Dr. Holló Csaba, SZTE TTIK, Szoftverfejlesztés Tanszék

Programozási alapismeretek kurzus

Olvasási idő:
5 perc

Alapvető programozásnyelvi elemek C-ben és PHP-ban

III. Kifejezések kiértékelése

A lecke célja A kifejezések kiértékelésének mélyebb megértése, és ennek következtében bizonyos típushibák elkerülése

Tudás Ismeri a precedencia és az asszociativitás fogalmakat, illetve a műveletek elvégzésének folyamatát.

Képesség Helyesen értékeli ki a kifejezéseket.

Kifejezések kiértékelése

Két szempont szerint történik:

1. Műveletek prioritása

- a műveletek prioritási (precedencia) halmazokba vannak sorolva
- adott helyzetben (kifejezésben) a nagyobb prioritású műveletek hamarabb elvégződnek
- pl. a vonatok közlekedésének is elsőbbsége van

https://pixabay.com/hu/photos/auto-vonat-priorit%C3%A1s-3034401/

2. Műveletek asszociativitása

- azonos prioritású egymás utáni műveletek milyen sorrendben végződjenek el

Nézzük meg a műveletek precedenciáját és asszociativitását a következő oldalak valamelyikén:

- Wikipédia, BME,
- Aszódi Evangélikus Petőfi Gimnázium, Általános Iskola és Kollégium

Kifejezések kiértékelése: példa C-ben

Legyen int x, y;. Mi lesz az x = y = 7; eredménye?

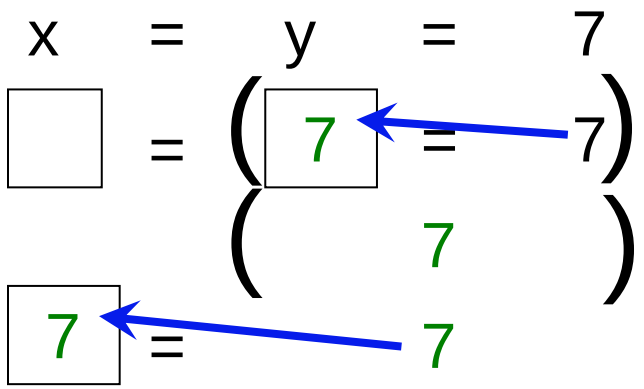
A két = művelet precedenciája egyenlő, a műveletek sorrendjét az asszociativitás határozza meg.

Az = jobbról balra asszociatív, ezért

x = y = 7; → x = (y = 7);, továbbá

y = 7 eredménye az értékül adott érték,

ezért x = (y = 7); → x = 7;



Hogyha az = balról jobbra asszociatív lenne, akkor

- először elvégeznénk az x = y; értékadást és
- annak az eredményének (ami nem egy változó, hanem egy érték) szeretnénk értékül adni a 7-et → ami értelmetlen lenne.

Kifejezések kiértékelése PHP-ban

Míg C-ben minden műveletnek meg van adva az asszociativitása,

PHP-ban vannak olyan művelethalmazok, melyeknél ez nem definiált, azaz „nem köthető”.

Nem köthető műveletek például: <, <=, >, >=, ==, !=

További részleteket a Sapiencia Egyetem oldalán találunk.

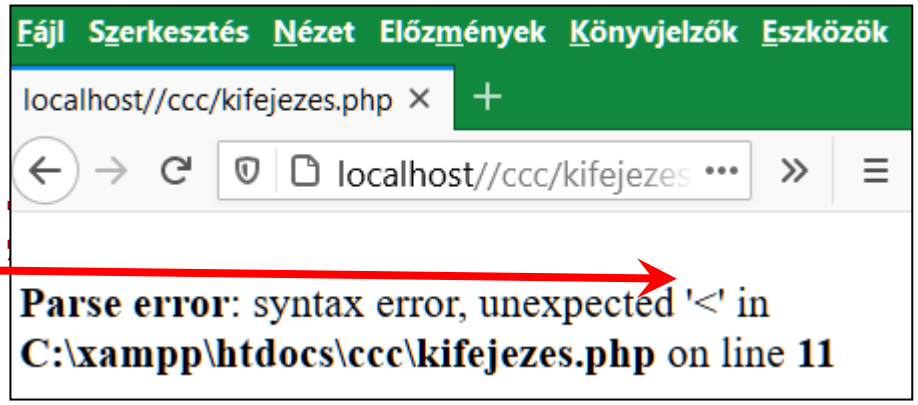
PHP-ban a nem köthető egymás utáni műveleteknél hibajelzést kapunk.

Kifejezések kiértékelése PHP-ban: példa

PHP-ban a változók nevét $-al kezdjük.

A következő programrészlet:

```
$x = 3;
if(4 < $x <6) echo "OK";
else echo "Nem OK";
```



hibajelzést fog kiírni, melynek oka, hogy

- a < művelet nem köthető, ezért
- nem lehet két < művelet egymás után.

Kérdések, feladatok

1. Legyen int x = 3, y= 8; Mennyi lesz az x értéke az x *= y /= 2; kifejezés kiértékelése után?

Megoldás. Mivel az értékadó operátorok jobbról balra asszociatívak, ezért:

x *= y /= 2; → x *= (y /= 2); de

y /= 2 → y = y / 2 → y = 4

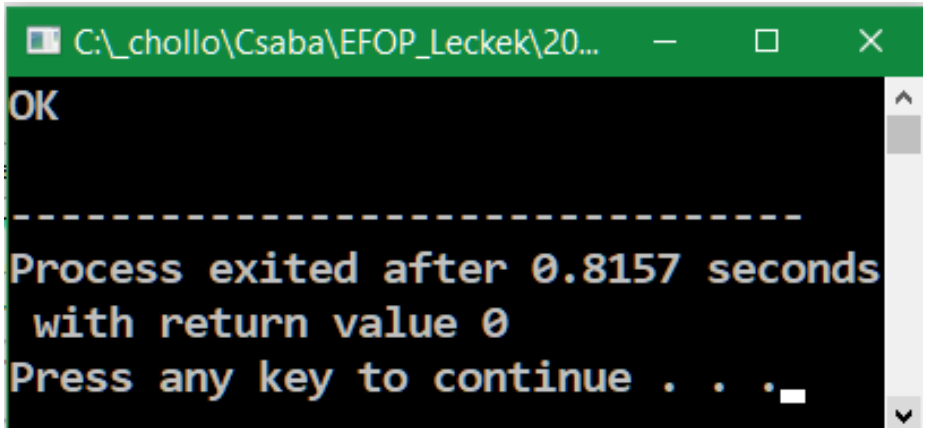
- az értékadás eredménye az értékül adott érték, azaz 4

x *= (y /= 2); → x *= 4; → x = x * 4; → x = 12;

Kérdések, feladatok

2. Mit fog kiírni a megfelelő programrészlet C-ben?

```
int x = 3;
if(4 < x < 6) printf("OK");
else printf("Nem OK");
```



De miért? Azért, mert (4 < x < 6):

- nem azt vizsgálja, hogy x értéke 4 és 6 között van-e, hanem
- egymás után két < műveletet végez el.

Kérdések, feladatok

A < művelet asszociativitásának megfelelően

(4 < x < 6) kiértékelése ((4 < x) < 6)

(4 < x), ez hamis, ami C-ben 0,

tehát ((4 < x) < 6) → (0 < 6), ami igaz, tehát

x = 3-ra: (4 < x < 6) **IGAZ**!

De akkor hogyan vizsgáljuk, hogy x értéke

4 és 6 között van-e? (4<x && x< 6)

Kérdések, feladatok

3. Milyen x értékekre lesz igaz C-ben a (4 > x > -1) kifejezés kiértékelése?

Megoldás

- Minden 4 vagy annál nagyobb x értékre a 4 > x hamis, tehát 0, ezért arra (4 > x > -1) → (0 > -1) igaz.
- 4-nél kisebb x értékekre a 4 > x igaz, így (4 > x > -1) → (igaz > -1), tehát a végeredmény attól függ, hogy az igaz implementálása nagyobb-e, mint -1.
- Az igaz implementálása többnyire 1 szokott lenni, ha ez nálunk is így van, akkor tehát a (4 > x > -1) kifejezés bármilyen x esetén igazra értékelődik ki.

EFOP-3.4.3-16-2016-00014

AP1 HALLGATÓI DIPLOMA-SZERZÉST SEGÍTŐ SZOLGÁLTATÁSOK

Jelen tananyag a
Szegedi Tudományegyetemen
készült az
Európai Unió támogatásával.
Projekt azonosító:
EFOP-3.4.3-16-2016-00014