

```
mh:=1
nem magánhangzó(szó[mh])
    mh:=mh+1
```

Összefoglalás

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

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

Programozási minták

1. Összegzés
2. Megszámolás

üres intervallumra is

3. Maximumkiválasztás *legalább 1 elemű intervallum*

4. Feltételes maximumkeresés
5. Keresés *üres intervallumra is*
6. Eldöntés

7. Kiválasztás *legalább 1 elemű intervallum*

Ellenőrző kérdések

Ellenőrző kérdések

1. Milyen adatszerezettel írjuk le, ha több különböző funkciójú adatot szeretnénk egységbe foglalni?
2. Milyen adatszerkezettel írjuk le, ha több ugyanolyan funkciójú adatot szeretnénk egységbe foglalni?
3. Hogyan hívjuk a tömböt a specifikációban?
4. Mik a visszavezetés lépései?
5. Add meg az egyes progamozási minták specifikációit!
6. Add meg az egyes programozási minták algoritmusait!