Lebegőpontos számábrázolás

Matematikai normálalak Javában

A lebegőpontos típusok (*floating point types* - **`float`** és **`double`**) elég nagy számok tárolását teszik lehetővé. Könnyedén el tudjuk tárolni pl. a Föld tömegét (akár grammban is), vagy Magyarország adósságállományának mértékét.

A Föld tömege 5 972 000 000 000 000 000 000 000 kg.

Ember nem ír le (pláne programban) ilyen hosszú számot, inkább rövidít:

$5{,}972 * 10^{24}$

Ha mindezt Javában akarjuk leírni, és betenni egy változóba, akkor ezt tehetjük:

```
double foldTomeg = 5.972E24;
```

Az **E** előtti rész a szorzótényező, mögötte pedig a 10 hatványa szerepel.

Lebegőpontosság

Alapja a matematikai normálalak:

$m*10^k$

Ez tizedestört és tízes számrendszer helyett kettes számrendszer és "kettedestört" esetén így fest:

$m*2^k$

Ebből azután tároljuk az *m*-et, mégpedig 1 és 2 közötti kettes számrendszerbeli törtszámként (a bevezető egyes nélkül), valamint a kitevőt (*k*), előjeles egész számként.

Hogyan írunk le egy 1 és 2 közötti, kettes számrendszerbeli törtszámot?

Úgy, hogy kezdünk egy fix 1-essel, majd a törtet úgy írjuk le, hogy a helyiértékek a következők lesznek: 2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}, azaz 0.5 (1/2), 0.25 (1/4), 0.125 (1/8), 0.0625 (1/16)… Ugyanúgy számítjuk ki az értéket, mint a pozitív egész számok esetén (tehát ugyanúgy szorozzuk össze a bitet a helyiértékkel), csak ezúttal a helyiértékek nem 1 és annál nagyobb egész számok, hanem 1-nél kisebb, pozitív törtszámok. (A példa segít)

Ha megvan a kettedestört, akkor melléírjuk a kitevőt, valamint még egy bitet előjelnek, és kész a szám. Így működik a 4 bájtos float és a 8 bájtos `double`.

Ha szeretnél játszadozni egy kicsit a lebegőpontos számok felépítésével, itt egy jó kalkulátor: https://www.h-schmidt.net/FloatConverter/IEEE754.html (exponent: *k, kitevő*; mantissa: *m*)

Példa

Egy táblázat segítségként:

helyiérték		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}		kitevő egész számként
bitek		1	1	0	1	0	0	0	* 2 ^	0 0 0 0 0 0 0 1
érték	1	0,5	0,25	0	0,0625	0	0	0	* 2 ^	1
érték	1,8125								* 2 ^	1
azaz	3,625									

Amennyiben a fenti linken játszadozol egy kicsit a számokkal (javaslom!), akkor feltűnhet, hogy a kalkulátor az exponenst (kitevő) nem ilyen szépen, egyszerűen tárolja, ahogyan a fenti példában van, hanem az exponens előjel nélküli értelmezéséből (legfelső bit értéke pozitív) még kivon 127-et, és úgy értelmezi a 2-es szám kitevőjeként. Ez részletkérdés, csak a pontosság kedvéért tartottam fontosnak megemlíteni.

A tárolás hátulütői

A normálalakszerű tárolásnak van néhány hátulütője, és **ez a fontos!**

- **nem tudjuk tetszőlegesen pontosan megközelíteni a 0-t (mert a kitevő minimuma meg van szabva).**
- **nem tudunk tetszőlegesen nagy számot tárolni, bár ezen nem lepődünk meg (amúgy azért nem, mert a kitevő legnagyobb értéke is meghatározott).**
- **nem tudunk minden számot tárolni (két "egymás melletti" tárolható törtszám között végtelen sok nem tárolt valós szám van. Műveletek esetén a számítógép kerekít...** nem intelligensen, hanem ahogy sikerül: az el nem tárolható biteket nem tárolja el.
    - Ez még tovább gondolkodva azt jelenti, hogy egy törtszámmal végzett művelet nem biztos, hogy a matematikai eredményt hozza.
    - Egyenlőségvizsgálat lehet, hogy hibás.
    - Helyette inkább azt használjuk, hogy a kapott eredmény és a várt eredmény különbségének abszolút értéke egy adott (pici) határnál kisebb:
      `| kapott - várt | < pici különbség`
    - jobb, ha egészet használunk pénzügyi mennyiségek tárolására is.
    - ha fontos a pontosság, akkor inkább egészeket tároljunk, és az utolsó lépésben számítsuk ki az egészből a törtszám eredményt.

Példák hibákra

a. A törtszámokkal végzett műveletek lehet, hogy hibás eredményt hoznak:

```
double y = 0.1+0.2;
System.out.println(y);
```

b. Egyenlőségvizsgálat lehet, hogy hibás:

```
double x = 0.15 + 0.15;
double y = 0.1 + 0.2;
System.out.println(x==y); // elméletileg igaz, gyakorlatilag hamis.
```

c. Egyenlőségvizsgálat helyesen (folytatva az előzőt):

```
System.out.println(Math.abs(x-y)<0.0000000001);
```

vagy ami ugyanezt jelenti:

```
System.out.println(Math.abs(x-y)<1e-10);
```