

# Számtípusok, műveletek

Kormányos Andor

Komplex Rendszerek Fizikája Tanszék

2022. szeptember 13.

# Számtípusok a C nyelvben

## Egész számok

- `char` - 1 bájt  
előjel nélküli egész
- `short` - 2 bájt  
előjeles egész
- `int` - 4 bájt  
előjeles egész
- `long` - 4 bájt  
előjeles egész
- `long long` - 8 bájt  
előjeles egész

## Előjeles/előjel nélküli egészek, pl.:

- `char` [0 : 256] vs  
`signed char`: [-127 : 127]
- `short`: [-32767 : 32767] vs  
`unsigned short`: [0 : 65535]

## Lebegőpontos számok (floating point)

- `float` - 4 bájt  
kb. 7 jegy pontos
- `double` - 8 bájt  
kb. 16 jegy pontos
- `long double` - 10 bájt  
kb. 19 jegy pontos (ritkán  
használt)

# Műveletek számokkal: összeadás, kivonás

Az eredmény típusa függ az operandusok típusától

- ha az operandusok azonos típusúak, az eredmény is olyan típusú lesz
- pl.: `int + int = int`
- pl.: `float + float = float`
  
- kivéve, ha valamelyik kevésbé pontos, mint `int`, mert akkor a fordító előbb `int`-re konvertál
- pl.: `short + char = signed int`
  
- kivéve, ha valamelyik pontosabb, mint a másik, mert akkor a fordító előbb a pontosabbra konvertál
- pl.: `int + double = double`
- pl.: `float + double = double`

# Műveletek számokkal: összeadás, kivonás, szorzás

Mi történik, ha kifutunk az ábrázolt tartományból?

- pl. két **short** számot összeadva 32756-nél nagyobb eredmény adódik?
- ilyenkor **csendes túlcsordulás** történik

```
1 int main() {
2     short a = 30000, b = 25000;
3     short s = a + b;
4     printf("%d\n", s); // eredmény: 14464
5     return 0;
6 }
```

Ilyenkor nem kapunk semmilyen “hibaüzenetet”

- fordításkor sincsen figyelmeztetés
- tudni kell róla, hogy ilyen történhet, és ha ez gond, akkor megfelelően ellenőrizni kell a számokat

# Műveletek egész számokkal: osztás

Az osztás kivezet az egész számok értékkészletéből

- ilyenkor nem történik automatikus típuskonverzió
- helyette: **egészosztás művelet lefelé kerekítéssel**

```
1 int main() {
2     int a = 12, b = 5;
3     printf("%d\n", a / b); // = 2
4     return 0;
5 }
```

Ha **lebegőpontos osztást** szeretnénk

- legalább az egyik operandus legyen **float** vagy **double**
- ehhez lehet, hogy **konvertálni** kell.

```
1 int main() {
2     int a = 12, b = 5;
3     printf("%f\n", (double)a / b); // = 2.400000
4     return 0;
5 }
```

# Típusok lefelé konvertálása

Ha különböző típusú operandusokon végzünk műveletet

- a fordító a pontosabb típus irányába konvertál
- ez a konverzió **implicit**, nem kell semmit sem kiírni

Mi a helyzet akkor, ha eredményül mi kevésbé pontos számot szeretnénk?

Pl egy **double** eredmény konvertálása **float**-ra, hogy kevesebb helyet foglaljon

# Típusok lefelé konvertálása

Ha különböző típusú operandusokon végzünk műveletet

- a fordító a pontosabb típus irányába konvertál
- ez a konverzió **implicit**, nem kell semmit sem kiírni

Mi a helyzet akkor, ha eredményül mi kevésbé pontos számot szeretnénk?

Pl egy **double** eredmény konvertálása **float**-ra, hogy kevesebb helyet foglaljon

- ilyenkor **explicit** konvertálás, azaz **cast**-olás szükséges
- pl. számolás után kerekítés, majd integerre konvertálás

```
1 int main() {
2     // osztás szabalyos kerekitessel
3     printf("%d\n", (int)(17.24 / 5.1 + 0.5));
4     // eredmeny: 3
5     return 0;
6 }
```

# Típusok lefelé konvertálása

Ha különböző típusú operandusokon végzünk műveletet

- a fordító a pontosabb típus irányába konvertál
- ez a konverzió **implicit**, nem kell semmit sem kiírni

Mi a helyzet akkor, ha eredményül mi kevésbé pontos számot szeretnénk?  
Pl egy **double** eredmény konvertálása **float**-ra, hogy kevesebb helyet foglaljon

- ilyenkor **explicit** konvertálás, azaz **cast**-olás szükséges
- pl. számolás után kerekítés, majd integerre konvertálás

```
1 int main() {
2     // osztas szabalyos kerekitessel
3     printf("%d\n", (int)(17.24 / 5.1 + 0.5));
4     // eredmeny: 3
5     return 0;
6 }
```

**Figyelem!**

- a **(int)** művelet csak a közvetlenül utána álló operandusra vonatkozik
- ha az egész kifejezést castolni kell, akkor zárójel szükséges