



ELTE | IK

PROGRAMOZÁS

Adatszerkezetek

Horváth Győző



Ismétlés



Szekvencia

Algoritmus:

utasítás1
utasítás2
utasítás3

Kód:

```
utasítás1;  
utasítás2;  
utasítás3;
```

Elágazások

Algoritmus:

feltétel	
T	F
utasítások1	utasítások2
...	...

kétirányú

\ feltétel1	\ feltétel2	\ ...	\ egyébként
utasítások1	utasítások2	...	utasítások

többsíriányú

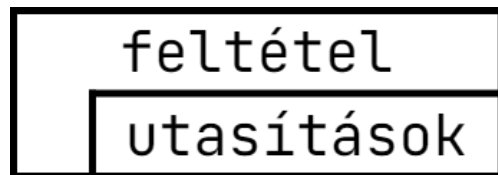
Kód:

```
if (feltétel1) {  
    utasítások1;  
} else {  
    utasítások2;  
}
```

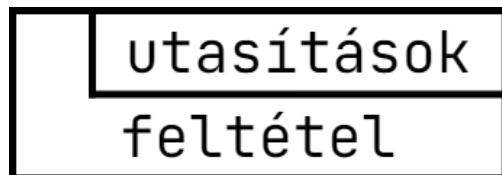
```
if (feltétel1) {  
    utasítások1;  
}  
else if (feltétel2) {  
    utasítások2;  
}  
// ...  
else {  
    utasítások;  
}
```

Ciklusok

Feltételes ciklus:

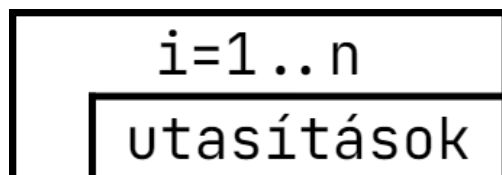


```
while (feltétel){  
    utasítások  
}
```

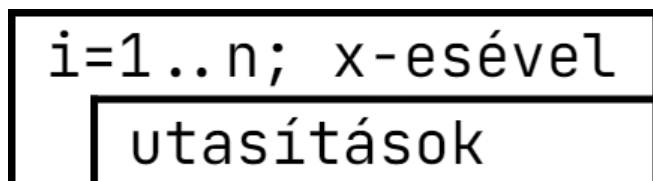


```
do {  
    utasítások  
} while (feltétel);
```

Számlálós ciklus:



```
for (int i=1;i<=n;++i) {  
    utasítások  
}
```



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

Rekord



Rekord

Példák:

$x=3,3; y=2,1 \rightarrow sn=1$

$x=3,3; y=-2 \rightarrow sn=4$

...

Feladat:

Adjuk meg, hogy egy síkbeli pont melyik síknegyedbe esik!

Specifikáció₁ és algoritmus₁:

Be: $x \in \mathbb{R}, y \in \mathbb{R}$

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

Ef: -

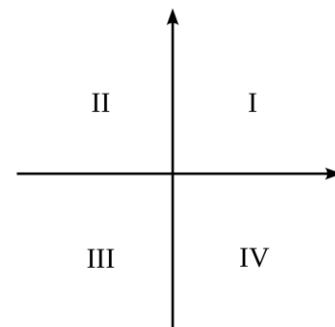
Uf: $((x \geq 0 \text{ és } y \geq 0) \rightarrow sn=1) \text{ és}$

$((x < 0 \text{ és } y \geq 0) \rightarrow sn=2) \text{ és}$

$((x < 0 \text{ és } y < 0) \rightarrow sn=3) \text{ és}$

$((x \geq 0 \text{ és } y < 0) \rightarrow sn=4)$

$x \geq 0 \text{ és } y \geq 0$	$x < 0 \text{ és } y \geq 0$	$x < 0 \text{ és } y < 0$	$x \geq 0 \text{ és } y < 0$
$sn:=1$	$sn:=2$	$sn:=3$	$sn:=4$



Be: $x \in \mathbb{R}, y \in \mathbb{R}$
Ki: $sn \in \mathbb{N}$

Rekord

// 1. deklarálás

```
double x, y;
```

```
int sn = 0;
```

// 2. beolvasás

```
Console.Write("x = ");
```

```
double.TryParse(Console.ReadLine(), out x);
```

```
Console.Write("y = ");
```

```
double.TryParse(Console.ReadLine(), out y);
```

// 3. feldolgozás

```
if (x >= 0 && y >= 0) { sn = 1; }
```

```
else if (x < 0 && y >= 0) { sn = 2; }
```

```
else if (x < 0 && y < 0) { sn = 3; }
```

```
else if (x >= 0 && y < 0) { sn = 4; }
```

// 4. kiírás

```
Console.WriteLine("Síknegyed = {0}", sn);
```

$x \geq 0$ és $y \geq 0$	$x < 0$ és $y \geq 0$	$x < 0$ és $y < 0$	$x \geq 0$ és $y < 0$
sn:=1	sn:=2	sn:=3	sn:=4

Rekord

Specifikáció₂ és algoritmus₂:

Uf: $(x \geq 0 \rightarrow (y \geq 0 \rightarrow sn = 1 \text{ és } y < 0 \rightarrow sn = 4))$

és

$(x < 0 \rightarrow (y \geq 0 \rightarrow sn = 2 \text{ és } y < 0 \rightarrow sn = 3))$

T		$x \geq 0$				F	
T	$y \geq 0$		F	T	$y \geq 0$		F
sn:=1		sn:=4		sn:=2		sn:=3	

```
if (x >= 0) {  
    if (y >= 0) { sn = 1; }  
    else { sn = 4; }  
}  
else {  
    if (y >= 0) { sn = 2; }  
    else { sn = 3; }  
}
```

Rekord

- Jelenlegi megoldás:
 - nincs x és y között szemantikus kapcsolat
 - pedig nemcsak két szám, hanem **együtt** jelölnek egy koordinátpárt
- **Rekord:**
 - *különböző funkciójú adatok egybezárása*
 - szemantikus egység létrehozása
 - „funkció”: mit *jelent* az adat

Rekord

Példák:

$(x=3,3; y=2,1) \rightarrow sn=1$

$(x=3,3; y=-2) \rightarrow sn=4$

Specifikációbeli jelölés:

- egy adat

- $x \in R$

$p \in R \times R$

- adattöbbség

új halmaz definíciója

- $p \in \text{Pont}, \text{Pont} = (x:R \times y:R)$

- x : direkt szorzat

- hivatkozás a **nevükkel**

- $p.x$ ($p.x \in R$)

- $p.y$ ($p.y \in R$)

Direkt szorzat példa:

$a \in [1..2], b \in [4..5]$

a	b
1	4
1	5
2	4
2	5

A direkt szorzat felsorolja az összes lehetséges adatként.

Rekord

Példák:

$(x=3,3; y=2,1) \rightarrow sn=1$

$(x=3,3; y=-2) \rightarrow sn=4$

Feladat:

Adjuk meg, hogy egy síkbeli pont melyik síknegyedbe esik!

Specifikáció₃ és algoritmus₃:

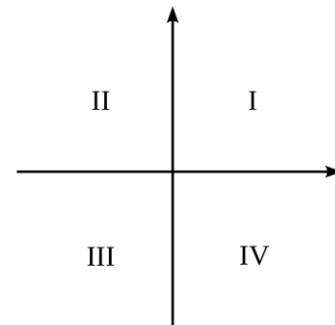
Be: $p \in \text{Pont}$,
Pont = $(x:R \ x \ y:R)$

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

Ef: -

Uf: $((p.x \geq 0 \text{ és } p.y \geq 0) \rightarrow sn=1)$ és
 $((p.x < 0 \text{ és } p.y \geq 0) \rightarrow sn=2)$ és
 $((p.x < 0 \text{ és } p.y < 0) \rightarrow sn=3)$ és
 $((p.x \geq 0 \text{ és } p.y < 0) \rightarrow sn=4)$

$p.x \geq 0 \text{ és } p.y \geq 0$	$p.x < 0 \text{ és } p.y \geq 0$	$p.x < 0 \text{ és } p.y < 0$	$p.x \geq 0 \text{ és } p.y < 0$
sn:=1	sn:=2	sn:=3	sn:=4



Rekord

Specifikáció → algoritmus_{adateleírás}

- $p \in \text{Pont}$, $\text{Pont} = \boxed{x:\text{R}} \boxed{x} \boxed{y:\text{R}}$

Típus $\text{Pont} = \text{Rekord}(x:\text{Valós}, y:\text{Valós})$

Változó $p:\text{Pont}$

Típus $\text{Pont} = \text{Rekord}(x, y:\text{Valós})$

Tehát a **Pont** egy új **adattípus**.

- A rekordok **összetett** adatszerkezetek, a részeikre „**nevük**” által meghatározott **szelektorok**kal hivatkozunk:
 $p = (p.x, p.y)$.

Rekord – típusdefiniálás kódban

Specifikáció → algoritmus → kód:

- Típus

Pont=Rekord(x,y:Valós)

Típusdefiníció

C# típusdefiníció:

```
struct Pont  
{  
    public double x,y;  
}
```

típusazonosító

hozzáférési jog

mezőtípus

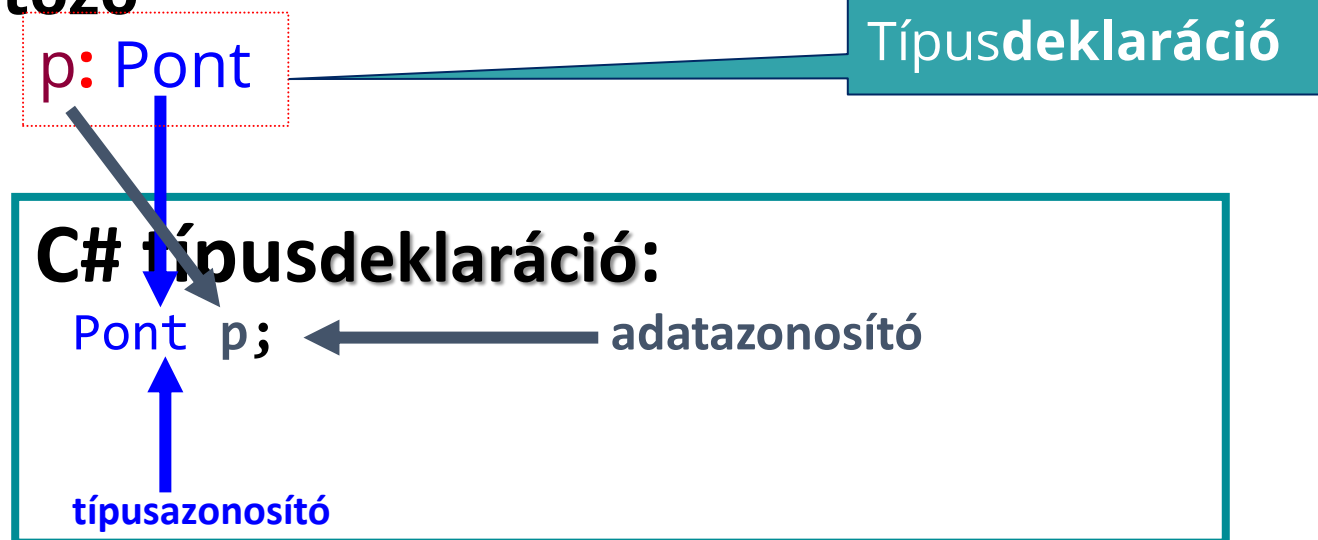
mezőazonosítók

```
struct Pont  
{  
    public double x;  
    public double y;  
}
```

Rekord – típusdeklarálás kódban

Specifikáció → algoritmus → kód:

- **Változó**



Kód

$p.x \geq 0$ és $p.y \geq 0$	$p.x < 0$ és $p.y \geq 0$	$p.x < 0$ és $p.y < 0$	$p.x \geq 0$ és $p.y < 0$
sn:=1	sn:=2	sn:=3	sn:=4

```

struct Pont {
    public double x, y;
}
static void Main(string[] args) {
    // 1. deklarálás
    //double x, y;
    Pont p;
    int sn = 0;
    // 2. beolvasás
    Console.Write("x = ");
    double.TryParse(Console.ReadLine(), out p.x);
    Console.Write("y = ");
    double.TryParse(Console.ReadLine(), out p.y);
    // 3. feldolgozás
    if (p.x >= 0 && p.y >= 0) { sn = 1; }
    else if (p.x < 0 && p.y >= 0) { sn = 2; }
    else if (p.x < 0 && p.y < 0) { sn = 3; }
    else if (p.x >= 0 && p.y < 0) { sn = 4; }
    // 4. kiírás
    Console.WriteLine("Síknegyed = {0}", sn);
}

```

Be: $p \in \text{Pont}$,
 $\text{Pont} = (x:R \ x \ y:R)$
 Ki: $sn \in \mathbb{N}$
 Ef: -
 Uf: $((p.x \geq 0 \text{ és } p.y \geq 0) \rightarrow sn=1)$ és
 $((p.x < 0 \text{ és } p.y \geq 0) \rightarrow sn=2)$ és
 $((p.x < 0 \text{ és } p.y < 0) \rightarrow sn=3)$ és
 $((p.x \geq 0 \text{ és } p.y < 0) \rightarrow sn=4)$

Tömb



Sok adat

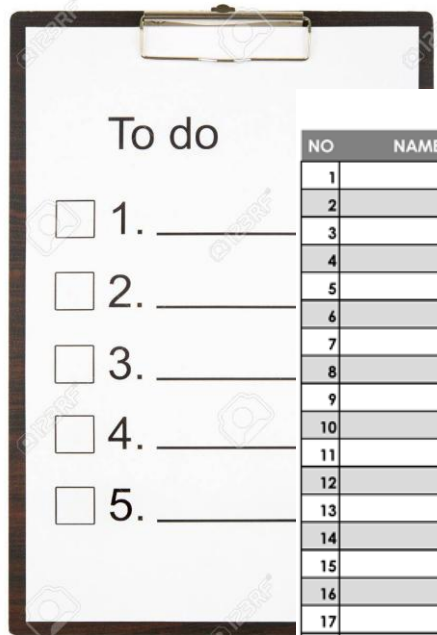
- Vannak olyan feladatok, amelyekben az adatleírás nehezen vagy egyáltalán nem végezhető el elemi típusokkal vagy rekorddal.
- Ezek, amikor **sok egyforma adattal** dolgozunk
- Példa: melyik a legnagyobb?
 - Be: $a \in \mathbb{Z}$, $b \in \mathbb{Z}$ \rightarrow Ki: $\max \in \mathbb{Z}$
 - Be: $a \in \mathbb{Z}$, $b \in \mathbb{Z}$, $c \in \mathbb{Z}$ \rightarrow Ki: $\max \in \mathbb{Z}$
 - Be: $a \in \mathbb{Z}$, $b \in \mathbb{Z}$, $c \in \mathbb{Z}$, $d \in \mathbb{Z}$ \rightarrow Ki: $\max \in \mathbb{Z}$
- Gond:
 - ez a megoldás nem skálázódik jól a számossággal
 - az utófeltétel egyre bonyolultabb
 - ennek megfelelően az algoritmus is 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...



Wedding Guest List

NO	NAME(S)	ADDRESS	CITY	STATE	ZIP CODE	✓
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						



Tömb vs táblázatkezelő oszlopa

Excel

	A	
1	zöld	
2	piros	
3	sárga	
4	fehér	
5	fekete	
6		
7		
8		

hivatkozás:

A2 → piros

Tömb

	szín
0	zöld
1	piros
2	sárga
3	fehér
4	fekete

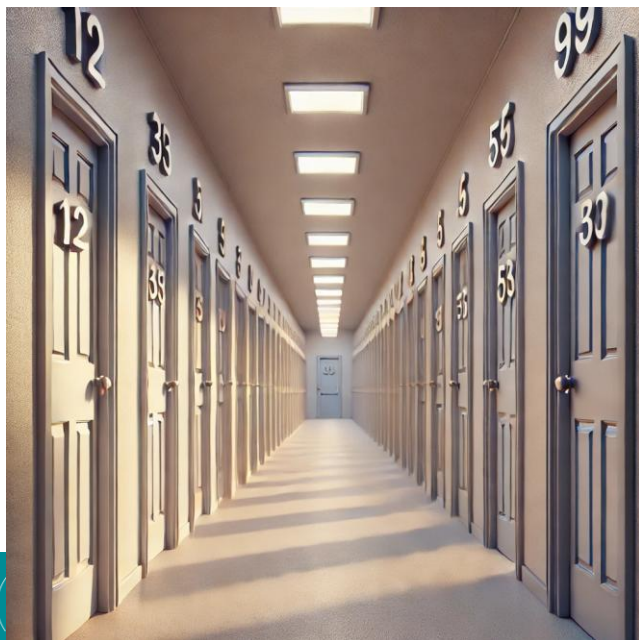
hivatkozás:

szín[**2**] → sárga

Számozott felsorolás,
hozzáférés a sorszámon keresztül

Tömb: index \rightarrow érték

- A tömb egy eleméhez az indexén (sorszámán) keresztül férünk hozzá: szín[2]
- Azaz az értékek *közvetlenül* nem érhetők el, csak *közvetve* az indexeken (sorszámokon) keresztül
- Az indextartomány tehát nagyon fontos!



	szín
0	zöld
1	piros
2	sárga
3	fehér
4	fekete

Tömb – specifikáció

Azonos funkció →
azonos halmazbeli

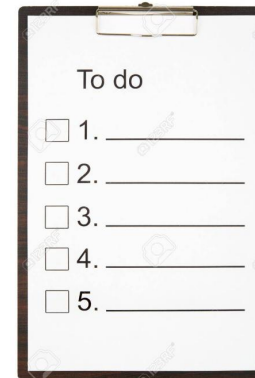
- **Sorozat:** azonos funkciójú elemek egymásutánja, az elemei sorszámozhatók.

- Példa a definiálásra (Be, Ki):

- $teendők \in S[1..5]$
- $totó \in K[1..14]$
- $n \in \mathbb{N}$, $vendégek \in S[1..n]$
- $dobozok \in S[1..] \equiv dobozok \in S[]$

- Példa a használatra (=hivatkozás egy elemre, Ef, Uf)

- $teendők[1]$
- $vendégek[n-2]$



Az aktuális elemszám
lekérdezhető: $hossz(dobozok)$

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

Kérdés: az elemek lehetnek sorozatok, azaz van-e **sorozatok sorozata**?

Tömb – algoritmus

- **Tömb:** véges hosszúságú *sorozat algoritmikus párja*, amelynek **i-edik tag**jával 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]
 - dobozok:Tömb[**1..**:Szöveg]
- Példa a használatra:
 - hivatkozás: todo[2]
 - értékadás: vendégek[1]:="Miklós atya"

Specifikáció:

teendők $\in S[1..5]$

totó $\in K[1..14]$

$n \in \mathbb{N}$, vendégek $\in S[1..n]$

dobozok $\in S[1..]$

adott a legkisebb és a legnagyobb index,
vagy az **elemszám**

Tömb – C# kód

- Statikus tömb – ismert méret

- `string[] teendok=new string[5];`
- `char[] toto=new char[14];`

Algoritmus:

- `teendők:Tömb[1..5:Szöveg]`

- Statikus tömb – max.méret

- `int n;`
`const int MAXN = 100;`
`string[] vendek = new string[MAXN];`

Algoritmus:

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

- Statikus tömb – igény szerinti méret

- `int n;`
`int.TryParse(Console.ReadLine(), out n);`
`string[] vendek = new string[n];`

vendégek

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

Algoritmus:

- `vendégek:Tömb[1..n:Szöveg]`

vendégek

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

Tömb – algoritmus → kód

C#-ban a tömbök 0-tól indexelődnek.

1. ötlet: ne használjuk a 0. elemet!

Deklarációs példa:

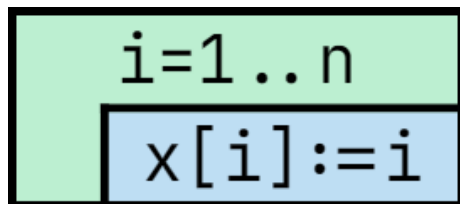
`x:Tömb[1..n:Valós]`

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

C# kód:

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

Algoritmus:



C# kód:

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

Tömb – algoritmus → kód

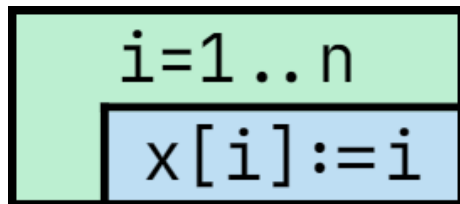
C#-ban a tömbök 0-tól indexelődnek.

2. ötlet: indexeltolás!

Deklarációs példa:

`x:Tömb[1..n:Valós]`

Algoritmus:



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

C# kód:

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

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

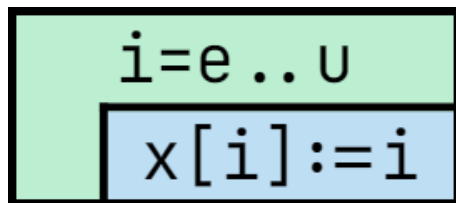
C#-ban a tömbök 0-tól indexelődnek.

3. ötlet: általános esetben!

Deklarációs példa:

`x:Tömb[e..u:Valós]`

Algoritmus:



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

C# kód:

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

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

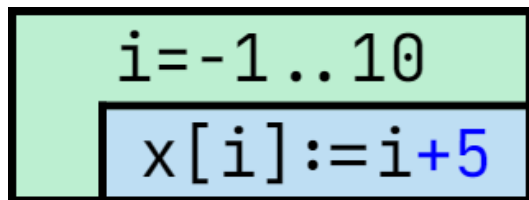
C#-ban a tömbök 0-tól indexelődnek.

3. ötlet: példa

Deklarációs példa:

`x:Tömb[-1..10:Valós]`

Algoritmus:



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

C# kód:

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

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:**

- $színek \in S[1..4] =$
["zöld", "piros", "tök", "makk"]

- **Algoritmus:**

- Konstans SZÍNEK:Tömb[1..4:Szöveg]=
["zöld", "piros", "tök", "makk"]

- **Kód:**

```
string[] SZINEK = new string[4]
    { "zöld", "piros", "tök", "makk" };
// vagy
string[] SZINEK =
    { "zöld", "piros", "tök", "makk" };
```

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

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

Példák konstans tömbökre



Feladat elágazásra – vagy más kell?

Feladat:

A japán naptár 60 éves ciklusokat tartalmaz, az éveket párosítják, s mindegyik párhoz valamilyen színt rendelnek (zöld, piros, sárga, fehér, fekete).

- o 1,2,11,12, ...,51,52: zöld évek
- o 3,4,13,14,...,53,54: piros évek
- o 5,6,15,16,...55,56: sárga évek
- o 7,8,17,18,...57,58: fehér évek
- o 9,10,19,20,...,59,60: fekete évek

Tudjuk, hogy 1984-ben indult az utolsó ciklus, amely 2043-ban fog véget érni.

Írj programot, amely megadja egy M évről ($1984 \leq M \leq 2043$), hogy milyen színű!

Feladat elágazásra – vagy más kel

Példa: $n=2024 \rightarrow s=„zöld”$

Specifikáció:

Be: $év \in \mathbb{N}$

Sa: $y \in \mathbb{N}$

Ki: $s \in S$

Ef: $1984 \leq év \leq 2043$

Uf: $y = ((év - 1984) \bmod 10) \div 2$ és

$y=0 \rightarrow s=„zöld”$ és

$y=1 \rightarrow s=„piros”$ és

...

- o 1,2,11,12, ...,51,52: zöld évek
- o 3,4,13,14,...,53,54: piros évek
- o 5,6,15,16,...55,56: sárga évek
- o 7,8,17,18,...57,58: fehér évek
- o 9,10,19,20,...,59,60: fekete évek

év	év-1984	mod 10	div 2
1984	0	0	0
1985	1	1	0
1986	2	2	1
1987	3	3	1
1988	4	4	2
1989	5	5	2
1990	6	6	3
1991	7	7	3
1992	8	8	4
1993	9	9	4
1994	10	0	0
1995	11	1	0
1996	12	2	1
1997	13	3	1
1998	14	4	2
1999	15	5	2
2000	16	6	3
2001	17	7	3
2002	18	8	4
2003	19	9	4

Állapottér bővítés

y	szín
0	zöld
1	piros
2	sárga
3	fehér
4	fekete

Feladat elágazásra – vagy más kell?

Algoritmus:

Local declarations

Változó y :Egész

Segédadatnak megfelelő
lokális változó deklarálása

$y := ((\text{év} - 1984) \bmod 10) \div 2$

$y=0$	$y=1$	$y=2$	$y=3$	$y=4$
$s := \text{"zöld"}$	$s := \text{"piros"}$	$s := \text{"sárga"}$	$s := \text{"fehér"}$	$s := \text{"fekete"}$

Kérdés:

Akkor is ezt tennénk, ha 5 helyett
90 ágat kellene írunk?

→ tömb

Be: év ∈

Sa: $y \in$

Ki: $s \in$

Ef: $1984 \leq \text{év} \leq 2043$

Uf: $y = ((\text{év} - 1984) \bmod 10) \div 2$ és
 $y=0 \rightarrow s = \text{"zöld"}$ és
 $y=1 \rightarrow s = \text{"piros"}$ és
...

y	szín
0	zöld
1	piros
2	sárga
3	fehér
4	fekete

Tömb

Összerendelés

y	szín
0	zöld
1	piros
2	sárga
3	fehér
4	fekete

Tömb

	szín
0	zöld
1	piros
2	sárga
3	fehér
4	fekete

hivatkozás:

szín[2] → sárga

Sorozat → tömb

SZÍNEK

0	zöld
1	piros
2	sárga
3	fehér
4	fekete

Példa – színes évek:

A feladat specifikációjában bevezetünk egy szöveg konstansokból álló **sorozat**ot:

```
színek ∈ S[0..4] =  
    ["zöld", "piros", "sárga", "fehér", "fekete"]
```

Az algoritmusban **tömb**bel reprezentálhatjuk:

```
Konstans SZÍNEK: Tömb[0..4: Szöveg] =  
    ["zöld", "piros", "sárga", "fehér", "fekete"]
```

Elágazás helyett tömb

Specifikáció (végleges):

Be: $\text{év} \in \mathbb{N}$,
színek $\in S[0..4] =$
["zöld", "piros", "sárga", "fehér", "fekete"]

Sa: $y \in \mathbb{N}$

Ki: $s \in S$

Ef: $1984 \leq \text{év} \leq 2043$

Uf: $y = ((\text{év} - 1984) \bmod 10) \div 2$ és
 $s = \text{színek}[y]$

Be: $\text{év} \in \mathbb{N}$

Sa: $y \in \mathbb{N}$

Ki: $s \in S$

Ef: $1984 \leq \text{év} \leq 2043$

Uf: $y = ((\text{év} - 1984) \bmod 10) \div 2$ és
 $y=0 \rightarrow s = \text{"zöld"}$ és
 $y=1 \rightarrow s = \text{"piros"}$ és
...

színek	
0	zöld
1	piros
2	sárga
3	fehér
4	fekete

Elágazás helyett tömb

Adatreprezentálás:

Változó

év:Egész

s:Szöveg

Konstans

SZÍNEK:Tömb[0..4:Szöveg]=
("zöld","piros","sárga","fehér","fekete")

Programparaméterek
deklarálása

Be: év $\in\mathbb{N}$,
színek $\in\mathbb{S}[0..4]=$
["zöld","piros","sárga","fehér","fekete"]
Sa: y $\in\mathbb{N}$
Ki: s $\in\mathbb{S}$

Algoritmus:

Uf: y=((év-1984) mod 10) div 2 és
s=színek[y]

Local declarations

Változó y:Egész

y:=((év-1984) Mod 10) Div 2

s:=SZÍNEK[y]

Konstans tömb alkalmazása

Feladat:

Példa: $n=42 \rightarrow s=$ „negyvenkettő”

Írj programot, amely egy 1 és 99 közötti számot betűkkel ír ki!

Specifikáció:

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

$\text{egy} \in S[0..9] = ["", "egy", \dots, "kilenc"],$
 $\text{tíz} \in S[0..9] = ["", "tizen", \dots, "kilencven"]$

Leglogikusabb helyre téve.
Az algoritmus szempontjából
„adottság”, azaz bemenet...

Ki: $s \in S$

Ef: $1 \leq n \leq 99$

Uf: $n=10 \rightarrow s=$ „tíz” és

$n=20 \rightarrow s=$ „húsz” és

$(n \neq 10 \text{ és } n \neq 20) \rightarrow$

$s = \text{tíz}[n \text{ div } 10] + \text{egy}[n \text{ mod } 10]$

	tíz	egy
0		
1	tizen	egy
2	huszon	kettő
3	harminc	három
4	negyven	négy
5	ötven	öt
6	hatvan	hat
7	hetven	hét
8	nyolcvan	nyolc
9	kilencven	kilenc

Konstans tömb alkalmazása

Algoritmus:

Változó n : Egész

Konstans **EGYES**: Tömb[0..9: Szöveg] =
("","egy",..., "kilenc")

TIZES: Tömb[0..9: Szöveg] =
("","tizen",..., "kilencven")

Változó s : Szöveg

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

$\text{egyes} \in S[0..9] = ["", "egy", \dots, "kilenc"],$
 $\text{tizes} \in S[0..9] = ["", "tizen", \dots, "kilencven"]$

Ki: $s \in S$

Uf: $n=10 \rightarrow s="tíz"$ és
 $n=20 \rightarrow s="húsz"$ és
 $(n \neq 10 \text{ és } n \neq 20) \rightarrow$
 $s = \text{tizes}[n \text{ div } 10] + \text{egyes}[n \text{ mod } 10]$

$n=10$	$n=20$	$n \neq 10 \text{ és } n \neq 20$
$s := "tíz"$	$s := "húsz"$	$s := \text{TIZES}[n \text{ div } 10] +$ $\text{EGYES}[n \text{ mod } 10]$

Konstans tömb alkalmazása

Feladat:

Írj programot, amely egy hónapnévhez a sorszámát rendeli!

Specifikáció:

Be: $h \in S$,

$\text{hónév} \in S[1..12] =$
["január", ..., "december"]

Ki: $s \in N$

Ef: $\exists i \in [1..12] : (h = \text{hónév}[i])$

Uf: $1 \leq s \leq 12$ és $\text{HÓNÉV}[s] = h$

Példa:

$h=9 \rightarrow s=\text{"szeptember"}$

HÓNÉV

1	január
2	február
3	március
4	április
5	május
6	június
7	július
8	augusztus
9	szeptember
10	október
11	november
12	december

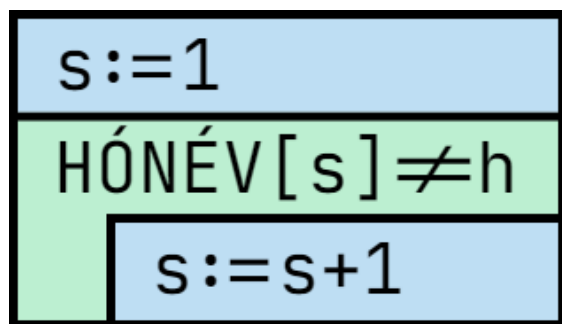
$h \in \text{hónév}$

Konstans tömb alkalmazása

Algoritmus:

Változó h :Szöveg, s :Egész

Konstans $HÓNÉV:Tömb[1..12: Szöveg] =$
("január", ..., "december")



Be: $h \in S$,
 $hónév \in S[1..12] =$
["január", ..., "december"]
Ki: $s \in N$
Ef: $\exists i \in [1..12] : (h = hónév[i])$
Uf: $1 \leq s \leq 12$ és $HÓNÉV[s] = h$

Kérdés: mi lenne, ha az előfeltétel nem teljesülne?
Futási hiba? Végtelen ciklus?

	HÓNÉV
1	január
2	február
3	március
4	április
5	május
6	június
7	július
8	augusztus
9	szeptember
10	október
11	november
12	december

Konstans tömb – mit tárolunk?

Példa:

$h=9, n=18 \rightarrow s=261$

Feladat:

Egy nap a nem szökőév hányadik napja?

Specifikáció₁:

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

$HÓ \in \mathbb{N}[1..12] = [31, 28, 31, \dots, 31]$

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

Ef: $1 \leq h \leq 12$ és $1 \leq n \leq HÓ[h]$

Uf: $s = \text{SZUMMA}(i=1..h-1, HÓ[i]) + n$

	HÓ
1	0
2	31
3	59
4	90
5	120
6	151
7	181
8	212
9	243
10	273
11	304
12	334

Példa:

$h=9, n=18 \rightarrow s=261$

Konstans tömb – mit tárolunk?

Algoritmus:

Változó h, n, s : Egész

Konstans $HÓ: Tömb[1..12: Egész] = (31, 28, 31, \dots, 31)$

Local declarations

Változó i : Egész

$s := 0$

$i = 1 \dots h-1$

$s := s + HÓ[i]$

$s := s + n$

Programparaméterek
deklarálása

Lokális változó
deklarálása

Be: $h \in \mathbb{N}, n \in \mathbb{N},$
 $HÓ \in [1..12] = [31, 28, 31, \dots, 31]$
Ki: $s \in \mathbb{N}$
Ef: $1 \leq h \leq 12$ és $1 \leq n \leq HÓ[h]$
Uf: $s = SZUMMA(i=1..h-1, HÓ[i]) + n$

HÓ

1	0
2	31
3	59
4	90
5	120
6	151
7	181
8	212
9	243
10	273
11	304
12	334

Megjegyzés: szökőév esetén $h \geq 3$ esetén s -et 1-gyel meg kellene növelni! (És az előfeltétel is módosul.)

Konstans tömb – mit tárolunk?

Egy másik megoldás:

Példa:
 $h=9, n=18 \rightarrow s=261$

Tároljuk minden hónapra, hogy az előző hónapokban összesen hány nap van!

Specifikáció₂:

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

$HÓ \in S[1..12] = (0, 31, 59, 90, \dots, 334)$

Uf: $s = HÓ[h] + n$

Kérdés: Ez jobb megoldás? Mi lesz az előfeltétellel?

	HÓ
1	0
2	31
3	59
4	90
5	120
6	151
7	181
8	212
9	243
10	273
11	304
12	334

Összefoglalás



Összefoglalás

- Adat
 - egy-szerű: elemi
 - több különböző: rekord
 - több egyforma: tömb
- Vezérlési szerkezetek
 - Szekvencia: és
 - Elágazás: $->$
 - Ciklus: \forall, \exists, Σ
- Feladatmegoldás
 1. Példa
 2. Specifikáció (\leftarrow példa)
 1. Adatok (Be, Ki)
 2. Megszorítás ($Ef \rightarrow Be$)
 3. Összefüggés (Uf)
 3. Adat \rightarrow Változó
 4. Algoritmus ($\leftarrow Uf$)
 5. Kód ($\leftarrow Spec + Alg$)

Megfeleltetések

Példa adat	Specifikáció halmaz	Algoritmus típus	Kód type
3	N	Egész	int
-3	Z	Egész	int
3,3	R	Valós	double
igaz	L	Logikai	bool
"alma"	S	Szöveg	string
"a"	K	Karakter	char
(név:"Győző", jegy: 5)	Név x Jegy, S x N	Rekord	struct
[3, 5, -6, 2]	Z[1..n]	Tömb	int[]

Ellenőrző kérdések



Ellenőrző kérdések

- Mikor érdemes rekordtípust használni?
- Milyen adatszerezettel írjuk le, ha több különböző funkciójú adatot szeretnénk egységbe foglalni?
- Milyen adatszerkezettel írjuk le, ha több ugyanolyan funkciójú adatot szeretnénk egységbe foglalni?
- Hogyan hívjuk a tömböt a specifikációban?