Programozás C nyelven
(2. ELŐADÁS)

Sapientia EMTE

2020-21

EMLÉKEZTETŐ / KIEGÉSZÍTŐ

- ALAP-típusok (C89):
  - `char, int, float, double`
- **`<típus> <azonosító> [=<kezdőérték>];`**
  - `int x;`
  - `long x1, x2=1000000, x3;`
  - `char a, b='B', c=1;`
- IN/OUT (C)
  - `scanf("%i", &x);`
  - `printf("Osszeg: %i\n", x+y);`

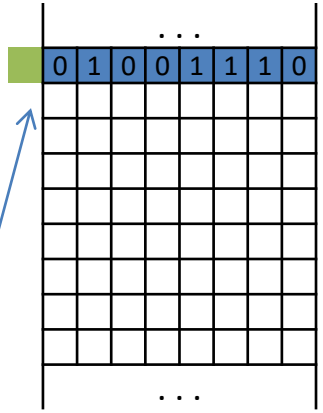
ASCII tábla:

…
10. **LF ('\n')**
…
13. **CR**
…
27. **ESC**
…
32. **<SPACE>**
…
48. **'0'**
…
65. **'A'**
…
97. **'a'**
…

EMLÉKEZTETŐ / KIEGÉSZÍTŐ

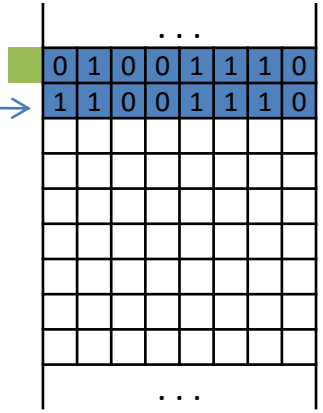
`sizeof(char) = 1; [-2^7 .. 2^7-1]`

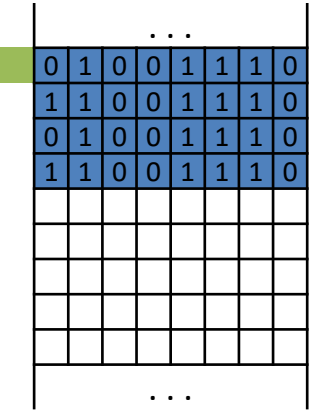
`sizeof(unsigned char) = 1; [0 .. 2^8-1]`



`sizeof(short) = 2; [-2^15 .. 2^15-1]`

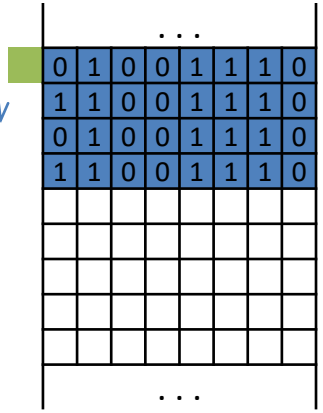
`sizeof(unsigned short) = 2; [0 .. 2^16-1]`





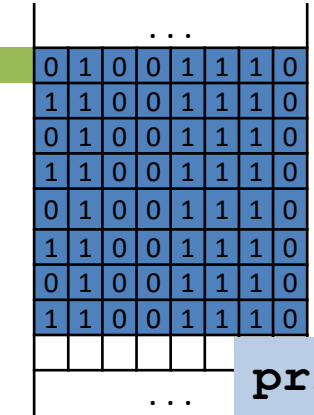
`sizeof(long) = 4; [-2^31 .. 2^31-1]`

`sizeof(unsigned long) = 4; [0 .. 2^32-1]`



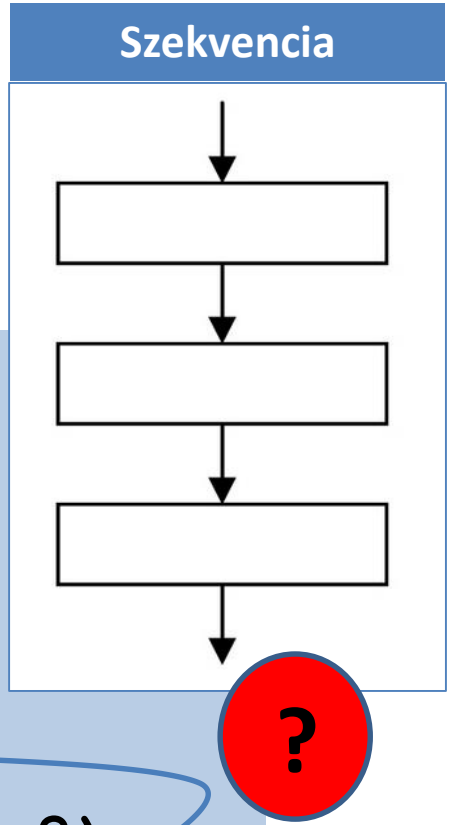
`sizeof(float) = 4;`

`sizeof(double) = 8;`




```
printf("%i%i%i",sizeof(char),sizeof(short),sizeof(long));
```

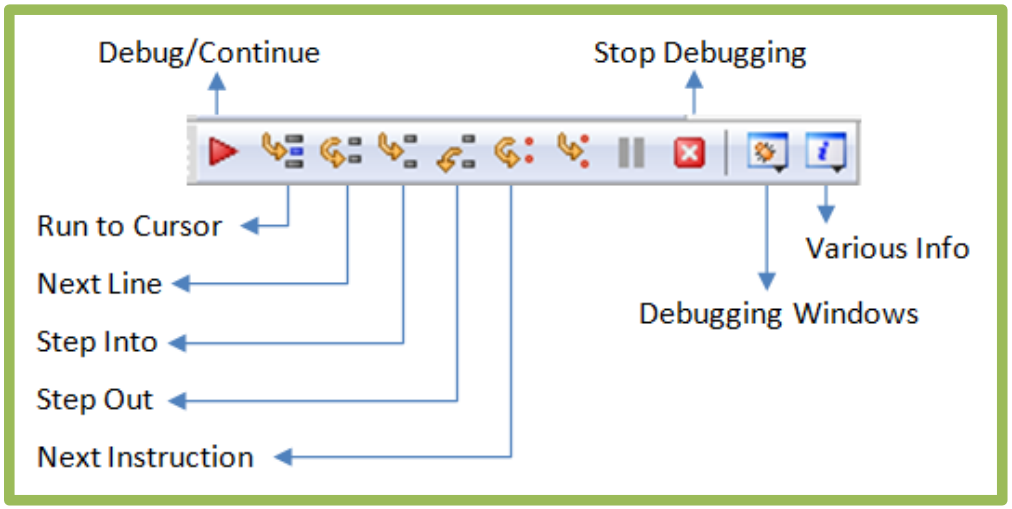
EMLÉKEZTETŐ / KIEGÉSZÍTŐ




```
double a,b,c,delta,x1,x2;
scanf("%lf%lf%lf", &a, &b, &c);
delta = b*b - 4*a*c;
x1 = (-b + sqrt(delta)) / (2*a);
x2 = (-b - sqrt(delta)) / (2*a);
printf("Gyokok: %.2lf, %.2lf", x1, x2);
```

```
1 -5 6
_
```

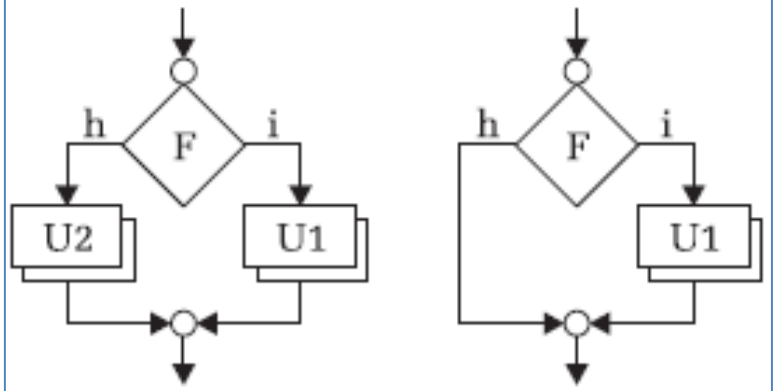
```
1 -5 6
Gyokok: 3.00, 2.00_
```



„Párhuzamos forgatókönyvek”

Elágazás (if-else)

Elágazás




```
if (<feltétel>) {
  <utasítások>
}
else {
  <utasítások>
}
```

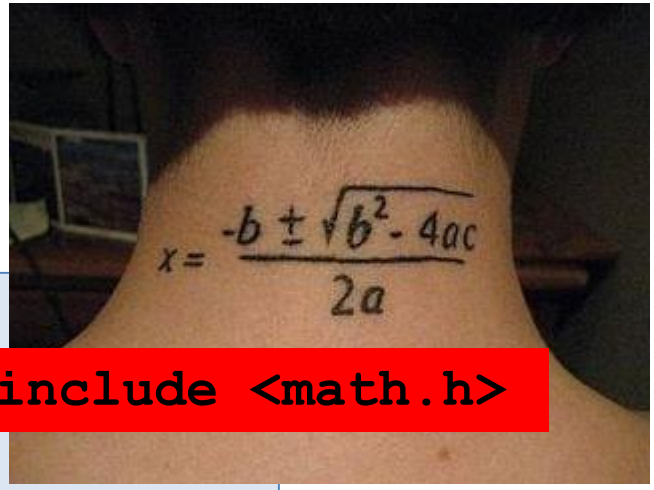
Összehasonlítási operátorok:

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

```
scanf("%lf%lf%lf",&a,&b,&c);
if (a != 0){
   ooo
}
else{
   ooo
}
```

```
scanf("%lf%lf%lf", &a, &b, &c);
if (a != 0){
   delta = b*b - 4*a*c;
   if (delta >= 0){
        ooo
   }
   else{
        ooo
   }
}
else{
   ooo
}
```

Elágazás (`if-else`)




```
double a,b,c,delta,x1,x2;
scanf("%lf%lf%lf", &a ,&b, &c);
if (a != 0){
   delta = b*b - 4*a*c;
   if (delta >= 0){
        x1 = (-b + sqrt(delta)) / (2*a);
        x2 = (-b - sqrt(delta)) / (2*a);
        printf("Gyokok: %.2lf, %.2lf", x1, x2);
   }
   else{
        printf("Komplex gyokok!");
   }
}
else{
   printf("Nem masodfoku!");
}
```

```
1 -5 6
Gyokok: 3.00, 2.00_
```

```
0 -5 6
Nem masodfoku!_
```

?

```
1 -5 7
Komplex gyokok!_
```

Logikai típus / operátorok


```
printf("%i",!!2020);
```

- C89: minden nem nulla numerikus érték logikai értéke IGAZ, a nulla pedig HAMIS. Az IGAZ-nak / HAMIS-nak megfelelő numerikus érték 1/0.
  - C99: _Bool, **stdbool.h** (**bool, true/false**)

- Operátorok: **!** (NEM), **&&** (ÉS), **||** (VAGY)

- FELADAT: *Adott 3 egész szám a billentyűzetről. Írassunk ki egy megfelelő üzenetet aszerint, hogy van-e köztük 0.*

C89 – megoldások

?

```
int x1,x2,x3;
scanf("%i%i%i", &x1, &x2, &x3);
if (0 == x1 || 0 == x2 || 0 == x3){
   printf("VAN");
}
else{
   printf("NINCS");
}
```

```
1 0 6
VAN_
```

```
int x1,x2,x3;
scanf("%i%i%i",%x1,&x2,&x3);
if (x1 && x2 && x3){
   printf("NINCS");
}
else{
   printf("VAN");
}
```

?

```
1 -5 6
NINCS_
```

a	b	a && b	a || b
false	false	false	false
false	**true**	false	**true**
true	false	false	**true**
true	**true**	**true**	**true**

C99 – megoldás

```
1 -5 6 0
VAN_
```

```
1 -5 6 33
NINCS_
```

```
0 0 0 77
VAN_
```

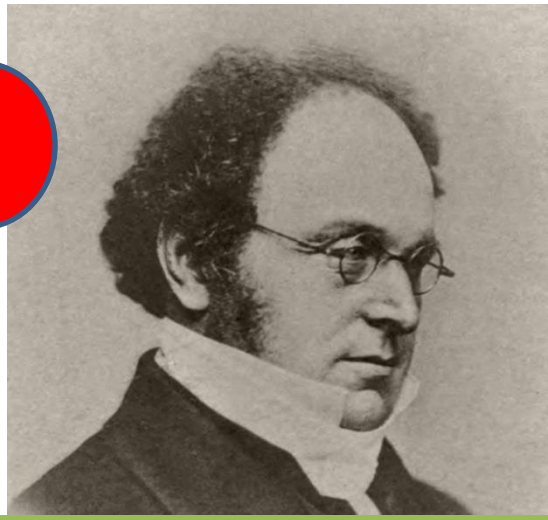
```
#include <stdbool.h>
. . .
int x;
bool b = false;
scanf("%i",&x); if (!x) {b = true;}
scanf("%i",&x); if (!x) {b = true;}
scanf("%i",&x); if (!x) {b = true;}
scanf("%i",&x); if (!x) {b = true;}
if (b) { printf("VAN"); }
else { printf("NINCS"); }
```

else !

?

Feltételes operátor

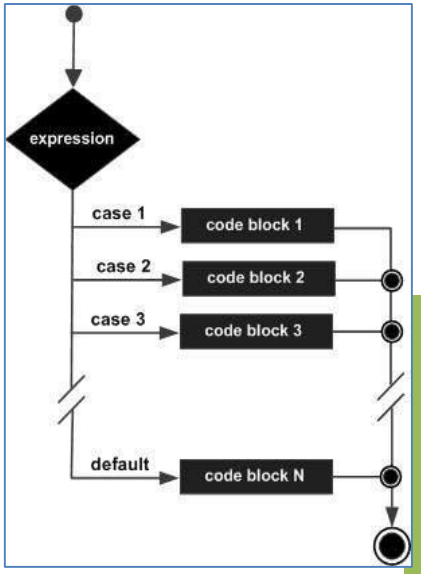
```
<kif1> ? <kif2> : <kif3>

printf( b ? "VAN" : "NINCS" );
```



De Morgan képletek:

$!(a \&\& b) \Leftrightarrow !a \,||\, !b$

$!(a \,||\, b) \Leftrightarrow !a \&\& !b$



Szelektor: egész típusú

Többszörös elágazás (switch)

```
switch (<kifejezés>){
    case <konstans_1> : <utasítások>; [break;]
    …
    case <konstans_n> : <utasítások>; [break;]
    [ default <utasítások> ]
}
```

```
int jegy;
scanf("%d", &jegy);
if(jegy == 1) printf("Elégtelen");
    else if(jegy == 2) printf("Elégséges");
        else if(jegy == 3) printf("Közepes");
            else if(jegy == 4) printf("Jó");
                else if(jegy == 5) printf("Jeles");
                    else printf("Téves beolvasás");
```

```
int n;
scanf("%i", &n);
switch (n){
   case 5 : printf("*");
   case 4 : printf("*");
   case 3 : printf("*");
   case 2 : printf("*");
   case 1 : printf("*");
}
```

?

```
int jegy;
scanf("%i", &jegy);
switch (jegy){
   case 5 : printf("jeles"); break;
   case 4 : printf("jo"); break;
   case 3 : printf("kozepes");break;
   case 2 : printf("elegseges");break;
   case 1 : printf("elegtelen");break;
   default printf("nem jegy");
}
```

?

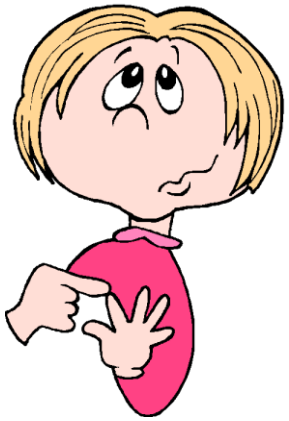
Határozzuk meg 5 szám maximumát!

WATCH	
x	**max**
?	?
?	3
-5	3
6	6
4	6
5	6

```
int x, max;
scanf("%i", &max);
scanf("%i",&x);if(x > max){max = x;}
scanf("%i",&x);if(x > max){max = x;}
scanf("%i",&x);if(x > max){max = x;}
scanf("%i",&x);if(x > max){max = x;}
printf("MAX = %i", max);
```

```
3
-5
6
4
5
MAX = 6_
```

Számoljuk meg hány páratlan!



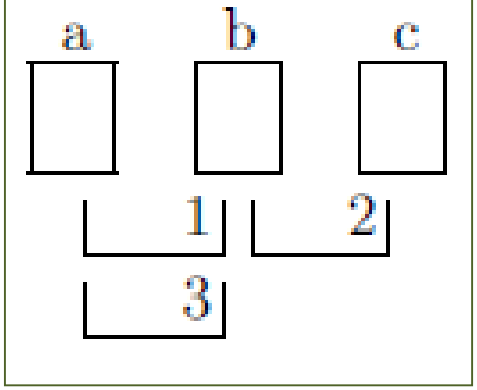
WATCH	
x	**db**
?	0
3	1
55	2
6	2
4	2
5	3

```
int x, db = 0;
scanf("%i", &x); if (x % 2) {++db;}
scanf("%i", &x); if (x % 2) {++db;}
scanf("%i", &x); if (x % 2) {++db;}
scanf("%i", &x); if (x % 2) {++db;}
scanf("%i", &x); if (x % 2) {++db;}
printf("DARAB = %i", db);
```

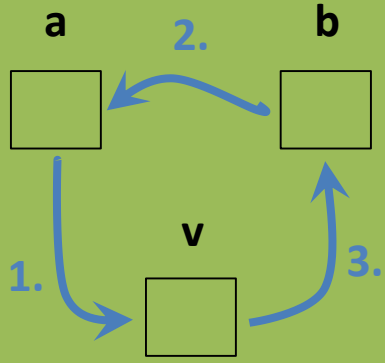
```
3
55
6
4
5
DARAB = 3_
```

Rendezzünk növekvő sorrendbe!

Buborékos rendezés



három-veder módszer



WATCH

x1	x2	x3
?	?	?
3	7	1
3	7	1
3	1	7
1	3	7

```
int x1, x2, x3, v;
scanf("%i%i%i", &x1, &x2, &x3);
if (x1 > x2) {v = x1; x1 = x2; x2 = v;}
if (x2 > x3) {v = x2; x2 = x3; x3 = v;}
if (x1 > x2) {v = x1; x1 = x2; x2 = v;}
printf("%i, %i, %i", x1, x2, x3);
```

```
3 7 1
1, 3, 7_
```

ISMÉTLÉS

- **SZEKVENCIA**
- **ELÁGAZÁS**
  - `if-else`
  - `< > ? < > : < >`
  - `switch`

Összehasonlítási operátorok

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

C99: `stdbool.h`

`bool, true/false`

Logikai operátorok

`!, &&, ||`

- **Létezés-ellenőrzés technikája**
- **Max-keresés technikája**
- **Számlálás technikája**
- **Rendezés technikája**

RÁADÁS (1)

Típus	N	Értéktartomány
unsigned char	8	$0 \ldots 2^8 - 1$
unsigned short int	16	$0 \ldots 2^{16} - 1$
unsigned long int	32	$0 \ldots 2^{32} - 1$

- Előjel nélküli egészek (unsigned) fix-pontos ábrázolása N biten

Ábrázolási algoritmus:
(Legyen x az ábrázolandó természetes szám; $0 \leq x \leq 2^N - 1$)
1. Átalakítjuk x-et kettes számrendszerbe.
2. Kipótoljuk x bináris alakját bal felől nullás bitekkel. (ha szükséges)

Példák:

a.) A 19 belső ábrázolása 8 biten (unsigned char típusú változóban)
   1. lépés: $(19)_{10} = (10011)_2$
   2. lépés: 000**10011**

b.) A 0 belső ábrázolása 8 biten (unsigned char típusú változóban)
   1. lépés: $(0)_{10} = (0)_2$
   2. lépés: 0000000**0**

c.) A 255 belső ábrázolása 8 biten (unsigned char típusú változóban)
   1. lépés: $(255)_{10} = (11111111)_2$
   2. lépés: 11111111

RÁADÁS (2)

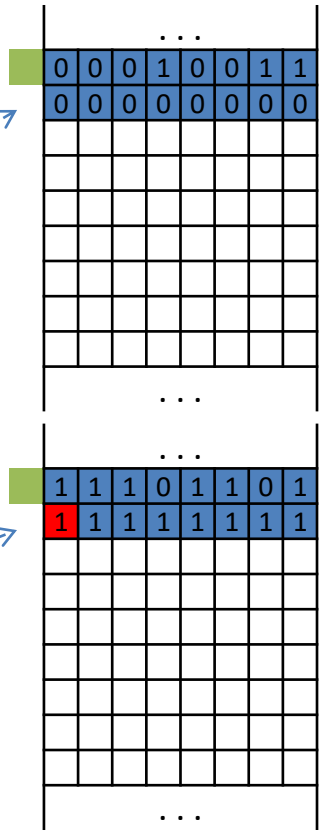
Típus	N	Értéktartomány
signed char	8	$-2^7 \dots 2^7 - 1$
signed short int	16	$-2^{15} \dots 2^{15} - 1$
signed long int	32	$-2^{31} \dots 2^{31} - 1$

- Előjeles egészek (signed) fix-pontos ábrázolása N biten

Ha $x < 0$, akkor
1. ábrázoljuk $|x|$-et N biten, mint előjel nélküli egészet.
2. képezzük a kapott N bites alaknak a 2-es komplemensét
a. minden 0 bitet 1-esre, és minden 1-es bitet 0-ra cserélünk. (1-es komplemens)
b. a kapott értékhez hozzáadunk 1-et.

a.) A 19 belső ábrázolása 16 biten (mint előjeles egésznek short int típusú változóban)
1. lépés: **0000000000010011**

b.) A (-19) belső ábrázolása 16 biten (mint előjeles egésznek short int típusú változóban)
1. lépés: a 19 belső ábrázolása 16 biten: 0000000000010011
2. lépés: Képezzük a fenti alak 2-es komplemensét
a. **1111111111101100**
b. **1111111111101101**



! előjel bit

RÁADÁS (3)

- Valós számok lebegőpontos belső ábrázolása

Típus	N	Pontosság
float (egyszerű pontosság)	32	6-7 tizedes
double (dupla pontosság)	64	15-16 tizedes
long double (bővített pontosság)	80	19 tizedes

Részletek végett lásd a JEGYZETET

(B. függelék, 261 oldal)

http://www.ms.sapientia.ro/~katai_zoltan

Teaching / Computer programming I (C - programming) [HU]