



ELTE | IK

PROGRAMOZÁS

Több programozási minta
együttes használata

Horváth Győző, Pluhár Zsuzsa



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

Most Common DUPLO Parts



Brick Architect 



Több programozási minta együttes használata egymás után



Több minta alkalmazása

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

Példa

Barbenheimer

Feladat:

Egy mozi a nyáron nyilvántartásba vette, hogy melyik filmre milyen nemű néző vett jegyet. Igaz-e, hogy arányaiban a Barbie című filmet több lány nézte meg, mint ahány fiú az Oppenheimert?



Példa

Barbenheimer



Feladat:

Egy mozi a nyáron nyilvántartásba vette, hogy melyik filmre milyen nemű néző vett jegyet. Igaz-e, hogy arányaiban a Barbie című filmet több lány nézte meg, mint ahány fiú az Oppenheimert?

Film	Nem
Barbie	L
Oppenheimer	L
Oppenheimer	F
Barbie	L
Barbie	F
Barbie	L
Oppenheimer	L
Barbie	L
Oppenheimer	F
Oppenheimer	F
Barbie	L
Barbie	L
Oppenheimer	F

1

- Barbie = 7 db
- Oppenheimer = 6 db
- Barbie és lány = 6 db
- Oppenheimer és fiú = 4 db

4 megszámolás

2

- Barbie: 6/7
- Oppenheimer: 4/6

3

→ igaz

Példa

Barbenheimer



Feladat:

Igaz-e, hogy arányaiban a Barbie című filmet több lány nézte meg, mint ahány fiú az Oppenheimert?

Specifikáció:

Be: $n \in \mathbb{N}$, $\text{nézők} \in \text{Néző}[1..n]$, $\text{Néző} = (\text{film}: S \times \text{nem}: K)$
Sa: $\text{dbb} \in \mathbb{N}$, $\text{dbo} \in \mathbb{N}$, $\text{dbbl} \in \mathbb{N}$, $\text{dbof} \in \mathbb{N}$
Ki: $\text{barbieanyerő} \in \mathbb{L}$
Ef: $\forall i \in [1..n]: (\text{nézők}[i].\text{nem} = "L" \text{ vagy } \text{nézők}[i].\text{nem} = "F")$
Uf: $\text{dbb} = \text{DARAB}(i=1..n, \text{nézők}[i].\text{film} = "Barbie")$ és
 $\text{dbo} = \text{DARAB}(i=1..n, \text{nézők}[i].\text{film} = "Oppenheimer")$ és
 $\text{dbbl} = \text{DARAB}(i=1..n, \text{nézők}[i].\text{film} = "Barbie" \text{ és } \text{nézők}[i].\text{nem} = "L")$ és
 $\text{dbof} = \text{DARAB}(i=1..n, \text{nézők}[i].\text{film} = "Oppenheimer" \text{ és } \text{nézők}[i].\text{nem} = "F")$ és
 $\text{barbieanyerő} = (\text{dbbl} / \text{dbb}) > (\text{dbof} / \text{dbo})$

1

3

2

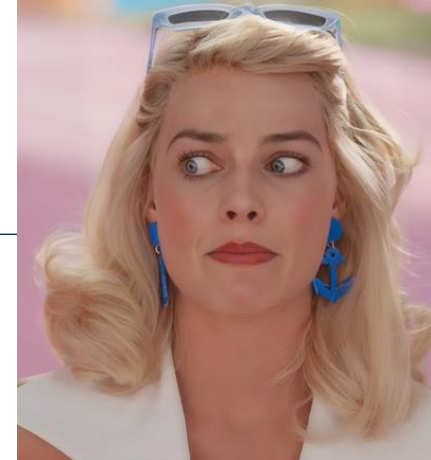
Másképp (lebegőpontos ábrázolás elkerülése végett):

$\text{dbbl} * \text{dbo} > \text{dbof} * \text{dbb}$



Példa

Barbenheimer



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

Feladat:

Uf: `dbb= DARAB(i=1..n, nézők[i].film="Barbie") és`
`dbo= DARAB(i=1..n, nézők[i].film="Oppenheimer") és`
`dbbl=DARAB(i=1..n, nézők[i].film="Barbie" és`
`nézők[i].nem="L") és`
`dbof=DARAB(i=1..n, nézők[i].film="Oppenheimer" és`
`nézők[i].nem="F") és`

1

`db` ~ `dbb`
`e..u` ~ `1..n`
`T(i)` ~ `nézők[i].film="Barbie"`

`db` ~ `dbbl`
`e..u` ~ `1..n`
`T(i)` ~ `nézők[i].film="Barbie" és`
`nézők[i].nem="L"`

`db` ~ `dbo`
`e..u` ~ `1..n`
`T(i)` ~ `nézők[i].film="Oppenheimer"`

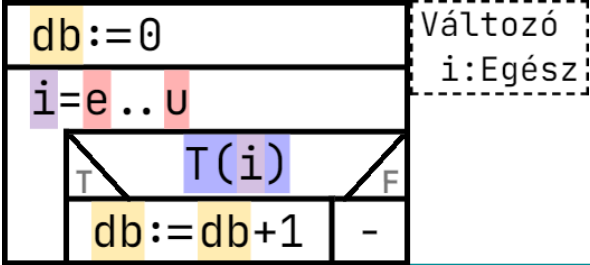
`db` ~ `dbof`
`e..u` ~ `1..n`
`T(i)` ~ `nézők[i].film="Oppenheimer"`
`és nézők[i].nem="F"`

Példa

Barbenheimer

HF: programtranszformációkkal egyszerűbbé tenni!

Sablon és feladat:



Változó
i:Egész

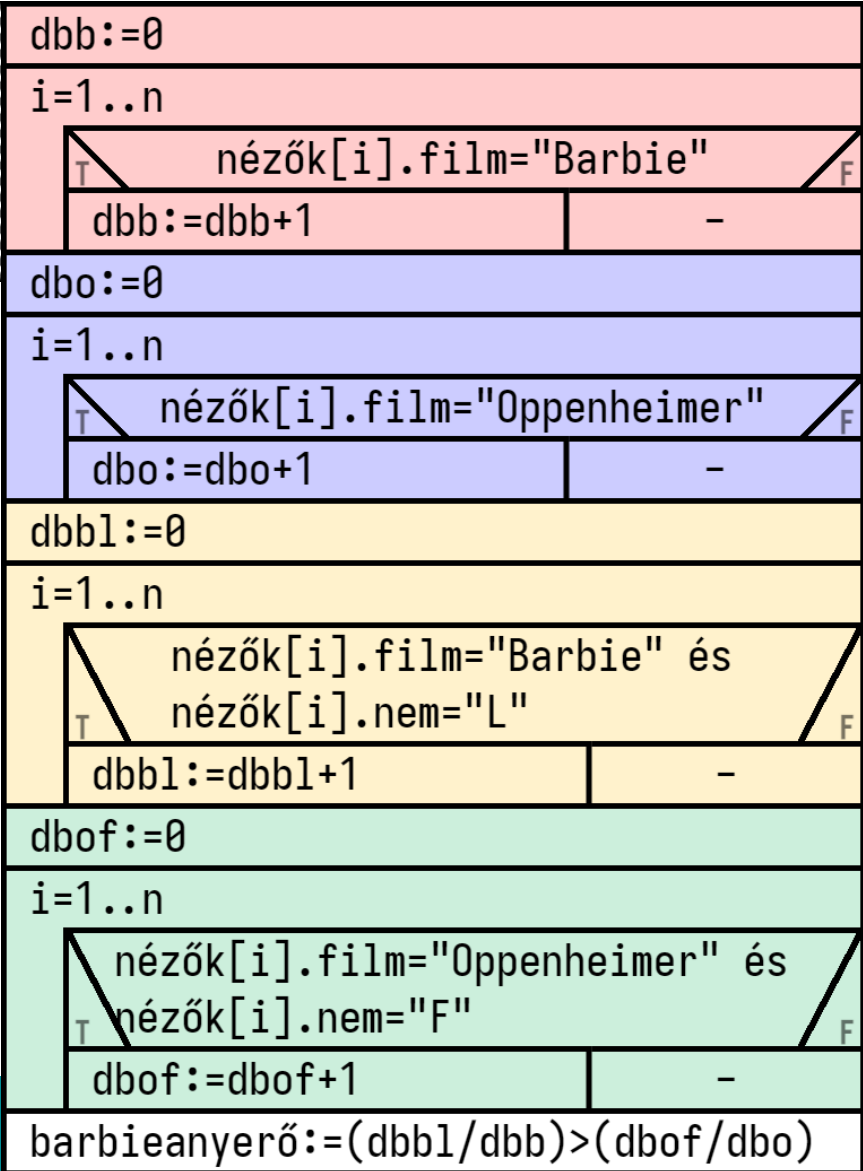
Változó
i:Egész,
dbb:Egész,
dbo:Egész,
dbbl:Egész,
dbof:Egész

db ~ dbb
e..u ~ 1..n
T(i) ~ nézők[i].film="Barbie"

db ~ dbo
e..u ~ 1..n
T(i) ~ nézők[i].film="Oppenheimer"

db ~ dbbl
e..u ~ 1..n
T(i) ~ nézők[i].film="Barbie" és
nézők[i].nem="L"

db ~ dbof
e..u ~ 1..n
T(i) ~ nézők[i].film="Oppenheimer"
és nézők[i].nem="F"



Példa

Barbenheimer



Tanulság:

- specifikációban egymás után (és)
- algoritmusban is egymás után (szekvencia)
- **segédadatok** közvetítenek

b
e
m
e
n
e
t

Film	Nem
Barbie	L
Oppenheimer	L
Oppenheimer	F
Barbie	L
Barbie	F
Barbie	L
Oppenheimer	L
Barbie	L
Oppenheimer	F
Oppenheimer	F
Barbie	L
Barbie	L
Oppenheimer	F

1

- Barbie = 7 db
- Oppenheimer = 6 db
- Barbie és lány = 6 db
- Oppenheimer és fiú = 4 db

segédadatok

2

- Barbie: 6/7
- Oppenheimer: 4/6

→ igaz

3

k
i
m
e
n
e
t

Példa

Barbenheimer



Tanulság:

- specifikációban egymás után (és)
- algoritmusban is egymás után (szekvencia)
- **segédadatok** közvetítenek

Specifikáció:

Be: $n \in \mathbb{N}$, $\text{nézők} \in \text{Néző}[1..n]$, $\text{Néző} = \text{Film} \times \text{Nem}$, $\text{Film} = S$, $\text{Nem} = K$

Ki: $\text{barbieanyerő} \in \mathbb{L}$

Sa: $\text{dbb} \in \mathbb{N}$, $\text{dbo} \in \mathbb{N}$, $\text{dbbl} \in \mathbb{N}$, $\text{dbof} \in \mathbb{N}$

Ef: $\forall i \in [1..n]: (\text{nézők}[i].\text{nem} = "L" \text{ vagy } \text{nézők}[i].\text{nem} = "F")$

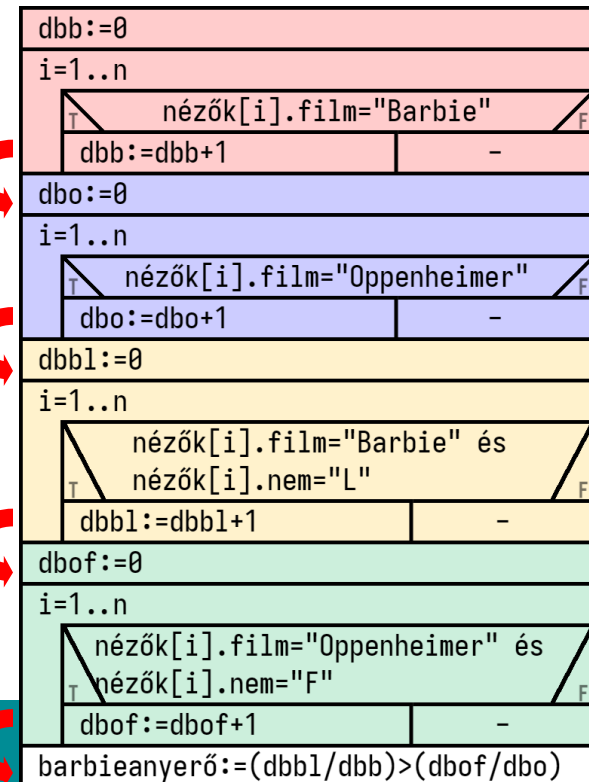
Uf: $\text{dbb} = \text{DARAB}(i=1..n, \text{nézők}[i].\text{film} = "Barbie")$ és

$\text{dbo} = \text{DARAB}(i=1..n, \text{nézők}[i].\text{film} = "Oppenheimer")$ és

$\text{dbbl} = \text{DARAB}(i=1..n, \text{nézők}[i].\text{film} = "Barbie" \text{ és } \text{nézők}[i].\text{nem} = "L")$ és

$\text{dbof} = \text{DARAB}(i=1..n, \text{nézők}[i].\text{film} = "Oppenheimer" \text{ és } \text{nézők}[i].\text{nem} = "F")$ és

$\text{barbieanyerő} = (\text{dbbl} / \text{dbb}) > (\text{dbof} / \text{dbo})$

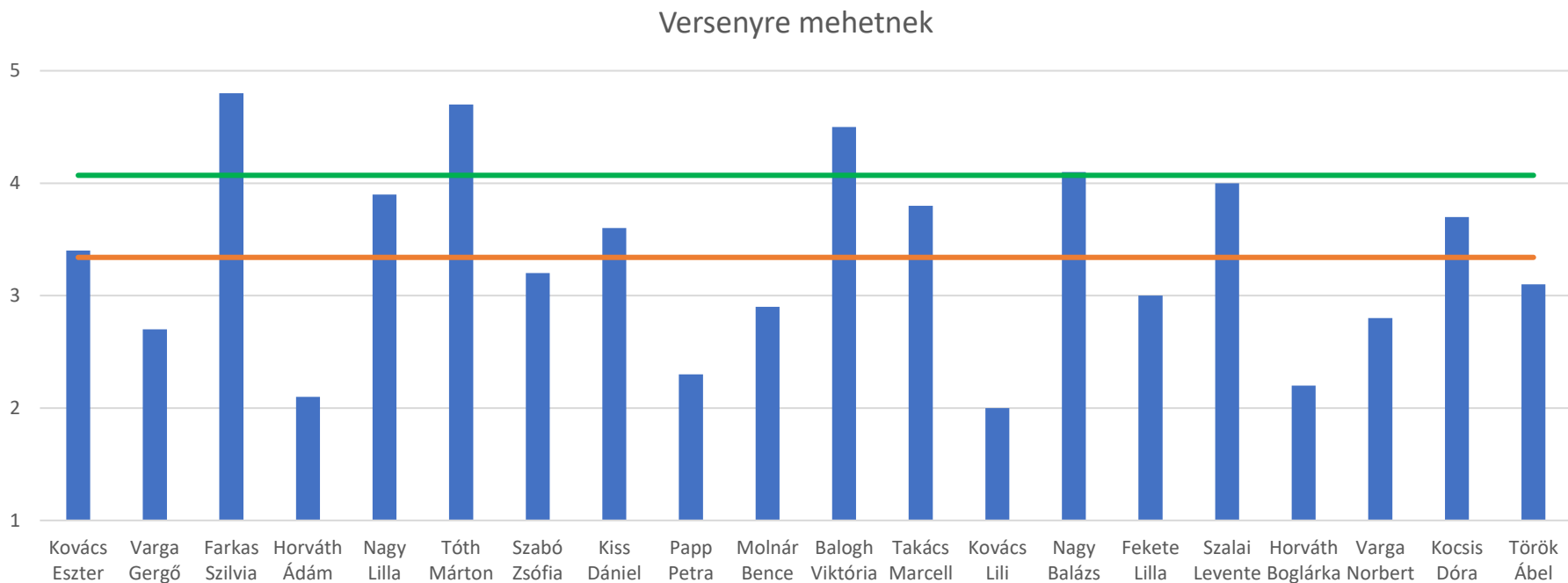


Példa

versenyválogatás

Feladat:

Egy iskolában a tanárnő azokat a gyereket nevezi be a versenyre, akiknek tanulmányi eredménye a legjobb tanuló átlaga és az osztályátlag közötti tartomány felső felébe esik. Kik ők?

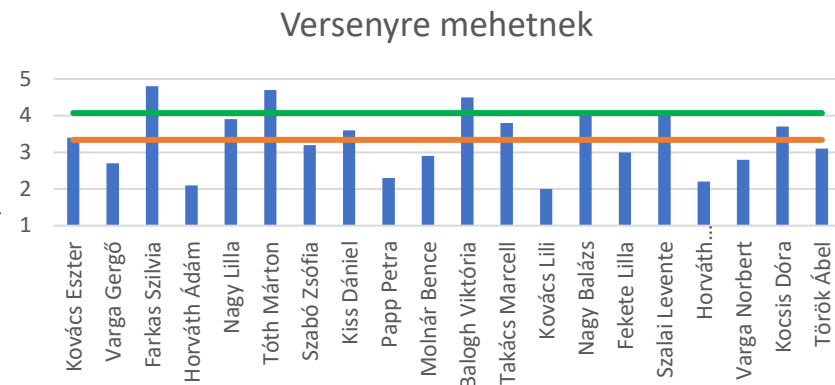


Példa versenyválogatás

Feladat:

Egy iskolában a tanárnő azokat a gyereket nevezi be a versenyre, akiknek tanulmányi eredménye a legjobb tanuló átlaga és az osztályátlag közötti tartomány felső felébe esik. Kik ők?

Név	Jegyátlag
Kovács Eszter	3,4
Varga Gergő	2,7
Farkas Szilvia	4,8
Horváth Ádám	2,1
Nagy Lilla	3,9
Tóth Márton	4,7
Szabó Zsófia	3,2
Kiss Dániel	3,6
Papp Petra	2,3
Molnár Bence	2,9
Balogh Viktória	4,5
Takács Marcell	3,8
Kovács Lili	2
Nagy Balázs	4,1
Fekete Lilla	3
Szalai Levente	4
Horváth Boglárka	2,2
Varga Norbert	2,8
Kocsis Dóra	3,7
Török Ábel	3,1



Gondolkodás:

1. Több diákot kell megadni (**kiválogatás**)
2. Mi alapján?
3. Kell az átlag (**összegzés**)
4. Kell a legjobb jegyátlag (**maximumkiválasztás**)
5. Kettő átlaga fölé eső jegyátlagokhoz tartozó diákok kellenek

Sorrend:

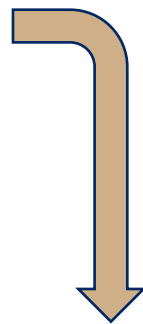
1. Összegzés (átlag)
2. Maximumérték
3. Kiválogatás

Példa versenyválogatás

Feladat:

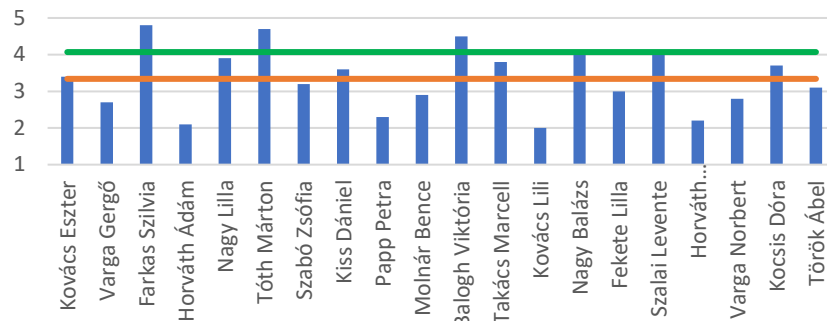
Egy iskolában a tanárnő azokat a gyereket nevezi be a versenyre, akiknek tanulmányi eredménye a legjobb tanuló átlaga és az osztályátlag közötti tartomány felső felébe esik. Kik ők?

Név	Jegyátlag
Kovács Eszter	3,4
Varga Gergő	2,7
Farkas Szilvia	4,8
Horváth Ádám	2,1
Nagy Lilla	3,9
Tóth Márton	4,7
Szabó Zsófia	3,2
Kiss Dániel	3,6
Papp Petra	2,3
Molnár Bence	2,9
Balogh Viktória	4,5
Takács Marcell	3,8
Kovács Lili	2
Nagy Balázs	4,1
Fekete Lilla	3
Szalai Levente	4
Horváth Boglárka	2,2
Varga Norbert	2,8
Kocsis Dóra	3,7
Török Ábel	3,1



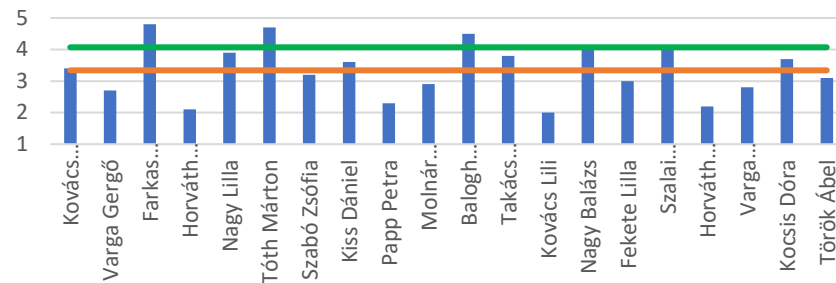
1. összeg= 66,8
2. átlag= $\text{összeg}/20 = 3,34$
3. max= 4,8
4. eltérés= $\text{max}-\text{átlag}=1,46$
5. határ= $\text{átlag}+\text{max}/2=4,07$
6. kiválogatás

Versenyre mehetnek



Példa versenyválogatás

Versenyre mehetnek



Feladat:

A legjobb tanuló átlaga és az osztályátlag közötti tartomány felső felébe eső jegyekhez tartozó diákok.

Specifikáció:

Be: $n \in \mathbb{N}$, $\text{diákok} \in \text{Diák}[1..n]$, $\text{Diák} = (\text{név}:S \times \text{jegy}:R)$

Sa: $\text{összeg} \in R$, $\text{átlag} \in R$, $\text{maxjegy} \in R$, $\text{eltérés} \in R$, $\text{határ} \in R$

Ki: $db \in \mathbb{N}$, $\text{versenyzők} \in S[1..db]$

Ef: $\forall i \in [1..n]: (1 \leq \text{diákok}[i].\text{jegy} \leq 5)$

Uf: $\text{összeg} = \text{SZUMMA}(i=1..n, \text{diákok}[i].\text{jegy})$ és

$\text{átlag} = \text{összeg} / n$ és

$(, \text{maxjegy}) = \text{MAX}(i=1..n, \text{diákok}[i].\text{jegy})$ és

$\text{eltérés} = \text{maxjegy} - \text{átlag}$ és

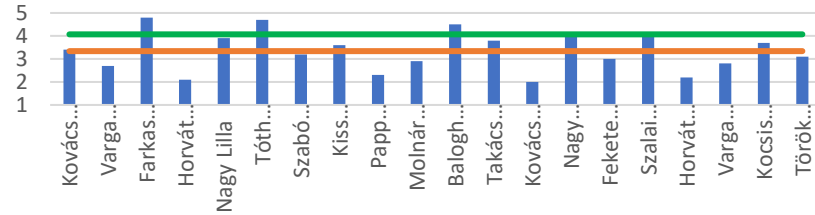
$\text{határ} = \text{átlag} + \text{eltérés} / 2$ és

$(db, \text{versenyzők}) = \text{KIVÁLOGAT}(i=1..n, \text{diákok}[i].\text{jegy} \geq \text{határ}, \text{diákok}[i].\text{név})$

1. összeg= 66,8
2. átlag= össze
3. max= 4,8
4. eltérés= max-a
5. határ= átlag-
6. kiválogatás

Példa versenyválogatás

Versenyre mehetnek



Sablonok:

- Uf: $s = \text{SZUMMA}(i = e..u, f(i))$
- Uf: $(\text{maxind}, \text{maxért}) = \text{MAX}(i = e..u, f(i))$
- Uf: $(\text{db}, y) = \text{KIVÁLOGAT}(i = e..u, T(i), f(i))$

Specifikáció:

Uf: ■ $\text{összeg} = \text{SZUMMA}(i = 1..n, \text{diákok}[i].\text{jegy})$ és
 $\text{átlag} = \text{összeg} / n$ és
 ■ $(, \text{maxjegy}) = \text{MAX}(i = 1..n, \text{diákok}[i].\text{jegy})$ és
 $\text{eltérés} = \text{maxjegy} - \text{átlag}$ és
 $\text{határ} = \text{átlag} + \text{eltérés} / 2$ és
 ■ $(\text{db}, \text{versenyzők}) = \text{KIVÁLOGAT}(i = 1..n,$
 $\text{diákok}[i].\text{jegy} \geq \text{határ}, \text{diákok}[i].\text{név})$

s ~ összeg
e..u ~ 1..n
f(i) ~ diákok[i].jegy

maxért ~ maxjegy
e..u ~ 1..n
f(i) ~ diákok[i].jegy

y ~ versenyzők
e..u ~ 1..n
T(i) ~ diákok[i.jegy >= határ
f(i) ~ diákok[i].név

Példa versenyválogatás

HF: programtranszformációkkal egyszerűbbé tenni!

Sablonok és feladat:

$s := 0$

$i = e .. u$

$s := s + f(i)$

Változó
 i : Egész

$ért := f(e); \text{maxind} := e$

$i = e + 1 .. u$

$f(i) > \text{maxért}$

$\text{maxért} := f(i)$

$\text{maxind} := i$

$db := 0$

$i = e .. u$

$T(i)$

$db := db + 1$

$y[db] := f(i)$

Változó
 i : Egész,
 összeg : Valós,
 átlag : Valós,
 maxjegy : Valós,
 eltérés : Valós,
 határ : Valós

Változó
 i : Egész

$\text{összeg} := 0$

$i = 1 .. n$

$\text{összeg} := \text{összeg} + \text{diákok}[i].\text{jegy}$

$\text{átlag} := \text{összeg} / n$

$\text{maxjegy} := \text{diákok}[1].\text{jegy}; \text{maxind} := 1$

$i = 1 .. n$

$\text{diákok}[i].\text{jegy} > \text{maxjegy}$

$\text{maxjegy} := \text{diákok}[i].\text{jegy}$

$\text{maxind} := i$

$\text{eltérés} := \text{maxjegy} - \text{átlag}$

$\text{határ} := \text{átlag} - \text{eltérés} / 2$

$db := 0$

$i = 1 .. n$

$\text{diákok}[i].\text{jegy} \geq \text{határ}$

$db := db + 1$

$\text{versenyzők}[db] := \text{diákok}[i].\text{név}$

$s \sim \text{összeg}$

$e .. u \sim 1 .. n$

$f(i) \sim \text{diákok}[i].\text{jegy}$

$\text{maxért} \sim \text{maxjegy}$

$e .. u \sim 1 .. n$

$f(i) \sim \text{diákok}[i].\text{jegy}$

$y \sim \text{versenyzők}$

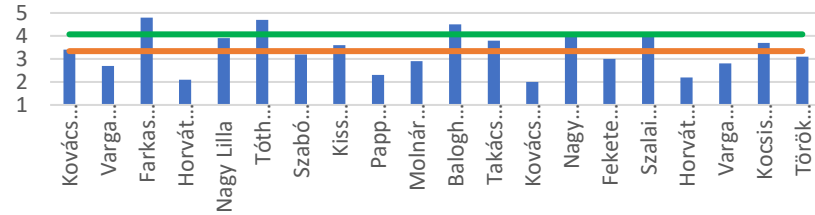
$e .. u \sim 1 .. n$

$T(i) \sim \text{diákok}[i].\text{jegy} \geq \text{határ}$

$f(i) \sim \text{diákok}[i].\text{név}$

Példa versenyválogatás

Versenyre mehetnek



Tanulság:

- specifikációban egymás után (**és**)
- algoritmusban is egymás után (**szekvencia**)
- **segédadatok** közvetítenek

Specifikáció:

Be: $n \in \mathbb{N}$, $\text{diákok} \in \text{Diák}[1..n]$, $\text{Diák} = (\text{név}: S \times \text{jegy}: R)$
 Sa: $\text{összeg} \in R$, $\text{átlag} \in R$, $\text{maxjegy} \in R$, $\text{eltérés} \in R$, $\text{határ} \in R$
 Ki: $\text{db} \in \mathbb{N}$, $\text{versenyzők} \in S[1..\text{db}]$
 Ef: $\forall i \in [1..n]: (1 \leq \text{diákok}[i].\text{jegy} \leq 5)$
 Uf: $\text{összeg} = \text{SZUMMA}(i=1..n, \text{diákok}[i].\text{jegy})$ és
 $\text{átlag} = \text{összeg}/n$ és
 $(, \text{maxjegy}) = \text{MAX}(i=1..n, \text{diákok}[i].\text{jegy})$ és
 $\text{eltérés} = \text{maxjegy} - \text{átlag}$ és
 $\text{határ} = \text{átlag} + \text{eltérés}/2$ és
 $(\text{db}, \text{versenyzők}) = \text{KIVÁLOGAT}(i=1..n,$
 $\text{diákok}[i].\text{jegy} \geq \text{határ}, \text{diákok}[i].\text{név})$

összeg:=0		
i=1..n		
összeg:=összeg+diákok[i].jegy		
átlag:=összeg/n		
maxjegy:=diákok[1].jegy; maxind:=1		
i=1..n		
T	diákok[i].jegy>maxjegy	F
	maxjegy:=diákok[i].jegy	-
	maxind:=i	
eltérés:=maxjegy-átlag		
határ:=átlag-eltérés/2		
db:=0		
i=1..n		
T	diákok[i].jegy≥határ	F
	db:=db+1	-
	versenyzők[db]:=diákok[i].név	



Példa

versenyválogatás

```
struct Diak {  
    public string nev;  
    public double jegy;  
}  
static void Main(string[] args) {  
    // deklaráció  
    Diak[] diakok;  
    List<string> versenyzok = new List<string>();  
  
    beolvas(out diakok);  
    versenyzok_meghatározasa(diakok, versenyzok);  
    kiiras(versenyzok);  
}  
static void beolvas(out Diak[] diakok) {  
  
}  
static void versenyzok_meghatározasa(Diak[] diakok, List<string> versenyzok) {  
  
}  
static void kiiras(List<string> versenyzok) {  
  
}
```

„Procedúrákra” bontás

Procedúra: visszatérési érték nélküli, void-os függvény

Példa

versenyválogatás

„Procedúrákra” bontás

```
struct Diak {
    public string nev;
    public double jegy;
}
static void Main(string[] args) {
    // deklaráció
    Diak[] diakok;
    List<string> versenyzok = new List<string>();

    beolvas(out diakok);
    versenyzok_meghatározasa(diakok, versenyzok);
    kiiras(versenyzok);
}
static void beolvas(out Diak[] diakok) {
    int n;
    Console.Write("n = ");
    int.TryParse(Console.ReadLine(), out n);
    diakok = new Diak[n];
    for (int i = 1; i <= n; i++) {
        Console.Write("{0}. diak neve = ", i);
        diakok[i - 1].nev = Console.ReadLine();
        Console.Write("{0}. diak jegye = ", i);
        double.TryParse(Console.ReadLine(), out diakok[i - 1].jegy);
    }
}
```

```

static void versenyzok_meghatarozasa(Diak[] diakok, List<string> versenyzok) {
    int n = diakok.Length;
    double osszeg = 0;
    for (int i = 1; i <= n; i++) {
        osszeg = osszeg + diakok[i - 1].jegy;
    }
    double atlag = osszeg / n;
    double maxjegy = diakok[1 - 1].jegy;
    int maxind = 1;
    for (int i = 2; i <= n; i++) {
        if (diakok[i - 1].jegy > maxjegy) {
            maxjegy = diakok[i - 1].jegy;
            maxind = i;
        }
    }
    double elteres = maxjegy - atlag;
    double hatar = atlag + elteres / 2;
    versenyzok.Clear();
    for (int i = 1; i <= n; i++) {
        if (diakok[i - 1].jegy >= hatar) {
            versenyzok.Add(diakok[i - 1].nev);
        }
    }
}

static void kiiras(List<string> versenyzok) {
    Console.WriteLine("{0} db diák mehet a versenyre:", versenyzok.Count);
    for (int i = 1; i <= versenyzok.Count; i++) {
        Console.WriteLine(versenyzok[i - 1]);
    }
}
}

```

„Procedúrákra” bontás

```

static void versenyzok_meghatarozasa(Diak[] diakok, List<string> versenyzok) {
    int n = diakok.Length;

    double osszeg;
    osszegzes(diakok, out osszeg);

    double atlag = osszeg / n;
    double maxjegy;
    maxkiv(diakok, out maxjegy);

    double elteres = maxjegy - atlag;
    double hatar = atlag + elteres / 2;
    kivalogat(diakok, hatar, versenyzok);
}
static void osszegzes(Diak[] diakok, out double osszeg) {
    int n = diakok.Length;
    osszeg = 0;
    for (int i = 1; i <= n; i++) {
        osszeg = osszeg + diakok[i - 1].jegy;
    }
}

```

A megoldás
részprocedúrákra bontása

```

static void maxkiv(Diak[] diakok, out double maxjegy) {
    int n = diakok.Length;
    maxjegy = diakok[1 - 1].jegy;
    int maxind = 1;
    for (int i = 2; i <= n; i++) {
        if (diakok[i - 1].jegy > maxjegy) {
            maxjegy = diakok[i - 1].jegy;
            maxind = i;
        }
    }
}
static void kivalogat(Diak[] diakok, double hatar,
    List<string> versenyzok) {
    int n = diakok.Length;
    versenyzok.Clear();
    for (int i = 1; i <= n; i++) {
        if (diakok[i - 1].jegy >= hatar) {
            versenyzok.Add(diakok[i - 1].nev);
        }
    }
}

```



Példa

versenyválogatás

Függvényekre bontás

```
struct Diak {  
    public string nev;  
    public double jegy;  
}  
  
static void Main(string[] args) {  
    // deklaráció  
    Diak[] diakok;  
    List<string> versenyzok = new List<string>();  
  
    diakok = beolvas();  
    versenyzok = versenyzok_meghatározasa(diakok);  
    kiiras(versenyzok);  
}  
  
static Diak[] beolvas() {  
    // ...  
    Diak[] diakok = new Diak[n];  
    // ...  
    return diakok;  
}  
  
static List<string> versenyzok_meghatározasa(Diak[] diakok) {  
    List<string> versenyzok = new List<string>();  
    // ...  
    return versenyzok;  
}
```



Példa

versenyválogatás

Függvényekre bontás

```
static void Main(string[] args) {  
    // deklaráció  
    Diak[] diakok;  
    List<string> versenyzok = new List<string>();  
  
    diakok = beolvas();  
    versenyzok = versenyzok_meghatározasa(diakok);  
    kiiras(versenyzok);  
}  
static Diak[] beolvas() {  
    int n;  
    Console.Write("n = ");  
    int.TryParse(Console.ReadLine(), out n);  
    Diak[] diakok = new Diak[n];  
    for (int i = 1; i <= n; i++) {  
        Console.Write("{0}. diak neve = ", i);  
        diakok[i - 1].nev = Console.ReadLine();  
        Console.Write("{0}. diak jegye = ", i);  
        double.TryParse(Console.ReadLine(), out diakok[i - 1].jegy);  
    }  
    return diakok;  
}
```



```
static List<string> versenyzok_meghatározasa(Diak[] diakok) {  
    List<string> versenyzok = new List<string>();
```

```
    double osszeg = osszegzes(diakok);
```

```
    int n = diakok.Length;  
    double atlag = osszeg / n;  
    double maxjegy = maxkiv(diakok);  
    double elteres = maxjegy - atlag;  
    double hatar = atlag + elteres / 2;  
    versenyzok = kivalogat(diakok, hatar);  
    return versenyzok;  
}
```

```
static double osszegzes(Diak[] diakok) {  
    int n = diakok.Length;  
    double osszeg = 0;  
    for (int i = 1; i <= n; i++) {  
        osszeg = osszeg + diakok[i - 1].jegy;  
    }  
    return osszeg;  
}
```

Megoldás részfüggvényekre bontása

```
static double maxkiv(Diak[] diakok) {  
    int n = diakok.Length;  
    double maxjegy = diakok[1 - 1].jegy;  
    int maxind = 1;  
    for (int i = 2; i <= n; i++) {  
        if (diakok[i - 1].jegy > maxjegy) {  
            maxjegy = diakok[i - 1].jegy;  
            maxind = i;  
        }  
    }  
    return maxjegy;  
}  
  
private static List<string> kivalogat(  
    Diak[] diakok, double hatar) {  
    int n = diakok.Length;  
    List<string> versenyzok =  
        new List<string>();  
  
    versenyzok.Clear();  
    for (int i = 1; i <= n; i++) {  
        if (diakok[i - 1].jegy >= hatar) {  
            versenyzok.Add(diakok[i - 1].nev);  
        }  
    }  
    return versenyzok;  
}
```



Több programozási minta
együttes használata
egymásba ágyazva



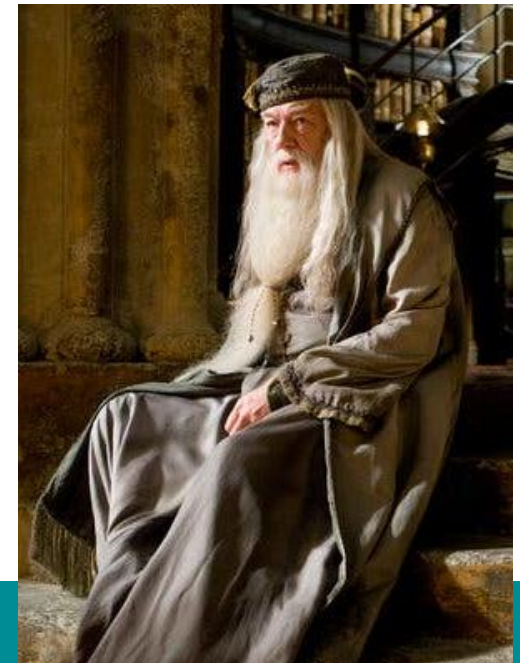
Példa

kitűnő tanulók száma



Feladat:

A Roxfortban év végén n varázslótanoncról ismerjük az m tárgyból szerzett jegyét egy táblázatban. Dumbledore szeretné a kitűnő tanulókat megajándékozni Bagoly Berti-féle Mindenízű Drazsével. Hány zacskóval kell vennie?



Példa

kitűnő tanulók száma



Feladat:

Egy egész számokat tartalmazó mátrixban **hány** olyan sor van, ami **csak 5-öst** tartalmaz?

Példa:

	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

→ db=2

Ötlet:

- hány darab „valamilyen” sor van?
- a kérdésre a válasz egy darabszám
- → megszámlálás minta alapján megoldható
- Kívülről befelé vagy fentről lefele haladunk

Példa

kitűnő tanulók száma



	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

Feladat:

Egy egész számokat tartalmazó mátrixban **hány** olyan sor van, ami **csak 5-öst** tartalmaz?

Specifikáció:

Be: $n \in \mathbb{N}$, $m \in \mathbb{N}$, $\text{jegyek} \in \mathbb{N}[1..n, 1..m]$

Ki: $db \in \mathbb{N}$

Ef: $\forall \text{sor} \in [1..n] : (\forall \text{oszlop} \in [1..m] : (1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

Uf: **db=DARAB(sor=1..n, csakötös(sor))**

Egyelőre elrejtettük egy függvény mögé azt a részfeladatot, hogy mi alapján számolunk egy sort.
Következőnek ezt kell átgondolnunk!

Példa

kitűnő tanulók száma



	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

Feladat:

Egy egész számokat tartalmazó mátrixban **hány** olyan **sor** van, **ami csak 5-öst tartalmaz?**

Tekintsük **egy sor** feldolgozását külön részfeladatnak

Sor, ami csak 5-öst tartalmaz

- fókuszáljunk egy sorra!
- minden elem 5-ös? → igen/nem
- ez egy mind eldöntés

Egy **sor** csak ötös (**csakötös**)?:

Be: $n \in \mathbb{N}$, $m \in \mathbb{N}$, jegyek $\in [1..n, 1..m]$, **sor** $\in \mathbb{N}$

Ki: **csakötös(sor)** $\in \mathbb{L}$

Ef: $\forall \text{ sor} \in [1..n] : (\forall \text{ oszlop} \in [1..m] : (1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

Uf: **csakötös(sor)** = **MIND(oszlop=1..m, jegyek[sor, oszlop]=5)**

Példa

kitűnő tanulók száma



	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

Feladat:

Egy egész számokat tartalmazó mátrixban **hány** olyan **sor** van, **ami csak 5-öst tartalmaz?**

Tekintsük **egy sor** feldolgozását külön részfeladatnak

Sor, ami csak 5-öst tartalmaz

- fókuszáljunk egy sorra!
- minden elem 5-ös? → igen/nem
- ez egy mind eldöntés

Egy **sor** csak ötös (**csakötös**)?:

Be: $n \in \mathbb{N}$, $m \in \mathbb{N}$, $\text{jegyek} \in \mathbb{N}[1..n, 1..m]$, **sor** $\in \mathbb{N}$

Ki: $\text{mind} \in \mathbb{L}$

Ef: $\forall \text{sor} \in [1..n] : (\forall \text{oszlop} \in [1..m] : (1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

Uf: $(\text{mind} = \text{csakötös}(\text{sor})) =$

$(\text{mind} = \text{MIND}(\text{oszlop} = 1..m, \text{jegyek}[\text{sor}, \text{oszlop}] = 5))$

Példa

kitűnő tanulók száma



	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

Feladat:

Egy egész számokat tartalmazó mátrixban **hány** olyan sor van, ami **csak 5-öst** tartalmaz?

Specifikáció:

Be: $n \in \mathbb{N}$, $m \in \mathbb{N}$,

jegyek $\in \mathbb{N}[1..n, 1..m]$

Ki: $db \in \mathbb{N}$

Fv: csakötös: $\mathbb{N} \rightarrow \mathbb{L}$,

csakötös(sor) = mind $\in \mathbb{L} : (\text{mind} = \text{MIND}(\text{oszlop} = 1..m, \text{jegyek}[\text{sor}, \text{oszlop}] = 5))$

Ef: $\forall \text{sor} \in [1..n] : (\forall \text{oszlop} \in [1..m] : (1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

Uf: $db = \text{DARAB}(\text{sor} = 1..n, \text{csakötös}(\text{sor}))$

olyan mind, amelyre igaz a : utáni rész

Példa

kitűnő tanulók száma



	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

Megszámolás sablon:

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

Ki: $db \in \mathbb{N}$

Ef: -

Uf: $db = \text{DARAB}(i = e..u, T(i))$

Mind eldöntés sablon:

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

Ki: $mind \in L$

Ef: -

Uf: $mind = \text{MIND}(i = e..u, T(i))$

Kitűnő tanulók száma

Be: $n \in \mathbb{N}, m \in \mathbb{N}, \text{jegyek} \in \mathbb{N}[1..n, 1..m]$

Ki: $db \in \mathbb{N}$

Fv: $\text{csakötös} : \mathbb{N} \rightarrow L$,

$\text{csakötös}(\text{sor}) = \text{mind} \in L : ($

$\text{mind} = \text{MIND}(\text{oszlop} = 1..m,$
 $\text{jegyek}[\text{sor}, \text{oszlop}] = 5))$

Ef: $\forall \text{sor} \in [1..n] : (\forall \text{oszlop} \in [1..m] :$

$(1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

Uf: $db = \text{DARAB}(\text{sor} = 1..n, \text{csakötös}(\text{sor}))$

Megszámolás

$i \sim \text{sor}$

$e..u \sim 1..n$

$T(i) \sim \text{csakötös}(\text{sor})$

Mind eldöntés

$i \sim \text{oszlop}$

$e..u \sim 1..m$

$T(i) \sim \text{jegyek}[\text{sor}, \text{oszlop}] = 5$

Példa

kitűnő tanulók száma



	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

Kitűnő tanulók száma

Be: $n \in \mathbb{N}$, $m \in \mathbb{N}$, jegyek $\in \mathbb{N}[1..n, 1..m]$

Ki: $db \in \mathbb{N}$

Fv: csakötös: $\mathbb{N} \rightarrow \mathbb{L}$,

$csakötös(sor) = \text{mind} \in \mathbb{L} : (\text{mind} = \text{MIND}(\text{oszlop} = 1..m, \text{jegyek}[sor, \text{oszlop}] = 5))$

Ef: $\forall sor \in [1..n] : (\forall oszlop \in [1..m] : (1 \leq \text{jegyek}[sor, \text{oszlop}] \leq 5))$

Uf: $db = \text{DARAB}(sor = 1..n, csakötös(sor))$

A továbbiakban eltekintünk ettől a redundáns megadástól. Feltételezzük, hogy a sablon kimenetét adja meg! A visszavezetésben így nem jelenik meg.

Megszámolás

$i \sim sor$

$e..u \sim 1..n$

$T(i) \sim csakötös(sor)$

Mind eldöntés

$i \sim oszlop$

$e..u \sim 1..m$

$T(i) \sim \text{jegyek}[sor, \text{oszlop}] = 5$

Példa

kitűnő tanulók száma



	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

Kitűnő tanulók száma

Be: $n \in \mathbb{N}$, $m \in \mathbb{N}$, jegyek $\in \mathbb{N}[1..n, 1..m]$

Ki: $db \in \mathbb{N}$

Fv: csakötös: $\mathbb{N} \rightarrow \mathbb{L}$,

$csakötös(sor) = \text{MIND}(\text{oszlop} = 1..m, \text{jegyek}[sor, \text{oszlop}] = 5)$

Ef: $\forall sor \in [1..n] : (\forall oszlop \in [1..m] : (1 \leq \text{jegyek}[sor, oszlop] \leq 5))$

Uf: $db = \text{DARAB}(sor = 1..n, csakötös(sor))$

Megszámolás

$i \sim sor$

$e..u \sim 1..n$

$T(i) \sim csakötös(sor)$

Mind eldöntés

$i \sim oszlop$

$e..u \sim 1..m$

$T(i) \sim \text{jegyek}[sor, oszlop] = 5$

Megszámolásban **mind eldöntés**

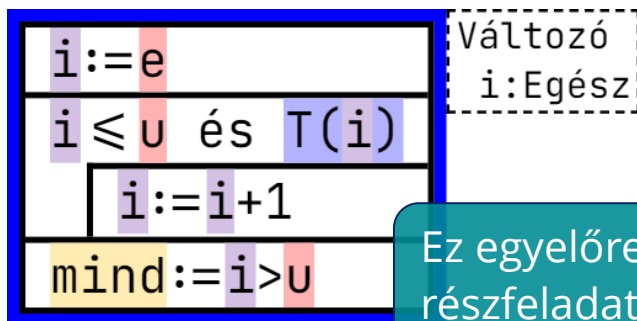
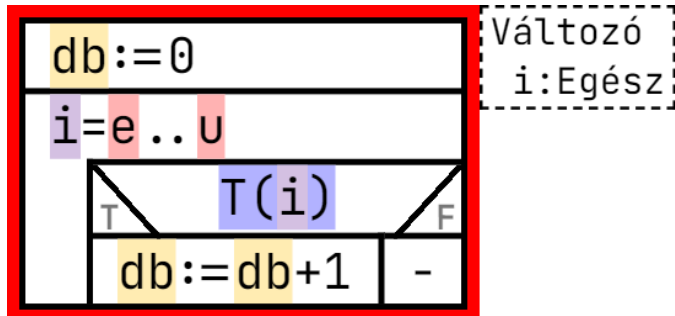
Példa

kitűnő tanulók száma



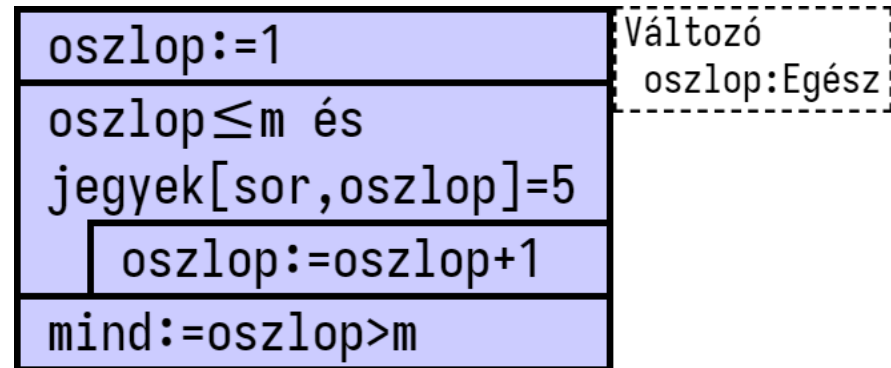
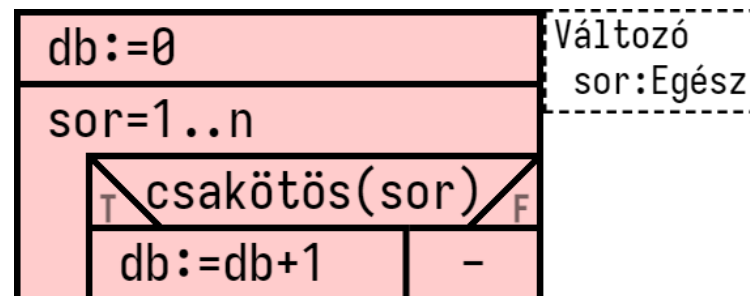
	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

Sablonok



Ez egyelőre két külön részfeladat algoritmusa

Kitűnő tanulók száma



Megszámolás

i ~ sor
e..u ~ 1..n
T(i) ~ csakötös(sor)

Mind eldöntés

i ~ oszlop
e..u ~ 1..m
T(i) ~ jegyek[sor,oszlop]=5

Példa

kitűnő tanulók száma



	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

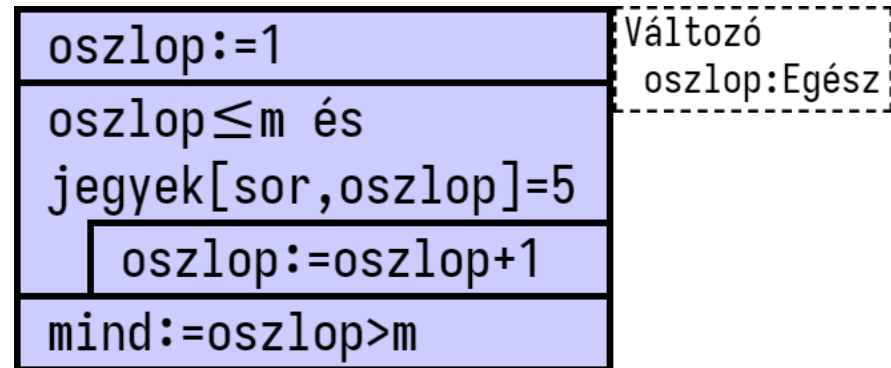
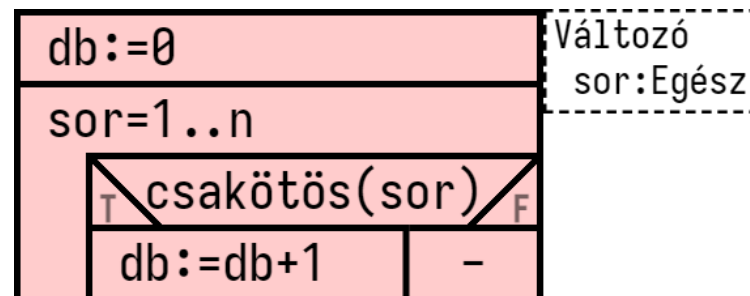
Algoritmusok összekapcsolása:

1. Függvényhívással

Azaz a beágyazott minta algoritmus egy függvény lesz, amit a külső minta hív meg.

2. Behelyettesítéssel

Azaz az alprogramot a hívás helyére másoljuk



Példa

kitűnő tanulók száma

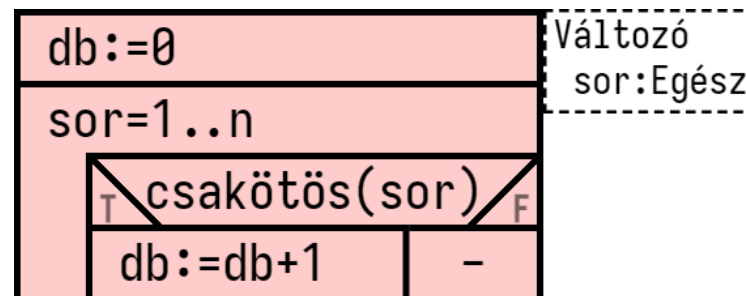


	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

Algoritmusok összekapcsolása:

1. Függvényhívással

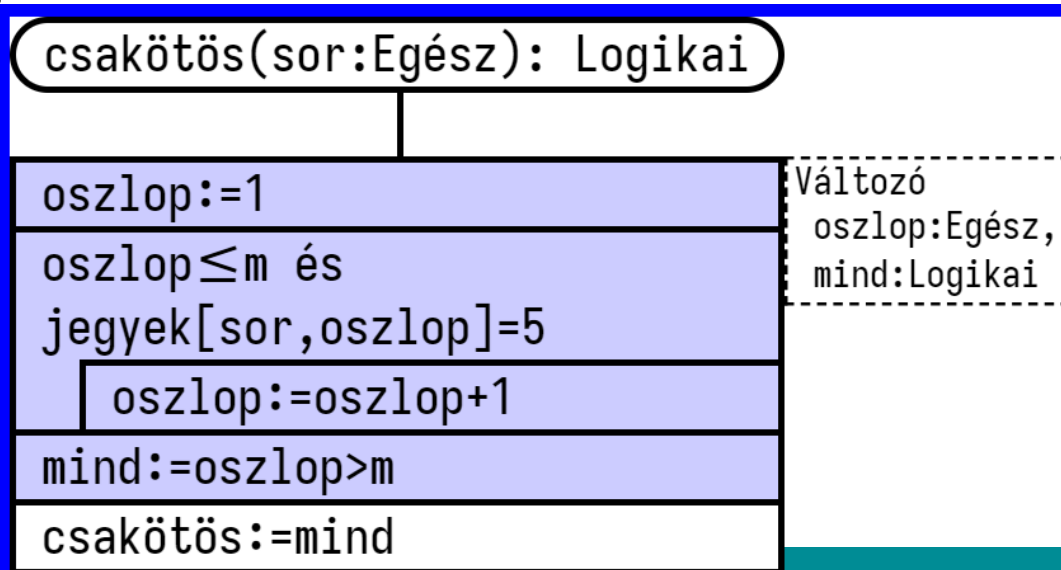
Azaz a beágyazott minta algoritmus egy függvény lesz, amit a külső minta hív meg.



2. Behelyettesítéssel

Azaz az alprogramot a hívás helyére másoljuk.

Ezt az utat követjük általában.
Ez az elvárás!



Példa

kitűnő tanulók száma



	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

Algoritmusok összekapcsolása:

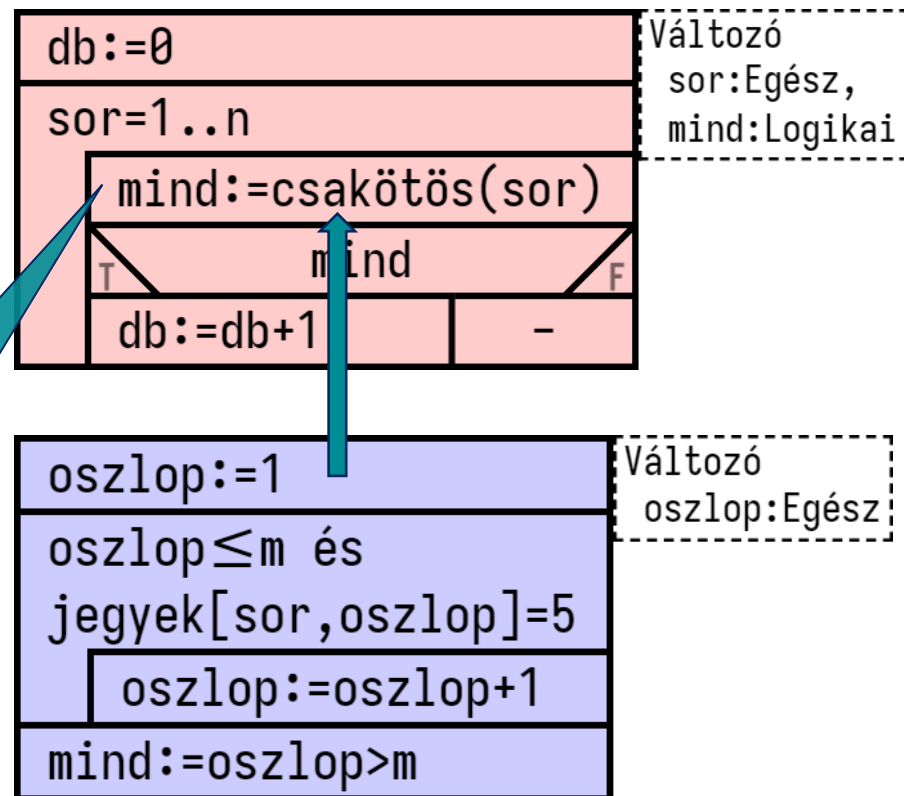
1. Függvényhívással

Azaz a beágyazott minta algoritmus egy függvény lesz, amit a külső minta hív meg.

2. Behelyettesítéssel

Azaz az alprogramot a hívás helyére másoljuk

1. lépés:
segédváltozó bevezetése



Példa

kitűnő tanulók száma



	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

Algoritmusok összekapcsolása:

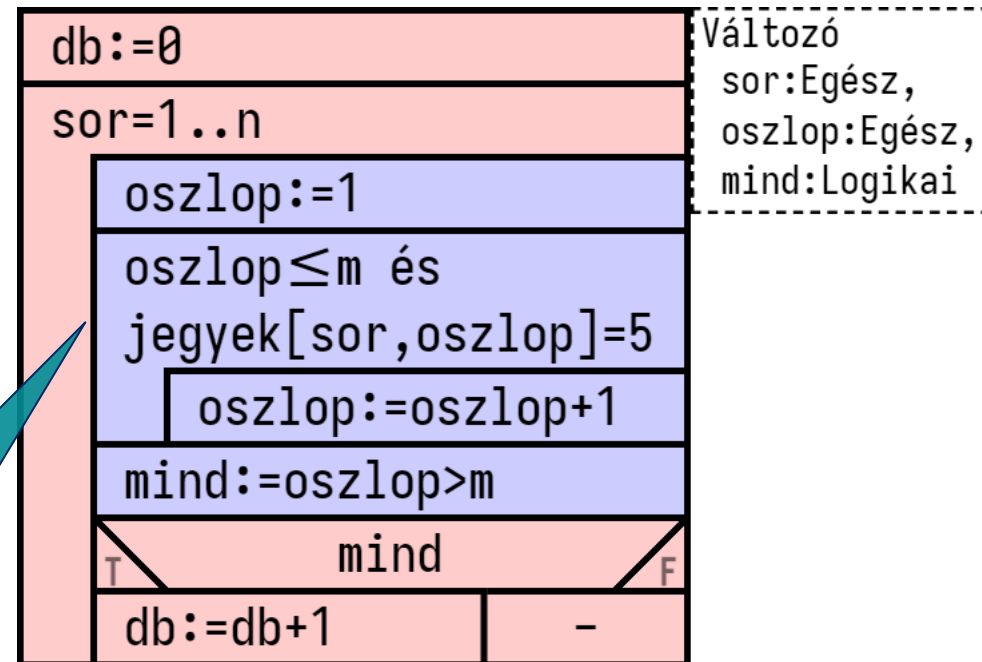
1. Függvényhívással

Azaz a beágyazott minta algoritmus egy függvény lesz, amit a külső minta hív meg.

2. Behelyettesítéssel

Azaz az alprogramot a hívás helyére másoljuk

2. lépés:
behelyettesítés



Példa

kitűnő tanulók száma

Megszámolásban **mind** eldöntés

Kitűnő tanulók száma

Be: $n \in \mathbb{N}$, $m \in \mathbb{N}$, jegyek $\in \mathbb{N}[1..n, 1..m]$

Ki: $db \in \mathbb{N}$

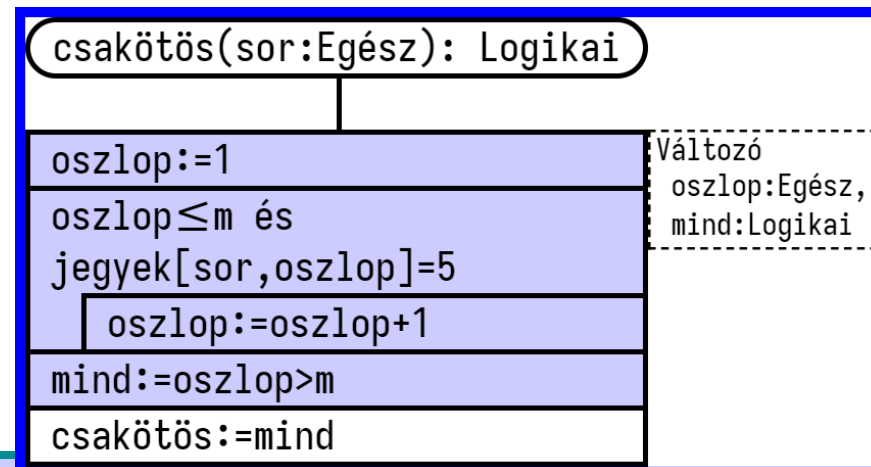
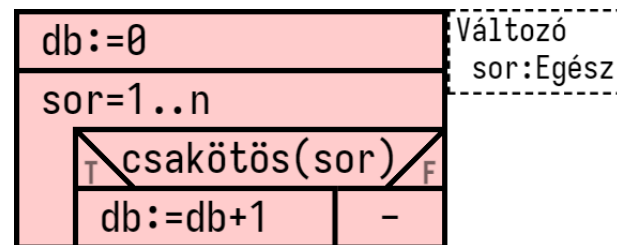
Fv: csakötös: $\mathbb{N} \rightarrow \mathbb{L}$,

$csakötös(sor) = \text{MIND}(\text{oszlop} = 1..m,$
 $jegyek[sor, \text{oszlop}] = 5)$

Ef: $\forall sor \in [1..n]: (\forall \text{oszlop} \in [1..m]:$
 $(1 \leq jegyek[sor, \text{oszlop}] \leq 5))$

Uf: $db = \text{DARAB}(sor = 1..n, csakötös(sor))$

	1	2	3	m=4
1	5	5	4	5
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5



Megszámolás

$i \sim sor$
 $e..u \sim 1..n$
 $T(i) \sim csakötös(sor)$

Mind eldöntés

$i \sim oszlop$
 $e..u \sim 1..m$
 $T(i) \sim jegyek[sor, oszlop]=5$

Példa

kitűnő tanulók száma

	1	2	3	m=4
1	5	5	4	5
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
5	5	3	4	5

Megjegyzés: az alkalmazott minták összevonhatók

Be: $n \in \mathbb{N}$, $m \in \mathbb{N}$, $\text{jegyek} \in \mathbb{N}[1..n, 1..m]$

Ki: $db \in \mathbb{N}$

Fv: $\text{csakötös} : \mathbb{N} \rightarrow \mathbb{L}$,

$\text{csakötös}(\text{sor}) = \text{MIND}(\text{oszlop} = 1..m, \text{jegyek}[\text{sor}, \text{oszlop}] = 5)$

Ef: $\forall \text{sor} \in [1..n] : (\forall \text{oszlop} \in [1..m] : (1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

Uf: $db = \text{DARAB}(\text{sor} = 1..n, \text{csakötös}(\text{sor}))$



Be: $n \in \mathbb{N}$, $m \in \mathbb{N}$, $\text{jegyek} \in \mathbb{N}[1..n, 1..m]$

Ki: $db \in \mathbb{N}$

Ef: $\forall \text{sor} \in [1..n] : (\forall \text{oszlop} \in [1..m] : (1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

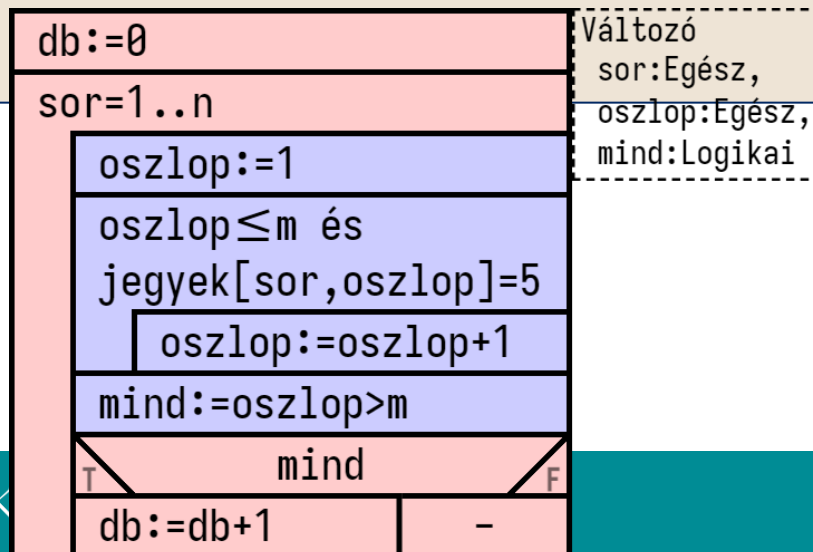
Uf: $db = \text{DARAB}(\text{sor} = 1..n, \text{MIND}(\text{oszlop} = 1..m, \text{jegyek}[\text{sor}, \text{oszlop}] = 5))$

Példa

kitűnő tanulók száma

```
static void Main(string[] args) {
    // deklarálás
    int n, m; int[,] jegyek;
    int db;
    // beolvasás
    Console.WriteLine("Varazstanoncok szama = ");
    int.TryParse(Console.ReadLine(), out n);
    Console.WriteLine("Jegyek szama = ");
    int.TryParse(Console.ReadLine(), out m);
    jegyek = new int[n, m];
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            Console.WriteLine("{0}. varazstanonc {1} jegye = ", i, j);
            int.TryParse(Console.ReadLine(), out jegyek[i - 1, j - 1]);
        }
    }
}
```

```
// feldolgozás
db = 0;
for (int sor = 1; sor <= n; sor++) {
    int oszlop = 1;
    while (oszlop <= m &&
        jegyek[sor - 1, oszlop - 1] == 5) {
        oszlop = oszlop + 1;
    }
    bool mind = oszlop > m;
    if (mind) {
        db = db + 1;
    }
}
// kiírás
Console.WriteLine("{0} db", db);
}
```



Változó
sor:Egész,
oszlop:Egész,
mind:Logikai

Függvények nélkül

Példa

kitűnő tanulók száma

„Procedúrákra” bontás

```
static void Main(string[] args) {  
    // deklarálás  
    int n; int m; int[,] jegyek;  
    int db;  
  
    beolvas(out n, out m, out jegyek);  
    kituno_tanulok_szama(n, m, jegyek, out db);  
    kiir(db);  
}  
static void beolvas(out int n, out int m, out int[,] jegyek) {  
    Console.Write("Varazstanoncok szama = ");  
    int.TryParse(Console.ReadLine(), out n);  
    Console.Write("Jegyek szama = ");  
    int.TryParse(Console.ReadLine(), out m);  
    jegyek = new int[n, m];  
    for (int i = 1; i <= n; i++) {  
        for (int j = 1; j <= m; j++) {  
            Console.Write("{0}. varazstanonc {1} jegye = ", i, j);  
            int.TryParse(Console.ReadLine(), out jegyek[i - 1, j - 1]);  
        }  
    }  
}
```

Példa

kitűnő tanulók száma

„Procedúrákra” bontás

```
static void kituno_tanulok_szama(int n, int m, int[,] jegyek, out int db) {
    db = 0;
    for (int sor = 1; sor <= n; sor++) {
        if (csakotos(sor, m, jegyek)) {
            db = db + 1;
        }
    }
}

static bool csakotos(int sor, int m, int[,] jegyek) {
    int oszlop = 1;
    while (oszlop <= m && jegyek[sor - 1, oszlop - 1] == 5) {
        oszlop = oszlop + 1;
    }
    bool mind = oszlop > m;
    return mind;
}

static void kiir(int db) {
    Console.WriteLine("{0} db zacsko cukrot kell venni.", db);
}
```

Példa

kitűnő tanulók száma

Implicit hossz

```
static void Main(string[] args) {  
    // deklarálás  
    int[,] jegyek;  
    int db;  
  
    beolvas(out jegyek);  
    kituno_tanulok_szama(jegyek, out db);  
    kiir(db);  
}  
static void beolvas(out int[,] jegyek) {  
    int n; int m;  
    // ...  
}  
static void kituno_tanulok_szama(int[,] jegyek, out int db) {  
    int n = jegyek.GetLength(0);  
    db = 0;  
    for (int sor = 1; sor <= n; sor++) {  
        if (csakotos(sor, jegyek)) {  
            db = db + 1;  
        }  
    }  
}  
  
static bool csakotos(int sor, int[,] jegyek) {  
    int m = jegyek.GetLength(1);  
    int oszlop = 1;  
    while (oszlop <= m &&  
           jegyek[sor - 1, oszlop - 1] == 5) {  
        oszlop = oszlop + 1;  
    }  
    bool mind = oszlop > m;  
    return mind;  
}
```



Példa

kitűnő tanulók száma

```
static void Main(string[] args) {  
    // deklarálás  
    int[,] jegyek;  
    int db;  
  
    jegyek = beolvas();  
    db = kituno_tanulok_szama(jegyek);  
    kiir(db);  
}  
static int[,] beolvas() {  
    int n; int m;  
    int[,] jegyek;  
  
    Console.WriteLine("Varazstanoncok szama");  
    int.TryParse(Console.ReadLine(), out n);  
    Console.WriteLine("Jegyek szama = ");  
    int.TryParse(Console.ReadLine(), out m);  
    jegyek = new int[n, m];  
    for (int i = 1; i <= n; i++) {  
        for (int j = 1; j <= m; j++) {  
            Console.WriteLine("{0}. varazstanonc {1} jegye = ", i, j);  
            int.TryParse(Console.ReadLine(), out jegyek[i - 1, j - 1]);  
        }  
    }  
    return jegyek;  
}
```

```
static int kituno_tanulok_szama(int[,] jegyek) {  
    int n = jegyek.GetLength(0);  
    int db = 0;  
    for (int sor = 1; sor <= n; sor++) {  
        if (csakotos(sor, jegyek)) {  
            db = db + 1;  
        }  
    }  
    return db;  
}  
static bool csakotos(int sor, int[,] jegyek) {  
    int m = jegyek.GetLength(1);  
    int oszlop = 1;  
    while (oszlop <= m &&  
           jegyek[sor - 1, oszlop - 1] == 5) {  
        oszlop = oszlop + 1;  
    }  
    bool mind = oszlop > m;  
    return mind;  
}
```

Függvényekre bontás

Példa

kitűnő tanulók száma

Függvényekre bontás

```
static void Main(string[] args) {  
    // deklarálás  
    int[,] jegyek;  
    int db;  
  
    kiir(kituno_tanulok_szama(beolvas()));  
}  
static int[,] beolvas() {  
    int n; int m;  
    int[,] jegyek;  
  
    Console.Write("Varazstanoncok szama  
int.TryParse(Console.ReadLine(), ou  
Console.Write("Jegyek szama = ");  
int.TryParse(Console.ReadLine(), ou  
jegyek = new int[n, m];  
    for (int i = 1; i <= n; i++) {  
        for (int j = 1; j <= m; j++) {  
            Console.Write("{0}. varazstanon  
int.TryParse(Console.ReadLine()  
        }  
    }  
    return jegyek;  
}
```

```
static int kituno_tanulok_szama(int[,] jegyek) {  
    int n = jegyek.GetLength(0);  
    int db = 0;  
    for (int sor = 1; sor <= n; sor++) {  
        if (csakotos(sor, jegyek)) {  
            db = db + 1;  
        }  
    }  
    return db;  
}  
static bool csakotos(int sor, int[,] jegyek) {  
    int m = jegyek.GetLength(1);  
    int oszlop = 1;  
    while (oszlop <= m &&  
        jegyek[sor - 1, oszlop - 1] == 5) {  
        oszlop = oszlop + 1;  
    }  
    bool mind = oszlop > m;  
    return mind;  
}
```

Példa

kitűnő tanulók száma



Tanulságok:

1. Mátrixban dimenzióként van egy-egy minta
2. Bontsuk fel részfeladatokra, pl. felülről lefelé
3. Algoritmusban függvényként valósítsuk meg

Kitűnő tanulók száma

Be: $n \in \mathbb{N}$, $m \in \mathbb{N}$, jegyek $\in [1..n, 1..m]$

Ki: $db \in \mathbb{N}$

Fv: csakötös: $\mathbb{N} \rightarrow \mathbb{L}$,

$csakötös(sor) = MIND(oszlop = 1..m,$
 $jegyek[sor, oszlop] = 5)$

Ef: $\forall sor \in [1..n] : (\forall oszlop \in [1..m] :$
 $(1 \leq jegyek[sor, oszlop] \leq 5))$

Uf: $db = DARAB(sor = 1..n, csakötös(sor))$

	1	2	3	m=4
1	5	5	4	3
2	5	5	5	5
3	5	3	2	4
4	5	5	5	5
n=5	5	3	4	5

Példa: sudoku

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?

Lehetséges ötletek:

- Alulról felfelé
 - Soronként számoljuk meg, hány ötös van, majd adjuk ezeket össze!
- Felülről lefelé
 - Adjuk össze soronként, hány ötös van egy sorban!

Példa: sudoku

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?


Lehetséges ötletek:

- **Alulról felfelé:** Soronként számoljuk meg, hány ötös van, majd adjuk ezeket össze!
- Minden sorban számoljuk meg az 5-ösöket, és adjuk ezeket a darabszámokat össze!
- Másolásban megszámlálás és összegzés



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

másolásban
megszámlálás



0
0
0
0
1
0
1
1
0

összegzés



db=3

Példa: sudoku

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

Feladat:

Hány 5-öst írtunk már be egy sudoku táblázatba?

Minden sorban számoljuk meg az 5-ösöket, és adjuk ezeket a darabszámokat össze!

Másolásban megszámolás és összegzés

Specifikáció:

Be: $s \in \mathbb{N}[1..9, 1..9]$

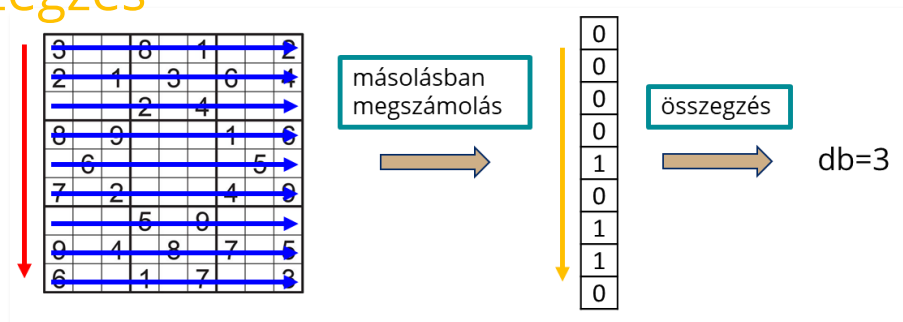
Sa: $dbk \in \mathbb{N}[1..9]$

Ki: $db \in \mathbb{N}$

Fv: $\text{hány5ös} : \mathbb{N} \rightarrow \mathbb{N}$, $\text{hány5ös}(i) = \text{DARAB}(j=1..9, s[i, j]=5)$

Ef: $\forall i \in [1..9] : (\forall j \in [1..9] : (0 \leq s[i, j] \leq 9))$

Uf: $dbk = \text{MÁSOL}(i=1..9, \text{hány5ös}(i))$ és
 $db = \text{SZUMMA}(i=1..9, dbk[i])$



Sorokon i-vel,
oszlopokon j-vel megyünk végig

Példa: sudoku

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

Másolás sablon:

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

Ki: $y \in H[1..u-e+1]$

Ef: -

Uf: $y = \text{MÁSOL}(i=e..u, f(i))$

Megszámolás sablon:

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

Ki: $db \in \mathbb{N}$

Ef: -

Uf: $db = \text{DARAB}(i=e..u, T(i))$

Összegzés sablon:

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

Ki: $s \in H$

Ef: -

Uf: $s = \text{SZUMMA}(i=e..u, f(i))$

Sudoku

Be: $s \in \mathbb{N}[1..9, 1..9]$

Sa: $dbk \in \mathbb{N}[1..9]$

Ki: $db \in \mathbb{N}$

Fv: $\text{hány5ös} : \mathbb{N} \rightarrow \mathbb{N},$

$\text{hány5ös}(i) = \text{DARAB}(j=1..9, s[i,j]=5)$

Ef: $\forall i \in [1..9] : (\forall j \in [1..9] : (0 \leq s[i,j] \leq 9))$

Uf: $dbk = \text{MÁSOL}(i=1..9, \text{hány5ös}(i))$ és

$db = \text{SZUMMA}(i=1..9, dbk[i])$

Másolás

$y \sim dbk$

$e..u \sim 1..9$

$f(i) \sim \text{hány5ös}(i)$

Megszámolás

$i \sim j$

$e..u \sim 1..9$

$T(i) \sim s[i,j]=5$

Összegzés

$s \sim db$

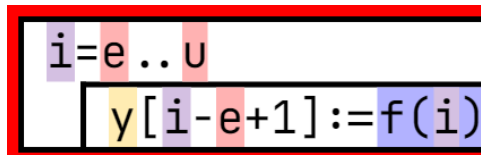
$e..u \sim 1..9$

$T(i) \sim dbk[i]$

Példa: sudoku

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

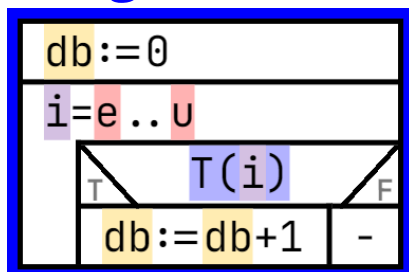
Másolás:



Másolás

$y \sim dbk$
 $e..u \sim 1..9$
 $f(i) \sim hány5ös(i)$

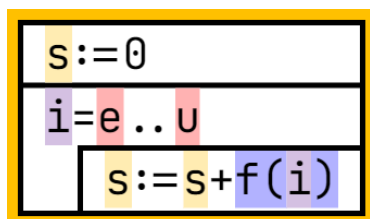
Megszámolás:



Megszámolás

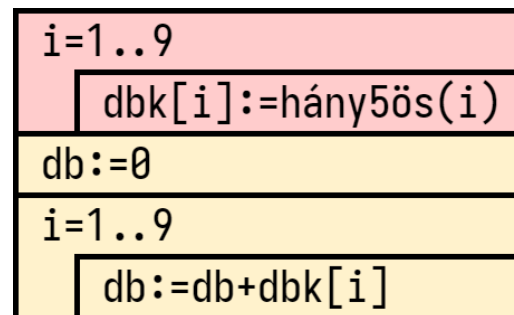
$i \sim j$
 $e..u \sim 1..9$
 $T(i) \sim s[i,j]=5$

Összegzés:



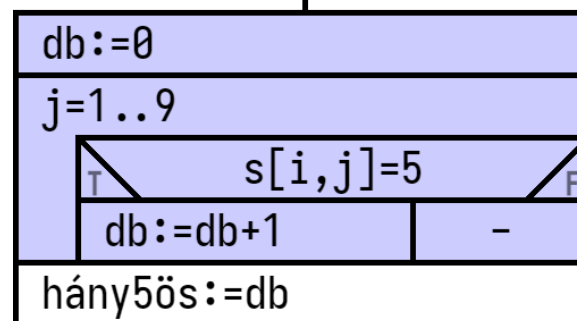
Összegzés

$s \sim db$
 $e..u \sim 1..9$
 $T(i) \sim dbk[i]$



Változó
 i : Egész,
 dbk : Tömb[1..9:Egész]

hány5ös(i :Egész): Egész



Változó
 j : Egész,
 db : Egész

hány5ös(i) = $db \in \mathbb{N} : (db = DARAB(j=1..9, s[i,j]=5))$

→ db lokális változó

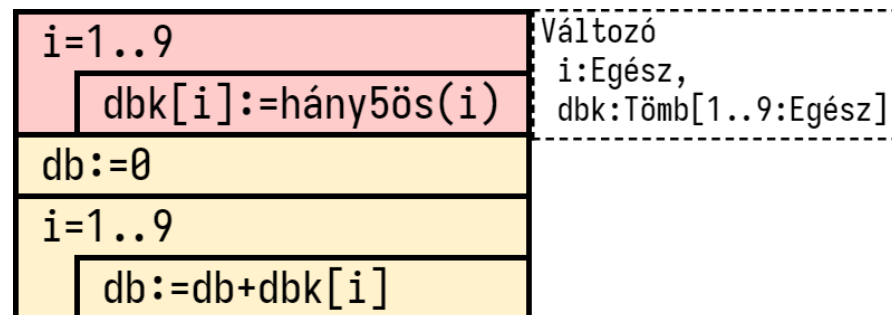
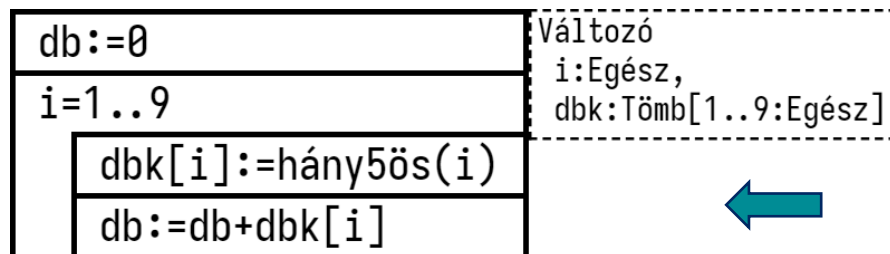
Példa: sudoku

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

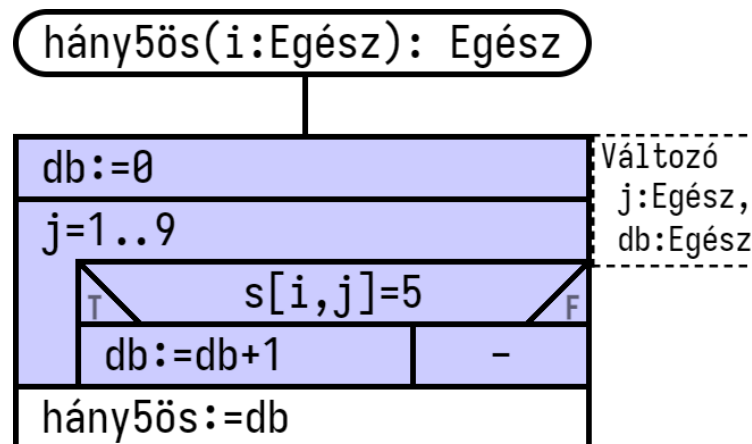
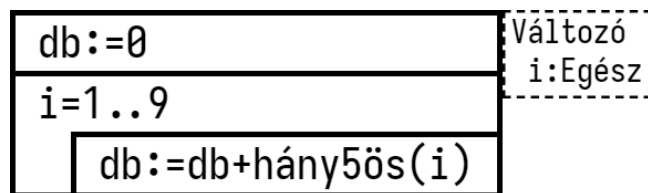
Ld. később

Programtranszformációk:

1. Ciklusok összevonása



2. Függvénykompozíció



Példa: sudoku

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?

Lehetséges ötletek:

- **Felülről lefelé:** Adjuk össze soronként, hány ötös van egy sorban!
- Minden sorra adjuk össze, hány ötös van egy sorban!
- Összegzésben megszámolás

0
0
0
0
1
0
1
1
0

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

összegzésben
megszámolás



db=3

Példa: sudoku

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

Feladat:

Hány 5-öst írtunk már be egy sudoku táblázatba?

Minden sorra adjuk össze, hány ötös van egy sorban!

Összegzésben megszámolás

Specifikáció:

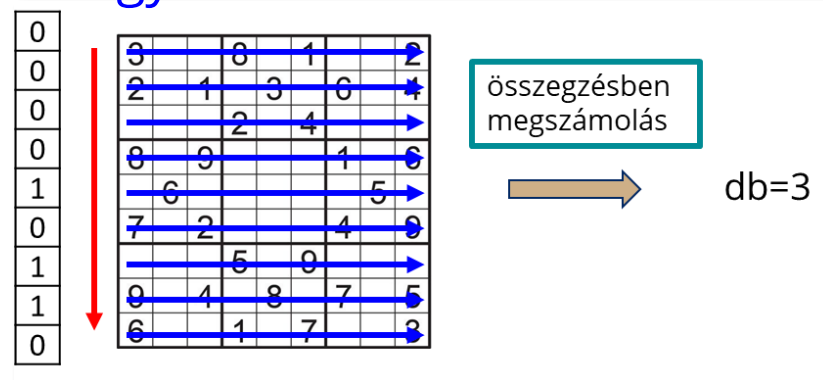
Be: $s \in \mathbb{N}[1..9, 1..9]$

Ki: $db \in \mathbb{N}$

Fv: $\text{hány5ös} : \mathbb{N} \rightarrow \mathbb{N}$, $\text{hány5ös}(i) = \text{DARAB}(j=1..9, s[i, j]=5)$

Ef: $\forall i \in [1..9] : (\forall j \in [1..9] : (0 \leq s[i, j] \leq 9))$

Uf: $db = \text{SZUMMA}(i=1..9, \text{hány5ös}(i))$



Sorokon i-vel,
oszlopokon j-vel megyünk végig

Példa: sudoku

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

Összegzés sablon:

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

Ki: $s \in H$

Ef: -

Uf: $s = \text{SZUMMA}(i = e..u, f(i))$

Megszámolás sablon:

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

Ki: $db \in \mathbb{N}$

Ef: -

Uf: $db = \text{DARAB}(i = e..u, T(i))$

Sudoku

Be: $s \in \mathbb{N}[1..9, 1..9]$

Ki: $db \in \mathbb{N}$

Fv: $\text{hány5ös} : \mathbb{N} \rightarrow \mathbb{N},$

$\text{hány5ös}(i) = \text{DARAB}(j = 1..9, s[i, j] = 5)$

Ef: $\forall i \in [1..9] : (\forall j \in [1..9] : (0 \leq s[i, j] \leq 9))$

Uf: $db = \text{SZUMMA}(i = 1..9, \text{hány5ös}(i))$

Összegzés

$s \sim db$

$e..u \sim 1..9$

$f(i) \sim \text{hány5ös}(i)$

Megszámolás

$i \sim j$

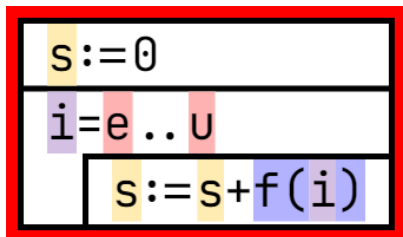
$e..u \sim 1..9$

$T(i) \sim s[i, j] = 5$

Példa: sudoku

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

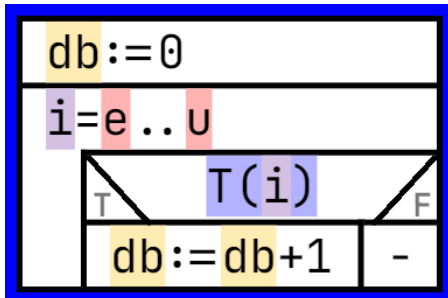
Összegzés:



Összegzés

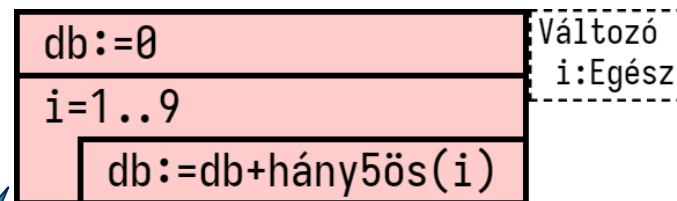
$s \sim db$
 $e \dots u \sim 1 \dots 9$
 $f(i) \sim \text{hány5ös}(i)$

Megszámolás:



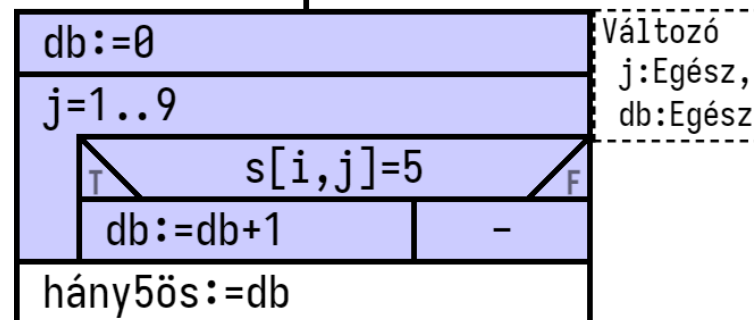
Megszámolás

$i \sim j$
 $e \dots u \sim 1 \dots 9$
 $T(i) \sim s[i, j] = 5$



Változó
 $i: \text{Egész}$

$\text{hány5ös}(i: \text{Egész}): \text{Egész}$



Változó
 $j: \text{Egész},$
 $db: \text{Egész}$

Ez ugyanaz, mint az, amit az előbb kaptunk programtranszformációkkal. Érthető, hiszen a másolás mintát minden mintán lehet használni, hiszen függvényeket használunk.

Példa leggyakoribb érték



Feladat:

A Spotify-nál tárolják, melyik Beatles dalt mikor kezdték el lejátszani. Állapítsuk meg, hogy a nyilvántartás szerint melyik a leggyakrabban lejátszott dal!



Példa leggyakoribb érték



Feladat:

A Spotify-nál tárolják, melyik Beatles dalt mikor kezdték el lejátszani? Állapítsuk meg, hogy a nyilvántartás szerint melyik a leggyakrabban lejátszott dal!

1	Here Comes The Sun
2	Hey Jude
3	Twist and Shout
4	I Wanna Hold Your Hand
5	Hey Jude
6	Here Comes The Sun
7	Let It Be
8	Hey Jude
9	Let It Be
10	Hey Dude



Igy="Hey Jude"



Példa leggyakoribb érték



Feladat:

Állapítsuk meg, hogy **melyik érték** fordult elő **leggyakrabban!**

Ötlet:

1. Tároljuk külön, melyiket hányszor játszottuk!

Problémás: új vagy régi, meg kell állapítani!

2. Egy maximumot kell meghatározni:
az adott érték **hány**szor fordul elő
(**megszámolás**), a darabszám **maximuma**
kell (**maximumkiválasztás**), és **ahol**
a legnagyobb, az **ott lévő érték!**

1	Here Comes The Sun
2	Hey Jude
3	Twist and Shout
4	I Wanna Hold Your Hand
5	Hey Jude
6	Here Comes The Sun
7	Let It Be
8	Hey Jude
9	Let It Be
10	Hey Dude

Maximumkiválasztásban **megszámolás**

Példa

leggyakoribb érték



Feladat:

Állapítsuk meg, hogy **melyik érték** fordult elő **leggyakrabban!**

Egy maximumot kell meghatározni: az adott érték **hányszor** fordul elő (**megszámolás**), a darabszám **maximuma** kell (**maximumkiválasztás**), és **ahol** a legnagyobb, az **ott lévő érték!**

Maximumkiválasztásban **megszámolás**

Specifikáció:

Be: $n \in \mathbb{N}$, $btls \in S[1..n]$

Sa: $\maxind \in \mathbb{N}$, $\maxdb \in \mathbb{N}$

Ki: $lgy \in S$

Fv: $\text{hány}: \mathbb{N} \rightarrow \mathbb{N}$, $\text{hány}(i) = \text{DARAB}(j=1..n, btls[i]=btls[j])$

Ef: -

Uf: $(\maxind, \maxdb) = \text{MAX}(i=1..n, \text{hány}(i))$ **és**

$lgy = btls[\maxind]$

maxdb-re nincs is szükség

1	Here Comes The Sun
2	Hey Jude
3	Twist and Shout
4	I Wanna Hold Your Hand
5	Hey Jude
6	Here Comes The Sun
7	Let It Be
8	Hey Jude
9	Let It Be
10	Hey Dude

A MAX értéke a legnagyobb gyakoriság lesz. Ehelyett nekünk a maxind által mutatott érték kell. Ezért a MAX után még ezt meg kell határozni ← **segédadatok**

Példa

leggyakoribb érték



Maximumkiválasztás:

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $\text{maxind} \in \mathbb{Z}$, $\text{maxért} \in H$

Ef: $e \leq u$

Uf: $(\text{maxind}, \text{maxért}) =$
 $\text{MAX}(i=e..u, f(i))$

Megszámolás:

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $db \in \mathbb{N}$

Ef: -

Uf: $db = \text{DARAB}(i=e..u, T(i))$

Leggyakoribb érték

Be: $n \in \mathbb{N}$, $\text{btls} \in S[1..n]$

Sa: $\text{maxind} \in \mathbb{N}$, $\text{maxdb} \in \mathbb{N}$

Ki: $\text{lgy} \in S$

Fv: $\text{hány}: \mathbb{N} \rightarrow \mathbb{N}$,

$\text{hány}(i) = \text{DARAB}(j=1..n, \text{btls}[i] = \text{btls}[j])$

Ef: -

Uf: $(\text{maxind}, \text{maxdb}) = \text{MAX}(i=1..n, \text{hány}(i))$

és $\text{lgy} = \text{btls}[\text{maxind}]$

Maximumkiválasztás

$\text{maxért} \sim \text{maxdb}$

$e..u \sim 1..n$

$f(i) \sim \text{hány}(i)$

Megszámolás

$i \sim j$

$e..u \sim 1..n$

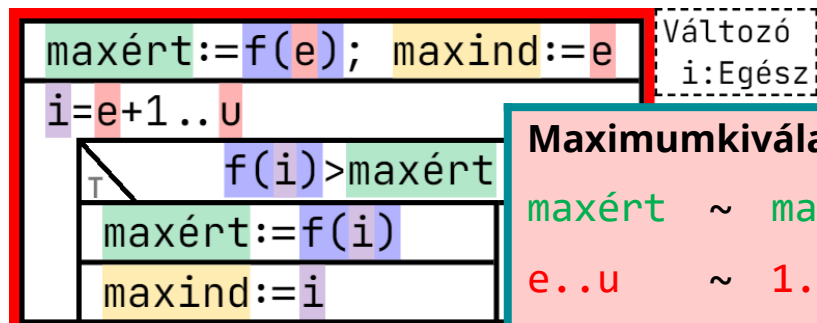
$T(i) \sim \text{btls}[i] = \text{btls}[j]$

Példa

leggyakoribb érték



Maximumkiválasztás:

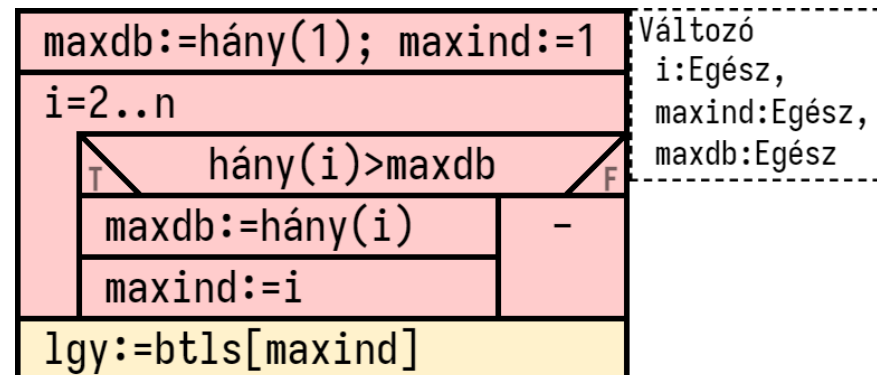


Maximumkiválasztás

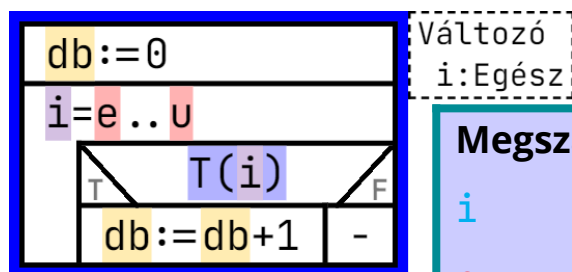
maxért ~ maxdb

e..u ~ 1..n

T(i) ~ hány(i)



Megszámolás:



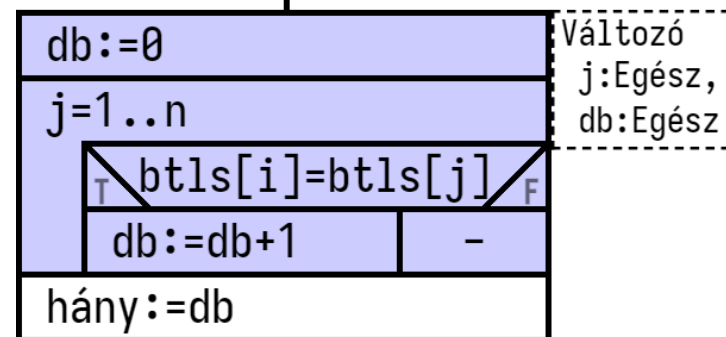
Megszámolás

i ~ j

e..u ~ 1..n

T(i) ~ btls[i]=btls[j]

hány(i:Egész): Egész



Példa

leggyakoribb érték



Megjegyzés:

Úgy is működik a megoldás, ha mindig csak azt számoljuk össze, hogy utána hány ugyanolyan értékű elem van.

Specifikáció:

Be: $n \in \mathbb{N}$, $btls \in S[1..n]$

Sa: $\maxind \in \mathbb{N}$, $\maxdb \in \mathbb{N}$

Ki: $lgy \in S$

Fv: $\text{hány} : \mathbb{N} \rightarrow \mathbb{N}$, $\text{hány}(i) = \text{DARAB}(j=i..n, btls[i]=btls[j])$

Ef: -

Uf: $(\maxind, \maxdb) = \text{MAX}(i=1..n, \text{hány}(i))$ és

$lgy = btls[\maxind]$

1	Here Comes The Sun
2	Hey Jude
3	Twist and Shout
4	I Wanna Hold Your Hand
5	Hey Jude
6	Here Comes The Sun
7	Let It Be
8	Hey Jude
9	Let It Be
10	Hey Dude

Példa

maximális oszlop

	1	2	3	4	n=5
1	1	2	4	3	1
2	2	6	4	2	1
3	5	3	2	1	0
4	2	8	2	8	2
n=5	5	3	2	3	0

Feladat:

Keressük meg a t négyzetes mátrixnak azt az oszlopát, amelyben a főátlóbeli és a feletti elemek **összege** a **legnagyobb!**

	1	2	3	4	n=5
1	1	2	4	3	1
2	2	6	4	2	1
3	5	3	2	1	0
4	2	8	2	8	2
n=5	5	3	2	3	0

Példa

maximális oszlop

	1	2	3	4	n=5
1	1	2	4	3	1
2	2	6	4	2	1
3	5	3	2	1	0
4	2	8	2	8	2
n=5	5	3	2	3	0

Feladat:

Keressük meg a t négyzetes mátrixnak azt az oszlopát, amelyben a főátlóbeli és a feletti elemek **összege** a **legnagyobb**!

Lépések:

1. Oszlop keresése a cél az átló fölötti elemek **összege** alapján
2. Ahol ez az **érték** a **legnagyobb** → keresett oszlop
3. Maximumkiválasztásban **összegzés**

	1	2	3	4	n=5
s	1	8	10	14	4

	1	2	3	4	n=5
1	1	2	4	3	1
2	2	6	4	2	1
3	5	3	2	1	0
4	2	8	2	8	2
n=5	5	3	2	3	0

maxoszlop=4

Példa

maximális oszlop

	1	2	3	4	n=5
1	1	2	4	3	1
2	2	6	4	2	1
3	5	3	2	1	0
4	2	8	2	8	2
n=5	5	3	2	3	0

Feladat:

Keressük meg a t négyzetes mátrixnak azt az oszlopát, amelyben a főátlóbeli és a feletti elemek **összege** a **legnagyobb!**

Specifikáció:

Be: $n \in \mathbb{N}$, $t \in \mathbb{Z}[1..n, 1..n]$

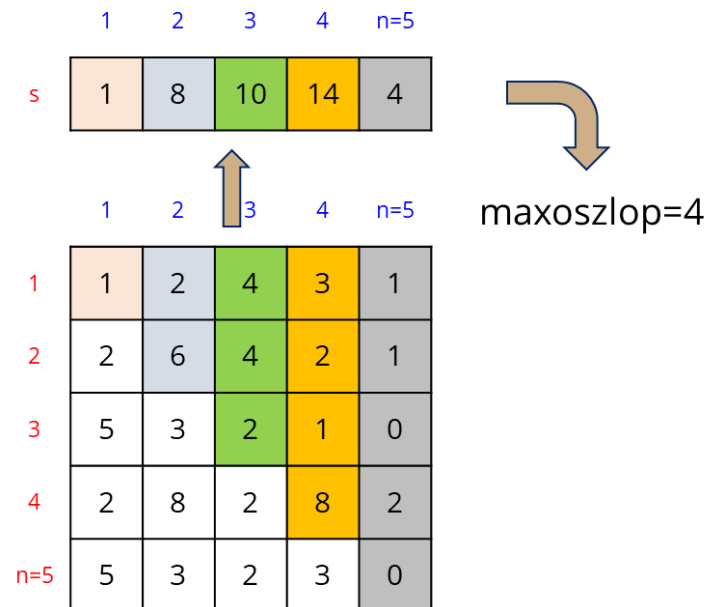
Ki: $\text{maxoszlop} \in \mathbb{N}$

Fv: $\text{összeg}: \mathbb{N} \rightarrow \mathbb{Z}$,

$\text{összeg}(o) = \text{SZUMMA}(s=1..o, t[s, o])$

Ef: -

Uf: $(\text{maxoszlop},) = \text{MAX}(o=1..n, \text{összeg}(o))$



Példa maximális oszlop

	1	2	3	4	n=5
1	1	2	4	3	1
2	2	6	4	2	1
3	5	3	2	1	0
4	2	8	2	8	2
n=5	5	3	2	3	0

Maximumkiválasztás:

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $\text{maxind} \in \mathbb{Z}$, $\text{maxért} \in \mathbb{H}$

Ef: $e \leq u$

Uf: $(\text{maxind}, \text{maxért}) =$

$\text{MAX}(i=e..u, f(i))$ Ef: -

Összegzés:

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $s \in \mathbb{H}$

Ef: -

Uf: $s = \text{SZUMMA}(i=e..u, f(i))$

Maxi

Be: $n \in \mathbb{N}$

Ki: $\text{maxoszlop} \in \mathbb{N}$

Fv: $\text{összeg}: \mathbb{N} \rightarrow \mathbb{Z}$,

$\text{összeg}(o) = \text{SZUMMA}(s=1..o, t[s, o])$

Ef: -

Uf: $(\text{maxoszlop},) = \text{MAX}(o=1..n, \text{összeg}(o))$

$\text{összeg}(o) = s \in \mathbb{Z} : (s = \text{SZUMMA}(s=1..o, t[s, o]))$

Vigyázat! Nem hívhatom ugyanúgy az

Maximumkiválasztás

$i \sim o$

$\text{maxind} \sim \text{maxoszlop}$

$e..u \sim 1..n$

$f(i) \sim \text{összeg}(i)$

Összegzés

$i \sim s$

$e..u \sim 1..o$

$f(i) \sim t[s, o]$

Példa

maximális oszlop

	1	2	3	4	n=5
1	1	2	4	3	1
2	2	6	4	2	1
3	5	3	2	1	0
4	2	8	2	8	2
n=5	5	3	2	3	0

Maximumkiválasztás:

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

Ki: $\text{maxind} \in \mathbb{Z}, \text{maxért} \in \mathbb{H}$

Ef: $e \leq u$

Uf: $(\text{maxind}, \text{maxért}) =$

$\text{MAX}(i=e..u, f(i))$ Ef: -

Összegzés:

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

Ki: $s \in \mathbb{H}$

Ef: -

Uf: $s = \text{SZUMMA}(i=e..u, f(i))$

Maximális oszlop

Be: $n \in \mathbb{N}, t \in \mathbb{Z}[1..n, 1..n]$

Ki: $\text{maxoszlop} \in \mathbb{N}$

Fv: $\text{összeg}: \mathbb{N} \rightarrow \mathbb{Z},$

$\text{összeg}(o) = \text{SZUMMA}(\text{sor}=1..o, t[\text{sor}, o])$

Ef: -

Uf: $(\text{maxoszlop},) = \text{MAX}(o=1..n, \text{összeg}(o))$

Maximumkiválasztás

$i \sim o$

$\text{maxind} \sim \text{maxoszlop}$

$e..u \sim 1..n$

$f(i) \sim \text{összeg}(i)$

Összegzés

$i \sim \text{sor}$

$e..u \sim 1..o$

$f(i) \sim t[\text{sor}, o]$

Példa

maximális oszlop

	1	2	3	4	n=5
1	1	2	4	3	1
2	2	6	4	2	1
3	5	3	2	1	0
4	2	8	2	8	2
n=5	5	3	2	3	0

Maximumkiválasztás:

```
maxért:=f(e); maxind:=e
```

```
i=e+1..u
```

Változó
i:Egész

Maximumkiválasztás

i ~ 0
maxind ~ maxoszlop
e..u ~ 1..n
f(i) ~ összeg(i)

```
maxért:=összeg(1);  
maxoszlop:=1
```

```
i=2..n
```

```
összeg(i)>maxért
```

```
maxért:=összeg(i)
```

```
maxoszlop:=i
```

Változó
i:Egész,
maxért:Egész

Összegzés:

```
s:=0
```

```
i=e..u
```

```
s:=s+f(i)
```

Változó
i:Egész

Összegzés

i ~ sor
e..u ~ 1..o
f(i) ~ t[sor,o]

összeg(i:Egész): Egész

```
s:=0
```

```
sor=1..n
```

```
s:=s+t[sor,o]
```

```
összeg:=s
```

Változó
j:Egész,
s:Egész

Példa

nyerő nap a lóversenyen



Feladat:

A Firpo testvérek apja minden nap lóversenyezik. Fiai, Johnny és Charlie arra kíváncsiak, volt-e apjuknak olyan napja, amikor úgy nyert, hogy a megelőző k napon mindig veszített?



Példa

nyerő nap a lóversenyen

1	2	3	4	5	6	7	n=8
2	-3	4	-2	-1	5	-2	23

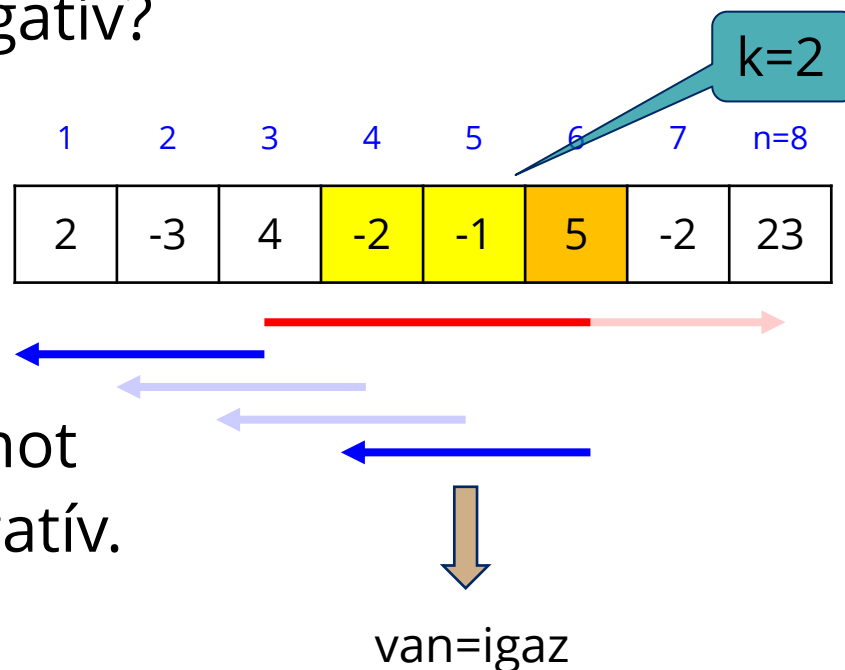


Feladat:

Van-e olyan pozitív szám egy számsorozatban, amely előtt k darab szám **mindegyike** negatív?

Lépések:

1. Azt el kell dönteni, **van-e** olyan index (\rightarrow **eldöntés**)
2. Amely előtt ha k darab számot nézzük, akkor az **összes** negatív.
3. **Eldöntésben** mind **eldöntés**



Példa

nyerő nap a lóversenyen

1	2	3	4	5	6	7	n=8
2	-3	4	-2	-1	5	-2	23



Feladat:

Van-e olyan pozitív szám egy számsorozatban, amely előtt k darab szám **mindegyike** negatív?

Eldöntésben mind eldöntés

Specifikáció:

Be: $n \in \mathbb{N}$, $\text{pénz} \in \mathbb{Z}[1..n]$, $k \in \mathbb{N}$

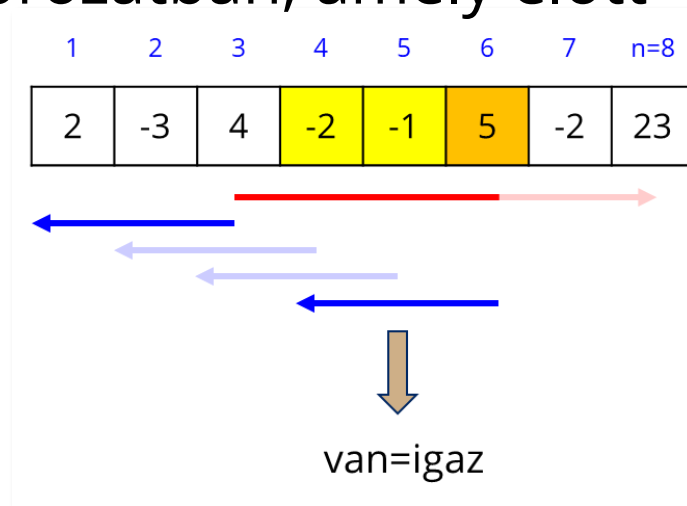
Ki: $\text{van} \in \mathbb{L}$

Fv: csupavesztés: $\mathbb{N} \rightarrow \mathbb{L}$,

$\text{csupavesztés}(i) = \text{MIND}(j=i-k..i-1, \text{pénz}[j] < 0)$

Ef: $k > 0$

Uf: $\text{van} = \text{VAN}(i=k+1..n, \text{pénz}[i] > 0 \text{ és } \text{csupavesztés}(i))$



Példa

nyerő nap a lóversenyen

1	2	3	4	5	6	7	n=8
2	-3	4	-2	-1	5	-2	23



Eldöntés:

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $\text{van} \in \mathbb{L}$

Ef: -

Uf: $\text{van} = \text{VAN}(i = e..u, T(i))$

Mind eldöntés:

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $\text{mind} \in \mathbb{L}$

Ef: -

Uf: $\text{mind} = \text{MIND}(i = e..u, T(i))$

Nyerő nap a lóversenyen:

Be: $n \in \mathbb{N}$, $\text{pénz} \in \mathbb{Z}[1..n]$, $k \in \mathbb{N}$

Ki: $\text{van} \in \mathbb{L}$

Fv: $\text{csupavesztés} : \mathbb{N} \rightarrow \mathbb{L}$,

$\text{csupavesztés}(i) = \text{MIND}(j = i - k..i - 1, \text{pénz}[j] < 0)$

Ef: $k > 0$

Uf: $\text{van} = \text{VAN}(i = k + 1..n, \text{pénz}[i] > 0 \text{ és } \text{csupavesztés}(i))$

Eldöntés

$e..u \sim k + 1..n$

$T(i) \sim \text{pénz}[i] > 0 \text{ és } \text{csupavesztés}(i)$

Mind eldöntés

$i \sim j$

$e..u \sim i - k..i - 1$

$T(i) \sim \text{pénz}[j] < 0$

Példa

nyerő nap a lóversenyen

1	2	3	4	5	6	7	n=8
2	-3	4	-2	-1	5	-2	23



Eldöntés:

$i := e$

$i \leq u$ és nem $T(i)$

$i := i + 1$

$\text{van} := i \leq u$

Változó
 $i: \text{Egész}$

Eldöntés

$e..u \sim k+1..n$

$T(i) \sim \text{pénz}[i] > 0$ és
 $\text{csupavesztés}(i)$

$i := k + 1$

$i \leq n$ és

nem($\text{pénz}[i] > 0$ és $\text{csupavesztés}(i)$)

$i := i + 1$

$\text{van} := i \leq n$

Változó
 $i: \text{Egész}$

Mind eldöntés:

$i := e$

$i \leq u$ és $T(i)$

$i := i + 1$

$\text{mind} := i > u$

Változó
 $i: \text{Egész}$

Mind eldöntés

$i \sim j$

$e..u \sim i-k..i-1$

$T(i) \sim \text{pénz}[j] < 0$

$\text{csupavesztés}(i: \text{Egész}): \text{Logikai}$

$j := i - k$

$j \leq i - 1$ és $\text{pénz}[j] < 0$

$j := j + 1$

$\text{mind} := j > i - 1$

$\text{csupavesztés} := \text{mind}$

Változó
 $j: \text{Egész},$
 $\text{mind}: \text{Logikai}$

Összefoglalás



Összefoglalás

- Bonyolultabb feladatok → több minta
 - egymás után
 - egymásba ágyazva
- Egymásba ágyazva
 - felülről lefele tervezés
 - részfeladatok függvényekként
 - mátrix: dimenzióként legalább egy minta