Típuskonverzió

Primitív típusok konverziója szélesítéssel

Ismétlésként: a *primitív típusok (primitive types)* azok a típusok, amelyeket eddig tanultunk (`int`, `float`, `double`, `char`…), ezekkel szemben lesznek majd az osztályok (class) (és tömbök - array).

Ha van egy kisebb számokat tárolni képes változónk, annak értéke eltárolható egy nagyobb értékek tárolására képes változóban. Ezt nevezzük *bővítés*nek, *szélesítés*nek, *automatikus típuskonverzió*nak vagy *implicit típuskonverzió*nak is.

Ha eltárolom például a 45-öt egy `byte` típusú változóban:

```
byte a = 45;
```

Később átadhatom az értéket egy `short`, `int`, `long`, stb. típusú változónak:

```
byte a = 45;
int b = a;
```

Ezt azért tehetem, mert a „b” változóban nagyobb szám tárolható, mint az „a” változóban. Fordítva azonban már nem lehetséges.

Az alábbiakban az egyes típusokat látjuk milyen *szélesítés* alkalmazható rajtuk:

- `byte` → `short`, `int`, `long`, `float`, `double`
- `short` → `int`, `long`, `float`, `double`
- `char` → `int`, `long`, `float`, `double`
- `int` → `long`, `float`, `double`
- `long` → `float`, `double`
- `float` → `double`

Másik példával:

```
byte a = 3;
short b = a;
```

A példában 3 számot az „a” változóban tároljuk először, azután szélesítést alkalmazunk meg „a” értékét egy olyan változóban tároljuk el, amely már nagyobb számok tárolására is képes. Vagyis, egy `byte` típusú változó tartalmát egy `short` típusúban tárolhatom, ehhez semmiféle konverzióra nincs szükség.

Primitív típusok konverziója szűkítéssel

Az értéket olyan változóban szeretném tárolni, amely kisebb számok tárolására alkalmas, mint az eredeti.

- `short` → `byte`, `char`
- `char` → `byte`, `short` (a `char` esetében inkább inkompatibilitásról beszélnék szűkítés helyett, mivel a karakter és a szám nem azonos értelmű, de a lényeg ugyanaz…)
- `int` → `byte`, `short`, `char`
- `long` → `byte`, `short`, `char`, `int`
- `float` → `byte`, `short`, `char`, `int`, `long`
- `double` → `byte`, `short`, `char`, `int`, `long`, `float`

Példa

```
short a = 5;
byte b = (byte) a;
```

A példánkban a `short` nagyobb számok tárolására alkalmas, így nem írhatjuk le:

```
b = a; // HIBÁS!
```

Helyette *típuskényszerítést (type casting)* használunk. A típuskényszerítés során zárójelbe tesszük annak a típusnak a nevét, amelybe szeretnénk átalakítani. Ha az egyenlőségjel baloldalán egy `byte` típus van, akkor zárójelbe azt írom `byte`. A típuskényszerítést *kasztolás, explicit típuskonverzió (explicit type conversion, explicit type cast)* névvel is használjuk.

A *kasztolás* egyébként az angol *casting* szóból magyarosodott(?). Eredeti formája: *„type casting”*. A C nyelvben terjedt el használata. Egy újabb példa a kasztolásra:

```
double a = 3;
int b = (int)a;
```

Az „a” változó értékét kasztoljuk `int` típussá.

Kérdések

- **Jól gondolom-e, hogy a szűkítésnél hiába használom a típuskényszerítést, ha a változó típus – amire szűkíteni akarok – nem tud akkora értéket tárolni, amekkorát bele szeretnék kényszeríteni, a program hibára fog futni?**
  Nem szakad meg a program, de jól se fog működni. Csonkítást hajt végre az értéken, és ami marad belőle, az lesz az új változó tartalma.
- **Szélesítésnek az az értelme, hogy ne ütközzek tárolási kapacitás korlátba?**
- **A szűkítésnek a memória használat csökkentése a célja? (vagy túlságosan előre gondolkodom ☺)**
  Általában akkor használjuk, ha külső rendszerekkel kommunikálunk, de memóriahasználat csökkentése is indokolhatja. De hangsúlyozom: *általában* egész számoknál `int` a nyerő. A többit kevésbé használjuk.