



ELTE | IK

PROGRAMOZÁS

Dinamikus tömb, mátrix

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 



Függvények

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

```
static double negyzet(double n) {  
    return n * n;  
}  
static int max(int a, int b) {  
    return a >= b ? a : b;  
}  
static void novel(ref int szam, int mivel) {  
    szam = szam + mivel;  
}  
static void csere(ref int x, ref int y) {  
    int z = x;  
    x = y;  
    y = z;  
}  
static void beolvas(out int n) {  
    int.TryParse(Console.ReadLine(), out n);  
}
```



Dinamikus tömb



Statikus tömb

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

Példa – kitűnő tanulók visszavezetés

Adjuk meg egy osztály kitűnő tanulóit!

Feladatsablon

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

Ki: $db \in \mathbb{N}$, $y \in H[1..db]$

Ef: -

Uf: $(db, y) =$

KIVÁLOGAT($i = e..u$,

$T(i)$,

$f(i)$)

$y \sim jelesek$

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

$T(i) \sim diákok[i].jegy=5$

$f(i) \sim diákok[i].név$

Kitűnő tanulók

Be: $n \in \mathbb{N}$, $diákok \in Diák[1..n]$,

$Diák = (név:S \times jegy:N)$

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

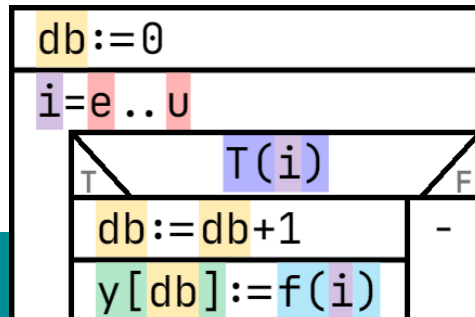
Ef: -

Uf: $(db, jelesek) =$

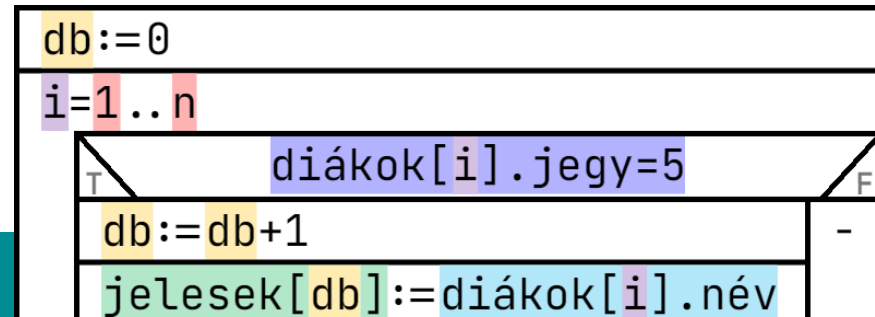
KIVÁLOGAT($i = 1..n$,

$diákok[i].jegy=5$,

$diákok[i].név$)




Változó
 $i: \text{Egész}$



Változó
 $i: \text{Egész}$

Statikus tömb kiválogatás

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



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

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



DEMO vagy házi feladat: függvényekre átírni!

Dinamikus tömb

- A dinamikus tömb változó elemszámú sorozatok ábrázolására szolgál
 - Mérete futás közben igény szerint változtatható.
 - A kiválogatásnál éppen erre van szükségünk
- Specifikáció
 - nincs értelme a dinamikusságnak
 - nincs „futás”, csak be- és kimeneti adatok
- Algoritmus

Változó `y:Tömb[1..db: Egész]`

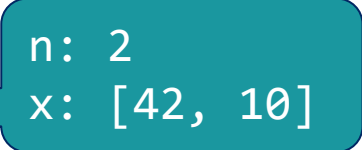


- Kód

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

Jelentése:

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

Specifikáció megjegyzés a sorozat jelöléséről

- A sorozatnak három paramétere van:
 - neve, elemek alaphalmaza, indextartománya
 - $\text{név} \in \mathcal{H}[e..u]$, például $x \in \mathbb{N}[1..5]$
- Leggyakrabban megadjuk a sorozat elemszámát külön és magát a sorozatot
 - $n \in \mathbb{N}$, $x \in \mathbb{N}[1..n]$ 
- Ha nem adjuk meg az indextartomány végét, az az aktuális adatból kiderül (nem kell elemszám külön)
 - $x \in \mathbb{N}[1..]$ 
 - $n = \text{hossz}(x)$
- Ha a sorozat kezdőindexe 1, akkor további rövidíthető:
 - $x \in \mathbb{N}[]$ 
 - $s = \text{SZUMMA}(i = \text{tól}(x) .. \text{ig}(x), x[i])$

Dinamikus tömb megjelenése

- Csak implementációs szinten
 - specifikáció: sorozat
 - algoritmus: tömb
 - kód: dinamikus tömb
- Algoritmus szintjén is → új sablon is kell
 - specifikáció: sorozat
 - algoritmus: dinamikus tömb
 - kód: dinamikus tömb

Példa – kitűnő tanuló visszavezetés

Adjuk meg egy osztály kitűnő tanulóit!

Feladatsablon

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

Ki: $db \in \mathbb{N}$, $y \in H[1..db]$

Ef: -

Uf: $(db, y) =$

KIVÁLOGAT($i = e..u$,

$T(i)$,

$f(i)$)

$y \sim jelesek$

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

$T(i) \sim diákok[i].jegy=5$

$f(i) \sim diákok[i].név$

Kitűnő tanuló

Be: $n \in \mathbb{N}$, $diákok \in Diák[1..n]$,

$Diák = (név:S \times jegy:N)$

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

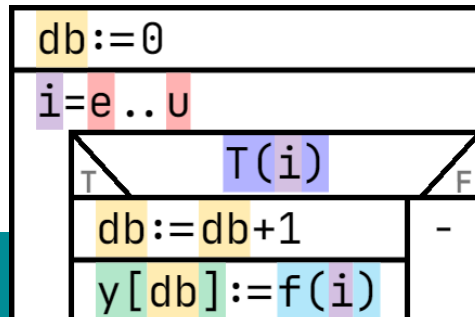
Ef: -

Uf: $(db, jelesek) =$

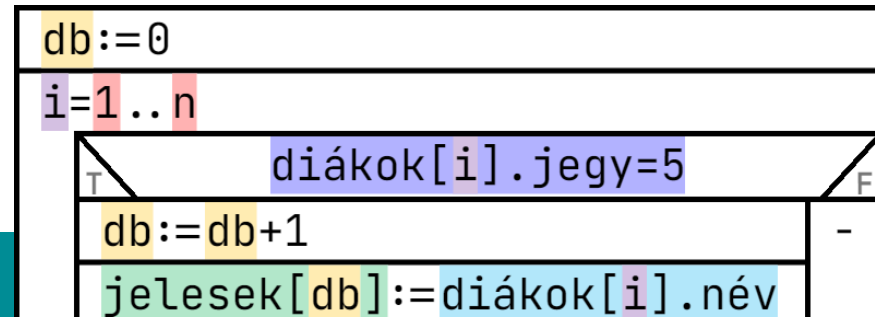
KIVÁLOGAT($i = 1..n$,

$diákok[i].jegy=5$,

$diákok[i].név$)



Változó
 i : Egész



Változó
 i : Egész

Dinamikus tömb kiválogatás

DEMO vagy házi feladat: függvényekre átírni!

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

db-ra nincs szükség

db:=0	
i=1..n	
T	F
diákok[i].jegy=5	
db:=db+1	
jeleksek[db]:=diákok[i].név	

jeleksek igény szerint lesz feltöltve

Kiválogatás sablon

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

Feladat

Adott az egész számok egy $[e..u]$ intervalluma, egy ezen értelmezett $T:[e..u] \rightarrow \text{Logikai feltétel}$ és egy $f:[e..u] \rightarrow H$ függvény. Határozzuk meg az f függvény az $[e..u]$ intervallum azon értékeinél felvett értékeit, amelyekre a T feltétel teljesül!

Specifikáció

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

Ki: $y \in H[1..]$

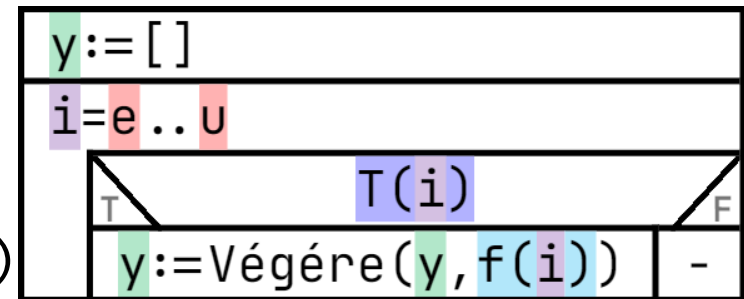
Ef: -

Uf: $\forall i \in [1..hossz(y)]:$
 $\exists j \in [e..u]: T(j) \text{ és } y[i] = f(j)$
 és $y \subseteq (f(e), f(e+1), \dots, f(u))$

Rövidítve:

Uf: $(,y) = \text{KIVÁLOGAT}(i=e..u, T(i), f(i))$

Algoritmus



Példa – kitűnő tanuló visszavezetés

Adjuk meg egy osztály kitűnő tanulóit!

Feladatsablon

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

Ki: $y \in H[1..]$

Ef: -

Uf: $(, y) =$

KIVÁLOGAT($i = e..u,$

$T(i),$

$f(i))$

$y \sim \text{jелеsek}$

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

$T(i) \sim \text{diákok}[i].\text{jegy}=5$

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

Kitűnő tanuló

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

$\text{Diák} = (\text{név}: S \times \text{jegy}: N)$

Ki: $\text{jелеsek} \in S[1..]$

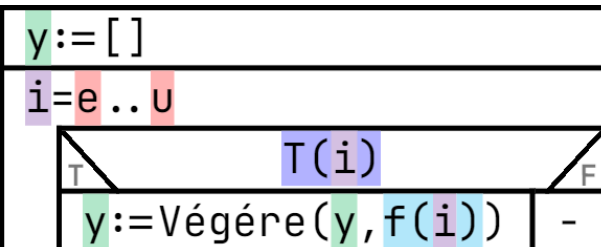
Ef: -

Uf: $(, \text{jелеsek}) =$

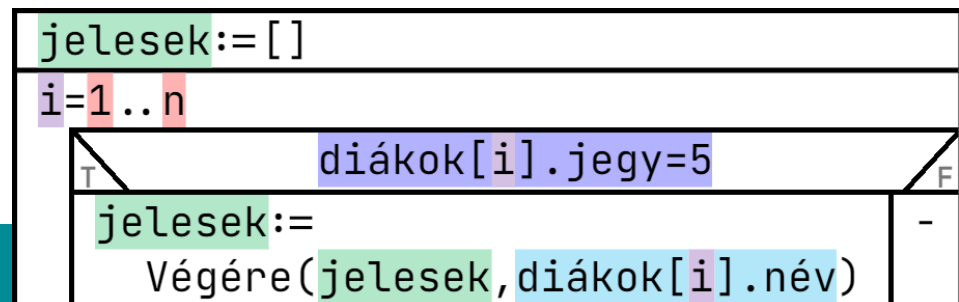
KIVÁLOGAT($i = 1..n,$

$\text{diákok}[i].\text{jegy}=5,$

$\text{diákok}[i].\text{név})$



Változó
 $i: \text{Egész}$



Változó
 $i: \text{Egész}$

Dinamikus tömb kiválogatás

DEMO vagy házi feladat: függvényekre átírni!

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

jeleksek:=[]

i=1..n

diakok[i].jegy=5

jeleksek:=

Végére(jeleksek, diakok[i].név)

jeleksek igény szerint lesz feltöltve

Mátrixok
egyelőre trükkösen



Mátrix



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

	x
1	-4
2	2
3	5

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

Példa

Példa:	3		8	1		2
	2	1		3	6	4
			2	4		
	8	9			1	6
		6			5	
	7	2			4	9
			5	9		
	9	4		8	7	5
	6		1	7		3
→ db=3						

Feladat:

Hány 5-öst írtunk már be egy sudoku táblázatba?

Specifikáció:

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

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

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

~~Uf: $db = \text{DARAB}(i=1..9, j=1..9, s[i, j]=5)$~~

Uf: $db = \text{DARAB}(i=??..??, s[??]=5)$

Olyan feladat, amelyben egy sémát kell alkalmazni egy mátrixra.

Nincs ilyen rövidítésünk.

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

2D \rightarrow 1D trükk

- Ábrázoljuk a kétdimenziós mátrixot egydimenzióban, pl. sorfolytonosan!

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

1 2 3 4 5 6 7 8 9

$$i = (k - 1) \text{ div } 3 + 1$$

$$j = (k - 1) \text{ mod } 3 + 1$$

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

Pl. $k=7 \rightarrow i=(7-1) \text{ div } 3 + 1=3, j=(7-1) \text{ mod } 3 + 1=1$

Pl. $i=2, j=3 \rightarrow k=(2-1)*3+3=6$

Példa

Példa:	3			8		1		2	→ db=3
	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?

Specifikáció:

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

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

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

Uf: $db = \text{DARAB}(k = 1..9*9, s[(k-1) \text{ div } 9 + 1, (k-1) \text{ mod } 9 + 1] = 5)$

Mátrix

Hány 5-öst írtunk már be egy sudoku táblázatba?

Feladatsablon

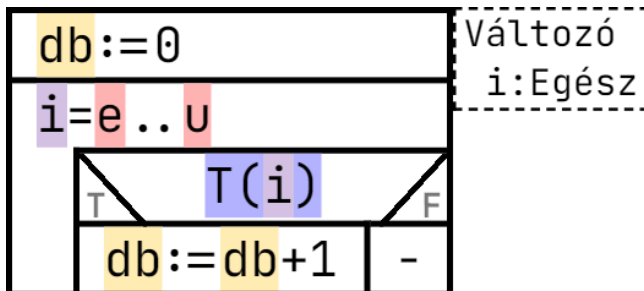
Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

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

Ef: -

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

i	\sim	k
$e..u$	\sim	$1..9*9$
$T(i)$	\sim	$s[(k-1) \text{ div } 9 + 1, (k-1) \text{ mod } 9 + 1] = 5$



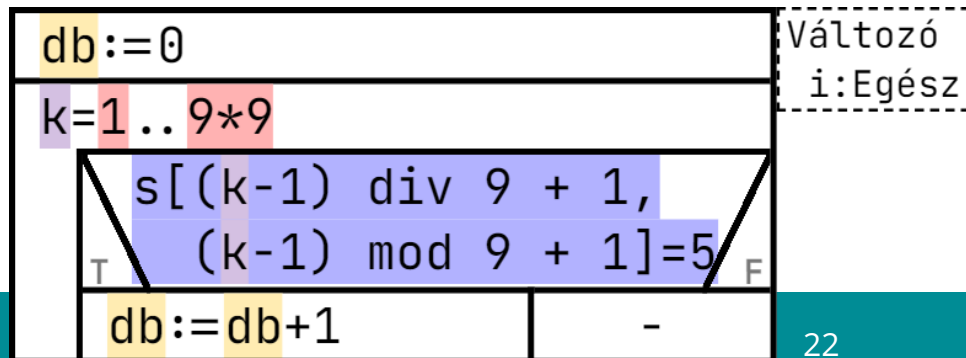
Sudoku

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

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

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

Uf: $db = \text{DARAB}(k = 1..9*9, s[(k-1) \text{ div } 9 + 1, (k-1) \text{ mod } 9 + 1] = 5)$



Mátrix

DEMO vagy házi feladat: függvényekre átírni!

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

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

Mátrix

Add meg soronként a számokat, soron belül szóközzel elválasztva.
1. sor: 3 0 0 8 0 1 0 0 2
2. sor:

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

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

DEMO vagy házi feladat: függvényekre átírni!

2D → 1D trükk

0-tól indexelve

- Ábrázoljuk a kétdimenziós mátrixot egydimenzióban, pl. sorfolytonosan!

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

0 1 2 3 4 5 6 7 8

$$i = k \text{ div } 3$$

$$j = k \text{ mod } 3$$

$$k = i * 3 + j$$

Pl. $k=7 \rightarrow i=7 \text{ div } 3=2, j=7 \text{ mod } 3=1$

Pl. $i=2, j=1 \rightarrow k=2*3+1=7$



Mátrix 0-tól indexelve

Hány 5-öst írtunk már be egy
sudoku táblázatba?

Feladatsablon

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

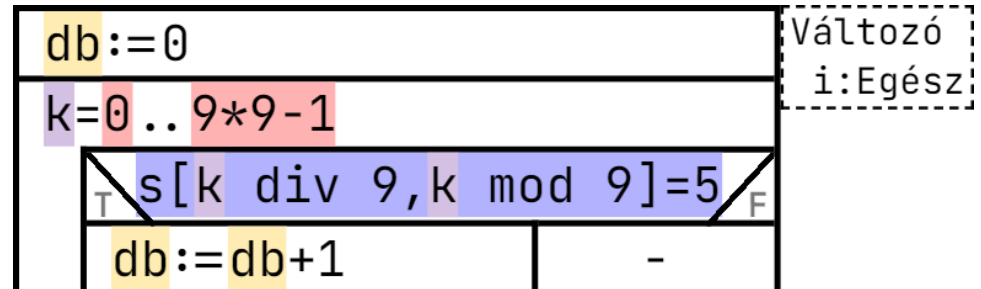
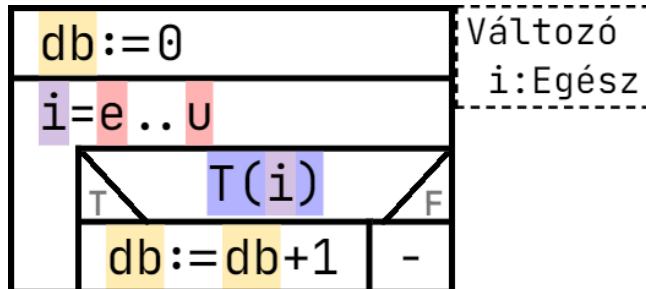
Be: $s \in \mathbb{N}[0..8, 0..8]$

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

Ef: $\forall i \in [0..8]: (\forall j \in [0..8]: (0 \leq s[i, j] \leq 9))$

Uf: $db = \text{DARAB}(k = 0..9*9-1, s[k \text{ div } 9, k \text{ mod } 9] = 5)$

i	\sim	k
$e..u$	\sim	$0..9*9-1$
$T(i)$	\sim	$s[k \text{ div } 9, k \text{ mod } 9] = 5$



Összefoglalás



Összefoglalás

- Dinamikus tömb
- Mátrix $2D \rightarrow 1D$ trükk (nem sokáig)