Programozás C nyelven (4. ELŐADÁS)

Sapientia EMTE

2021-22

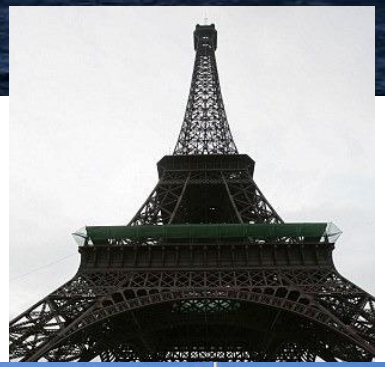


while vs. for


```
int szam, s;
scanf("%i", &szam);
s = 0;
while ( szam > 0 ){
   s += szam%10;
   szam /= 10;
}
printf("szamjegyosszeg: %i" ,s);
```

```
int szam, s;
scanf("%i", &szam);
for ( s = 0 ; szam > 0 ; szam /= 10 ){
   s += szam%10;
}
printf("szamjegyosszeg: %i" ,s);
```

Hány féle „téglából” építkezünk?



EMLÉKEZTETŐ / KIEGÉSZÍTŐ

Strukturált programozás

I. SZEKVENCIA

- <kifejezés>;(1)
  - értékadás

II.ELÁGAZÁS

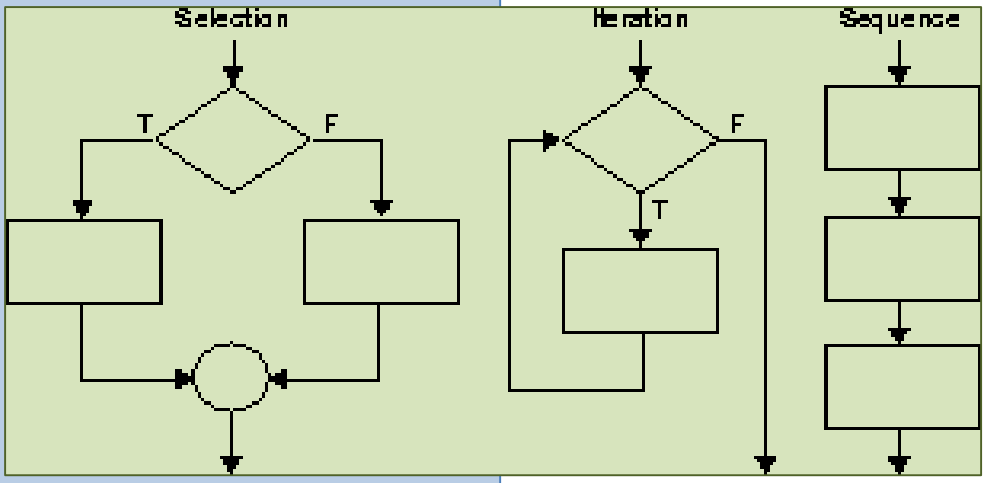
- if-else (2)
- switch (3)

III.CIKLUSOK

- for (4)
- while (5)
- do-while (6)
  - break(7), continue(8), goto(9)

```
if (5 == x);
else {
    a = 1; b = 2;
}
```

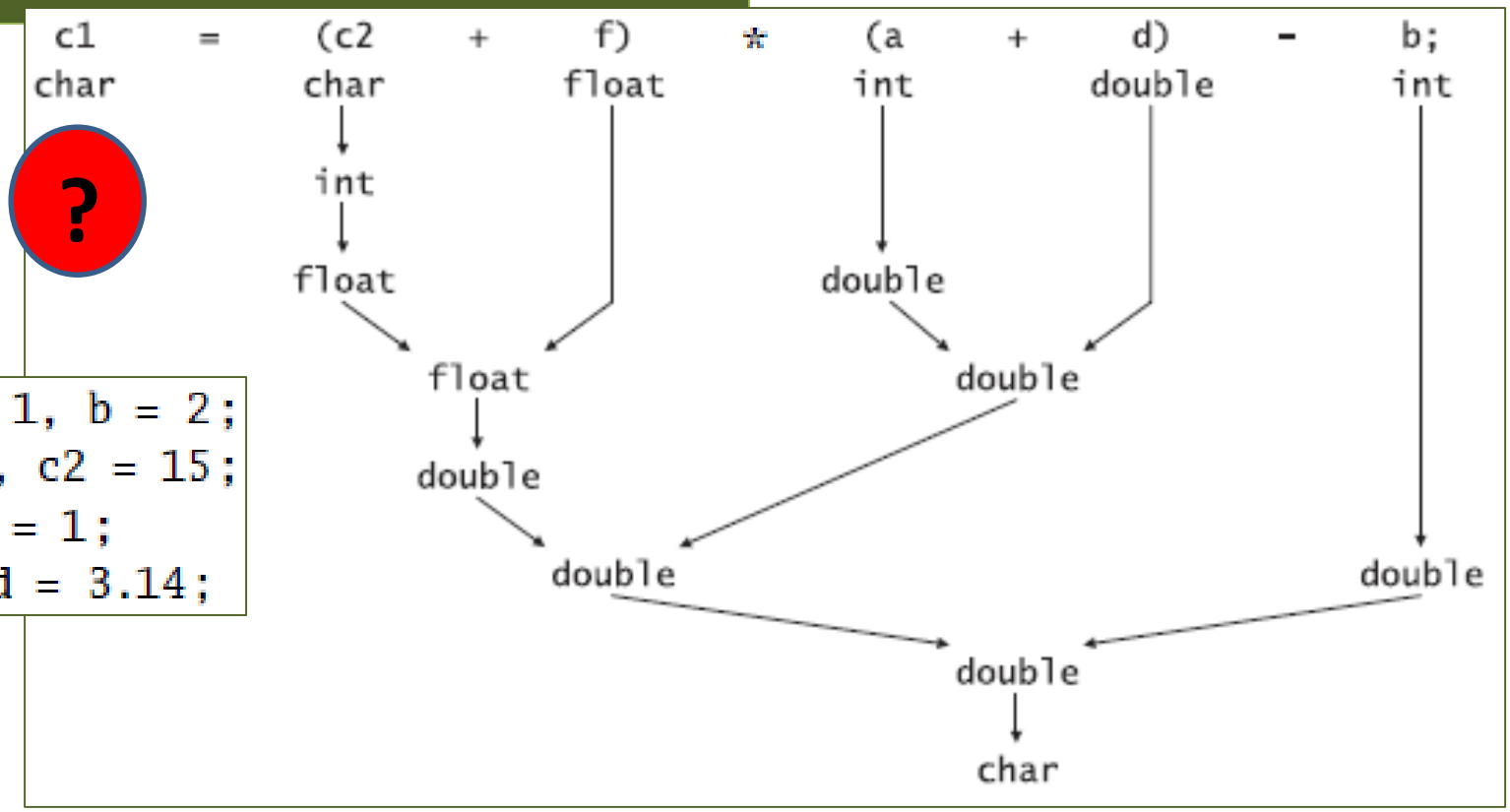
összetett/üres utasítás



? Hány utasításból „építkezünk” ?

KIFEJEZÉSEK

- **operátorok / operandusok**
  - **prioritás-sorrend**
  - **konverziók**



?

```
int a = 1, b = 2;
char c1, c2 = 15;
float f = 1;
double d = 3.14;
```

OPERÁTOROK

Értékadás
<változó> = <kifejezés>

operandusok / operátorok

```
x1 = (-b + sqrt(delta)) / (2*a);
```

```
a = b = c = 0;
```

Jobbról-balra

```
sizeof (<kifejezés>)
sizeof (<típus>)
```

```
sizeof (short)
```

```
short x; double y;
sizeof (x + y)
```

Implicit típuskonverzió

Aritmetikai operátorok
+, -, *, /, %

Összevont operátorok
+=, -=, *=, /=, %=

```
s += x;
```

```
x /= 10;
```

```
int i = 1, j = 1, x, y;
x = ++i; y = j--;
```

?

OPERÁTOROK

Összehasonlítási operátorok

`==, !=, <, <=, >, >=`

Logikai operátorok

`!, &&, ||`

```
while( i < n && j < m ){…}
```

CAST operátor

`(<tipus>)<operandus>`

```
double x = 3.14;
printf("%i",(int)x);
```

?

```
int x = 3;
printf("%f",(float)x / 2);
```

Feltételes operátor

`< > ? < > : < >`

```
max = a > b ? a : b;
```

Vessző operátor

$<kif_1>,<kif_2>,...,<kif_n>$

```
for ( i=1,j=n ; i<j ; ++i,--j ){…}
```

OPERANDUSOK

- **Változók**
- **Konstansok**
- **Függvényhívások**

```
x1 = (-b + sqrt(delta)) / (2*a);
```

```
0, 7, 19, 25432 (10-es számrendszerben)
0, 013, 0257 (8-as számrendszerben)
0x1a ,0x234, 0XAB2F (16-os számrendszerben)
```

65 ⟺ 0101 ⟺ 0X41

```
1   - int típusú 1-es
1U  - unsigned int típusú 1-es
1L  - long int típusú 1-es
1UL - unsigned long int típusú 1-es
```

```
3.14, -17.65, 1., 1.0, 0.1, .1, -0.025e-1,-12.6E2
-1.F
0.01E+5L
```

```
'A', 'a', '0', '9', '!', '+', ' '
'\ooo' '\xhh'
```

'\a'	BEL	csengő
'\b'	BS	visszalépés
'\f'	FF	lapdobás
'\n'	LF	új sor (soremelés)
'\r'	CR	kocsi vissza
'\t'	HT / TAB	vízszintes tabulátor
'\v'	VT	függőleges tabulátor
'\\'	\	backslash
'\''	'	aposztróf
'\"'	"	idézőjel
'\?'	?	kérdőjel

```
"Alma"
"két sorba tört hosszú\
 karakterlánc konstans"
"a kettő" "egy lesz"
```

```
#define PI 3.14
INT_MAX
#include <limits.h>
```

Találós kérdés (1)

- Mi egy közös vonás a halloween-ben és a karácsonyban?

Találós kérdés (2)

- Mi egy közös vonás a halloween-ben és a karácsonyban?

Október 31

December 25

Találós kérdés (3)

- Mi egy közös vonás a halloween-ben és a karácsonyban?

OKT
31

DEC
25

Találós kérdés (4)

- Mi egy közös vonás a halloween-ben és a karácsonyban?

OKT

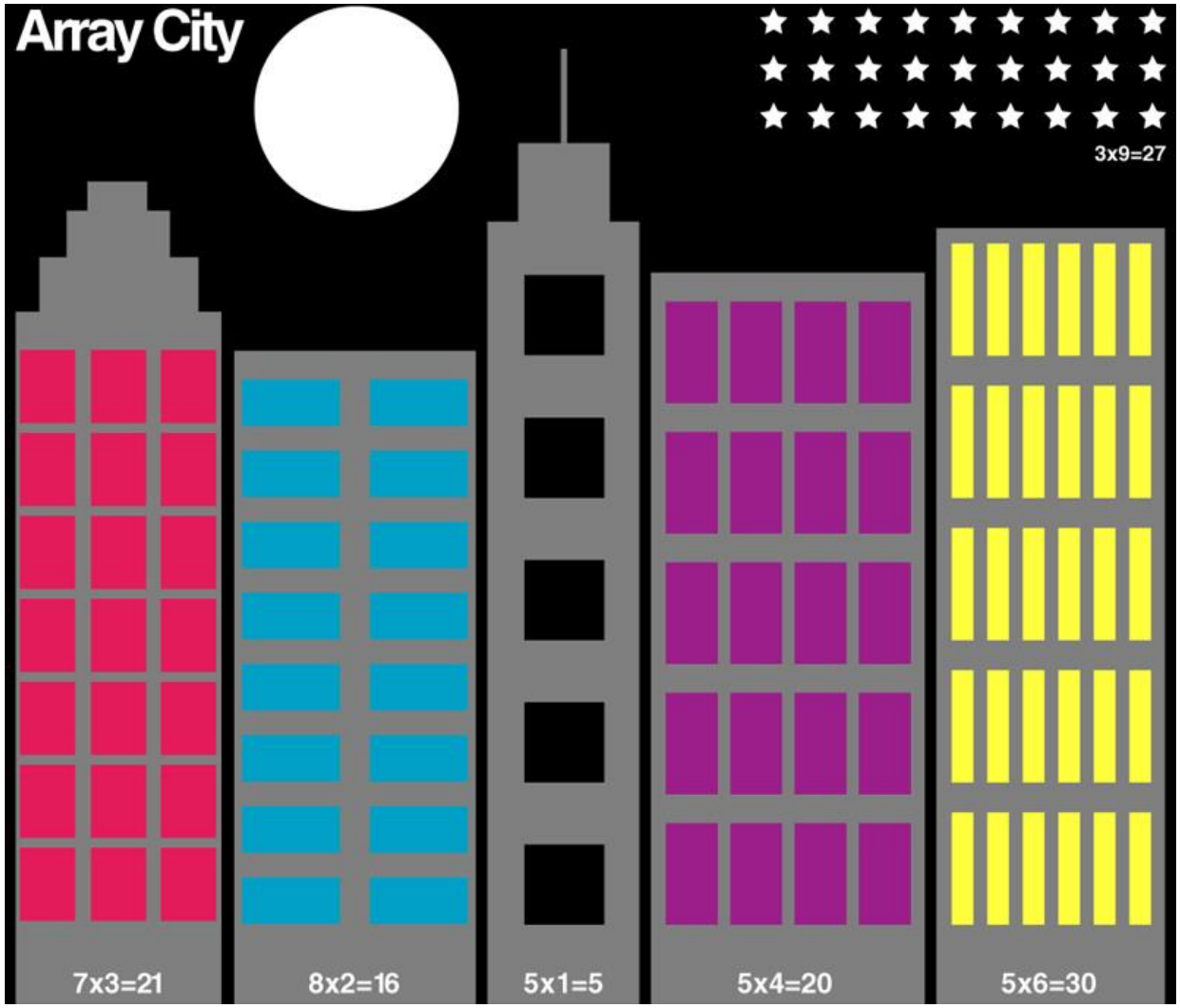
31

DEC

25

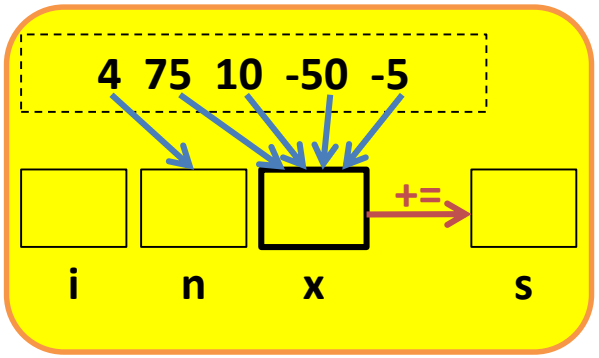
```
25 : 8 = 3 : 8 = 0
24       0
--       -
=1       3
```

TÖMBÖK



Szám-sorozat *egy* változóban

Adott egy n elemű számsorozat, számítsuk ki az elemek összegét.




```
int i,n,x,s=0;
scanf("%i", &n);
for ( i=1 ; i<=n ; ++i ){
    scanf("%i",&x); s += x;
}
printf("s= %i", s);
```

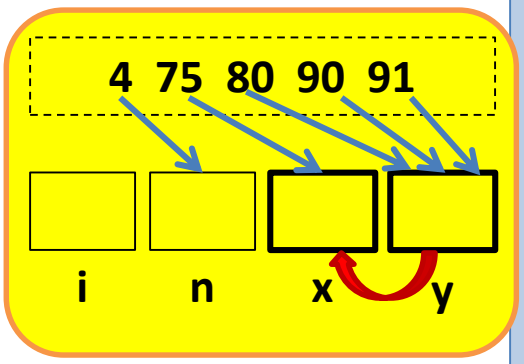
```
4
75
10
-50
-5
s = 30_
```

WATCH

x	s
?	0
75	75
10	85
-50	35
-5	30

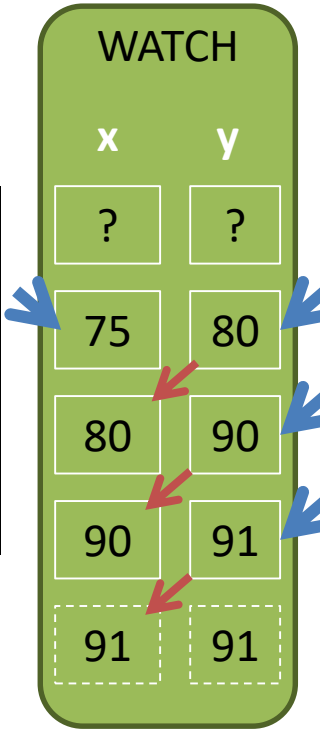
Szám-sorozat *két* változóban

Adott egy n (n>1) elemű számsorozat, ellenőrizzük, hogy növekvő-e.



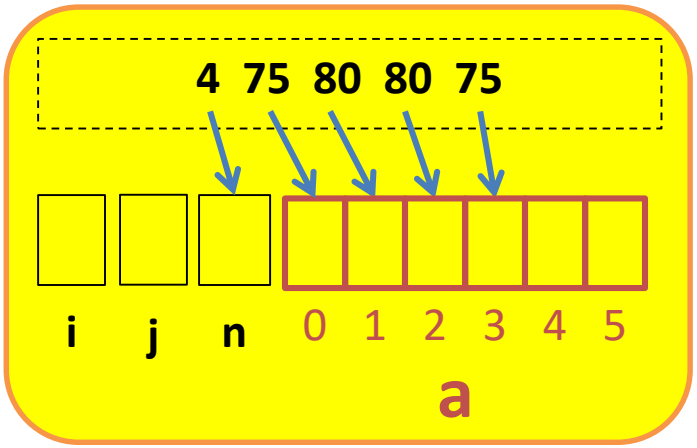

```
int i,n,x,y;
scanf("%i", &n);
scanf("%i", &x);
for ( i=2 ; i<=n ; ++i ){
    scanf("%i", &y);
    if(y < x) {break;}
    x = y;
}
if (i<=n) { printf("NEM"); }
else { printf("IGEN"); }
```

```
4
75
80
90
91
IGEN_
```



Szám-sorozat *tömb* változóban

Adott egy n (n>1) elemű számsorozat, ellenőrizzük, hogy tükörsorozat-e.




```
int i,j,n,a[6];
scanf("%i", &n);
for ( i=0 ; i<n ; ++i ){
   scanf("%i", &a[i]);
}
for ( i=0,j=n-1 ; i<j ; ++i,--j ){
   if(a[i] != a[j]) {break;}
}
if (i<j) { printf("NEM"); }
else { printf("IGEN"); }
```

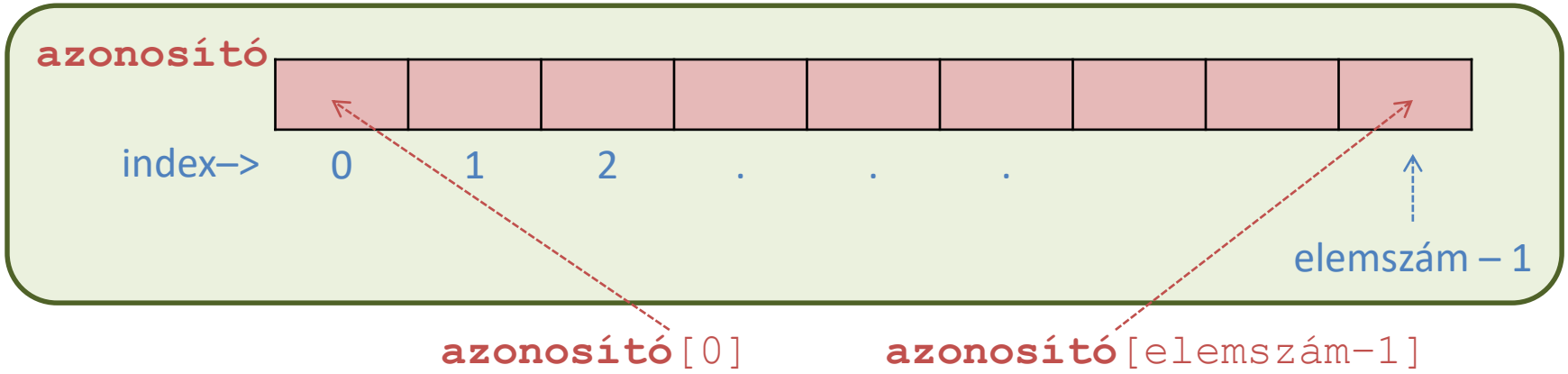
```
4
75
80
80
75
IGEN_
```

TÖMB – típus

Definiálás:

Egész típusú

```
<elemtípus> <azonosító>[<elemszám>];
```



Példák:

```
int a[100]; //elemei a[0], a[1], …, a[99]
double x[10], y, z[50]; // sizeof(x)=?
short b[5]={11,22,33,44,55}; //b[0]=?, b[4]=?
long c[1000]={0}, d[50]={1}; //c[999]=?, d[49]=?
char s[5]={'a','b','c','d','e','f'}; //???
int w[]={1,2,3,4,5,6,7,8,9,10}; // sizeof(w)=?
```

Számsorozat beolvasása egydimenziós tömbbe

```
int main(){
    freopen("szamsor.txt", "r", stdin);
    int n, a[100];
    scanf("%i", &n);
    for(int i = 0 ; i < n ; ++i){  // generálja i-ben az
        scanf("%i", &a[i]);        // 0,1,...,n-1 index-sort
    }
    ooo
    return 0;
}
```

C99
```
int n;
scanf("%i",&n);
int a[n];
```

C99
i csak a for-on belül ismert

szamsor.txt

```
6
44 5 13 7 -10 11
```

44	5	13	7	-10	11	?	...	?
0	1	2	3	4	5	6		99

Számsorozat kiírása egydimenziós tömbből

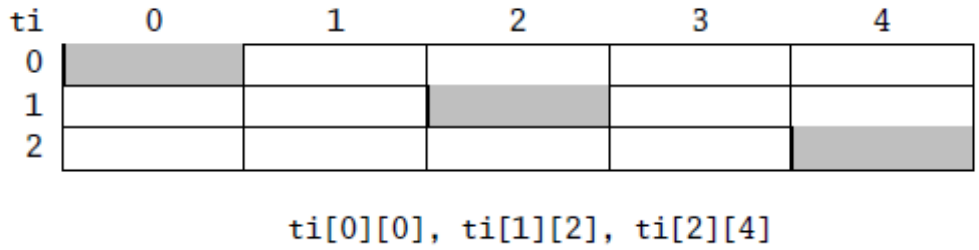
```
int main(){
   ooo
   for(int i = 0 ; i < n ; ++i){
      printf("%i ", a[i]);
   }
   ooo
   return 0;
}
```

Kétdimenziós tömbök

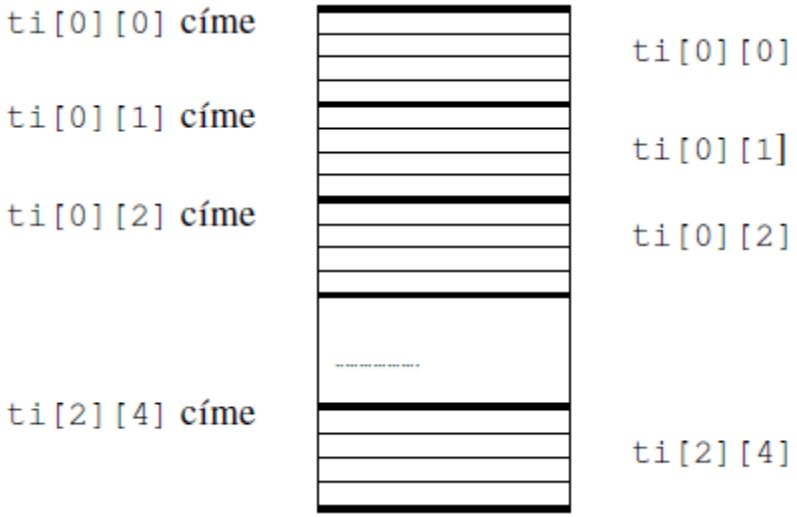
```
<elem_típus> <név>[<sorok_száma>][<oszlopok_száma>];
```

```
char tc[20][20];
int ti[3][5];
long double tld[10][10];
```

```
int a[2][3]={1, 3, -5, 0, 77, -13};
int b[][3]={1, 3, -5, 0, 77, -13, 9, 1, 34};
```



ti[0][0], ti[1][2], ti[2][4]




```
<elem_típus> <név>[<méret_1>][<méret_2>]...[<méret_n>];
```

```
<név>[<index_1>][<index_2>]...[<index_n>]
```

Mátrix beolvasása kétdimenziós tömbbe

matrix.txt

```
2 3
44 5 13
7 -10 11
```

```
int main(){
    freopen("matrix.txt", "r", stdin);
    int n, m, b[100][100];
    scanf("%i%i", &n, &m);
    for(int i = 0 ; i < n ; ++i){          // generálja (i,j)-ben az
        for(int j = 0 ; j < m ; ++j){      // (0,0),(0,1),…,(0,m-1 ),(1,0),…,(n-1,m-1)
            scanf("%i", &b[i][j]);         // indexpár-sort
        }
    }

    ooo

    return 0;
}
```

	0	1	2	3	…	99
0	44	5	13	?	…	?
1	7	-10	11	?	…	?
2	?	?	?	?	…	?
…	…					
99	?	?	?	?	…	?

Mátrix kiírása kétdimenziós tömbből

```
int main(){
    ooo
    for(int i = 0 ; i < n ; ++i){
        for(int j = 0 ; j < m ; ++j){
            printf("%i ", b[i][j]);
        }
        printf("\n");
    }
    ooo
    return 0;
}
```

	0	1	2	3	…	99
0	44	5	13	?	…	?
1	7	-10	11	?	…	?
2	?	?	?	?	…	?
…	…					
99	?	?	?	?	…	?

```
44 5 13
7 -10 11

_
```

Mátrix bejárása oszloponként

```
int main(){
    ooo
    for(int j = 0 ; j < m ; ++j){
        for(int i = 0 ; i < n ; ++i){
            printf("%4i", b[i][j]);
        }
        printf("\n");
    }
    ooo
    return 0;
}
```

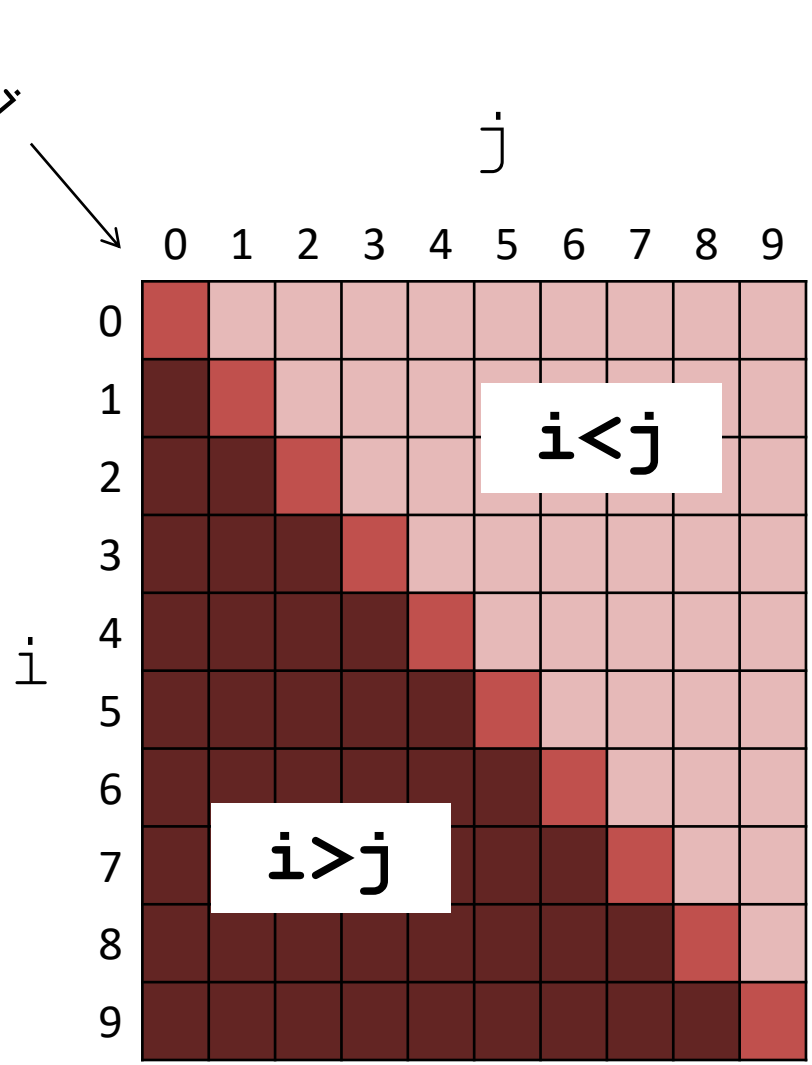
	0	1	2	3	…	99
0	44	5	13	?	…	?
1	7	-10	11	?	…	?
2	?	?	?	?	…	?
…	…					
99	?	?	?	?	…	?

```
  44   7
   5 -10
  13  11

_
```

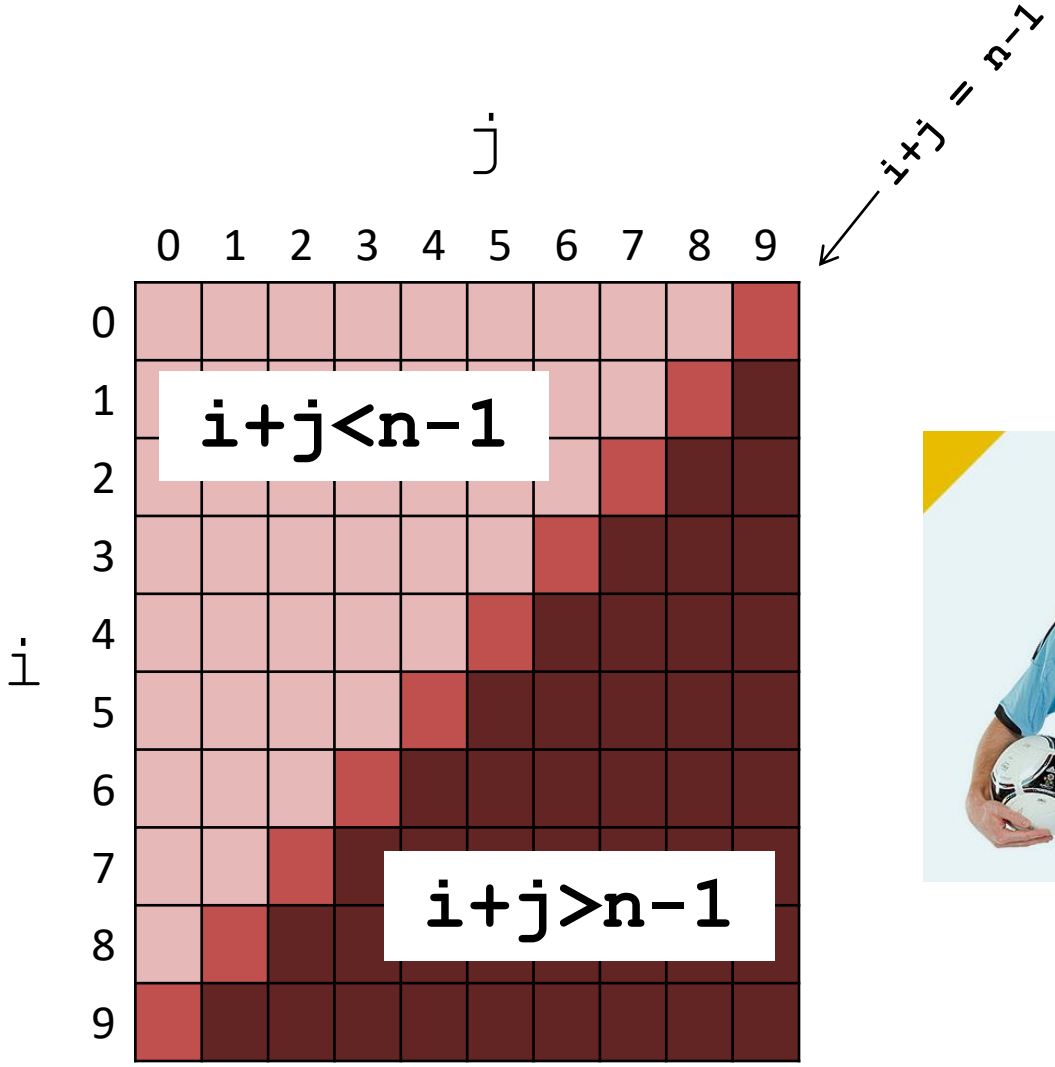
Mozogjunk otthonosan négyzetes mátrixokban

főátlón / alatta / felette



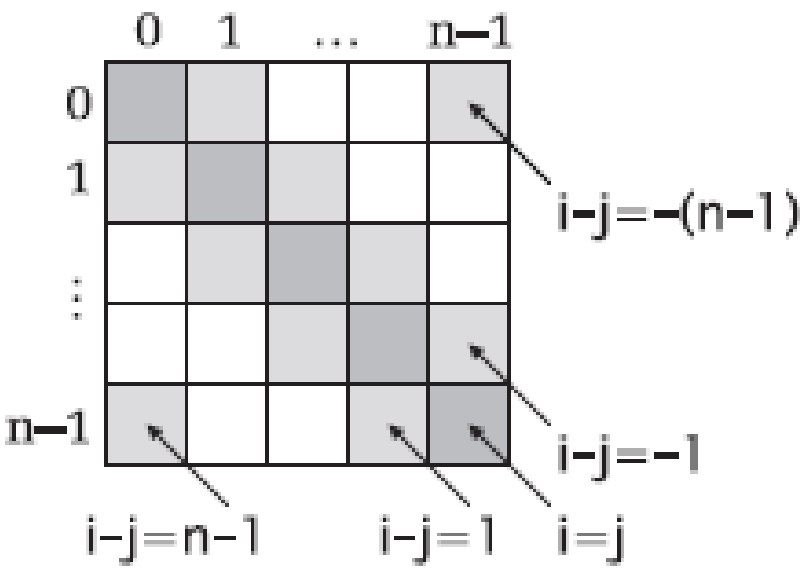
Mozogjunk otthonosan négyzetes mátrixokban

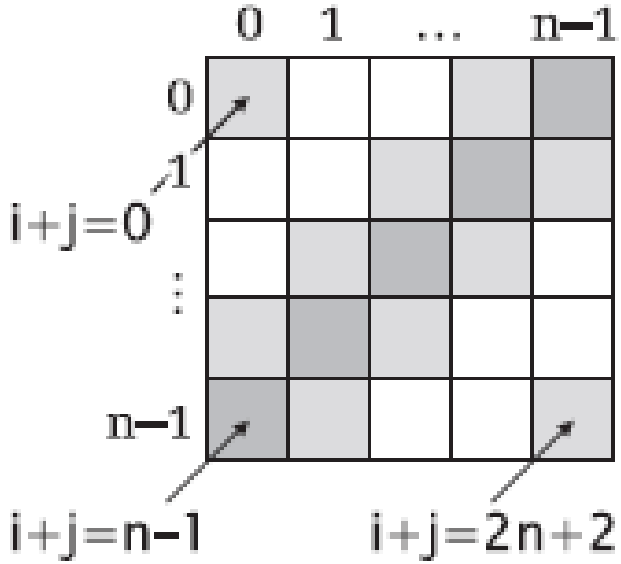
mellékátlón / alatta / felette



Mozogjunk otthonosan négyzetes mátrixokban

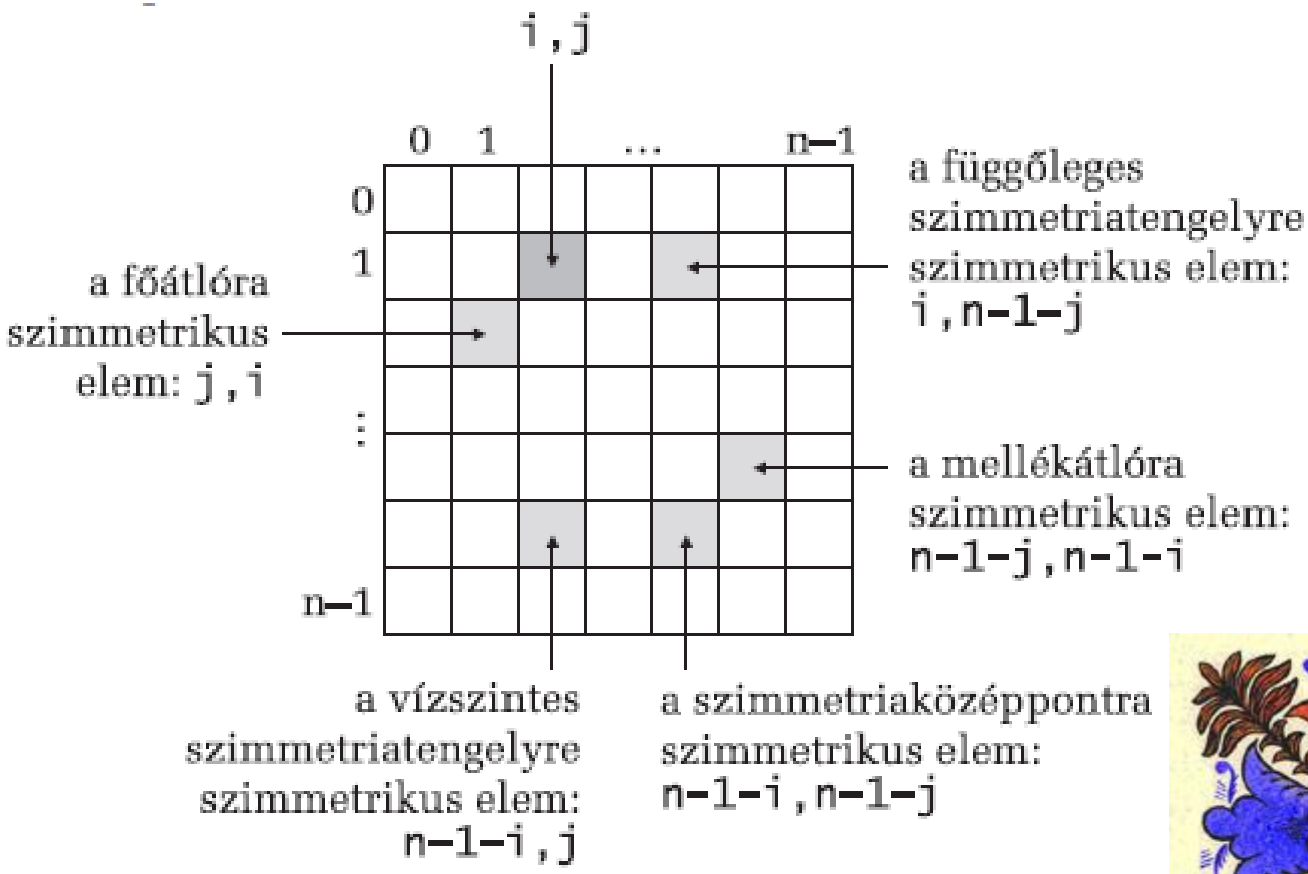
főátlóval/mellékátlóval *párhuzamos* átlókon





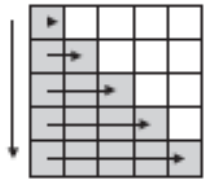
Mozogjunk otthonosan négyzetes mátrixokban

szimmetriapontok

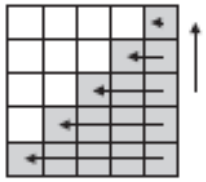


Mozogjunk otthonosan négyzetes mátrixokban

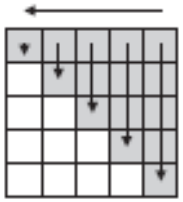
háromszögekben sétafikálva



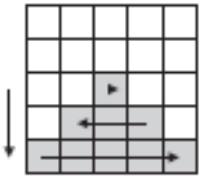
```
for(i=0; i<n; i++)
    for(j=0; j<=i; j++)
```



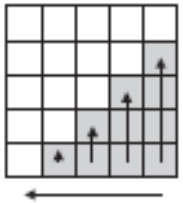
```
for(i=n-1; i>=0; i--)
    for(j=n-1; j>=n-1-i; j--)
```



```
for(j=n-1; j>=0; j--)
    for(i=0; i<=j; i++)
```



```
for(i=n/2; i<=n-1; i++)
    for(i%2?j=i:j=n-1-i;
        j>=n-1-i && j<=i;
        i%2?j--:j++)
```



```
for(j=n-1; j>0; j--)
    for(i=n-1; i>n-1-j; i--)
```

Összefoglalás

- 1-dimenziós tömbök (számsorok tárolására)
  - **<elemtípus> <név>[<elemszám>];**
    - `int a[100], b[1000]={0}, c[]={1,2,3};`
    - `for(int i=0 ; i<n ; ++i){… a[i] …}`

- 2-dimenziós tömbök (mátrixok tárolására)
  - **<elemtípus> <név>[<sorszám>][<oszlopszám>];**
    - `long x[50][100], y[][3]={1,2,3,4,5,6};`
    - ```
      for(int i=0 ; i<n ; ++i){
        for(int j=0 ; j<m ; ++j){
          … x[i][j] …
        }
      }
      ```


      ```