Léptető (és módosító) operátorok

Ha egy változó értékét meg akarjuk növelni eggyel, akkor arra a klasszikus megoldás az, hogy beírjuk

```
i = i + 1
```

Módosító operátorok (assignment operators)

Ezt azért lehet rövidebben is írni, így:

```
i += 1
```

melynek jelentése: növeljük meg az `i` változó értékét eggyel.

A legtöbb önálló operátorra létezik olyan verzió, ami módosításra alkalmazható, pl. `-=`, `/=`, `%=`, `*=` stb. Ezek mind a fenti módon értelmezendők.

Léptető operátorok (increment and decrement operators)

Az eggyel való növelést tovább rövidíthetjük:

```
i++
```

```
++i
```

Mindkettő megnöveli az `i` változó értékét eggyel.

Mi a különbség?

Ha az `i++` és a `++i` önállóan áll egy kifejezésben, akkor nem látszik semmi különbség. Akkor lehet felfedezni az eltérést, ha egy összetett kifejezés része a növelés. Pl.

```
a = i++;
```

```
a = ++i;
```

Ebben az esetben ugyanis az `i++` és `++i` kifejezések értékeit használjuk fel, mint az érték, amit az értékadás esetén átadunk az `a` változónak.

A két növelés között az a különbség, hogy a növelés előtti vagy utáni értéket használja-e, mint a saját értéke. (A növelés ugyanis mindkét esetben megtörténik).

Ha az `i++`-t választjuk, és kezdetben `i` értéke `8` volt, akkor az `i++` kifejezés értéke `8` lesz (és megnöveli az `i`-t `9`-re).

Ha a `++i`-t írjuk le, és kezdetben az `i` értéke 8 volt, akkor a `++i` kifejezés értéke 9 lesz (és persze megnövelte az `i` értékét 9-re).

Kicsit másképp fogalmazva:

Ha `i` változó értéke kezdetben 8, akkor `a = i++` esetén először megtörténik az értékadás, majd ezután az `i` értékének növelése, azaz `i` változó új értéke 9, az `a` változó értéke 8 (az `i` növelés előtti értéke) lesz. (Vizuálisan is így van sorban: `a = ++` a sor végén van, tehát azt a többi „után” hajtjuk végre).

Az `a = ++i` esetén először megnöveljük az `i`-t, majd ezt a növelés utáni értéket adjuk át az `a`-nak, így az `i` változó értéke 9 és az `a` változó értéke szintén 9 lesz (vizuálisan: a `++` az `i` előtt van, így az `i` értékének felhasználása „előtt” történik meg a megnövelés).

A `++i` először növel, utána átadja az értékét, az `i++` először átadja az értékét, utána növeli az `i` értékét.

Másik példa:

```
System.out.println(i++);
System.out.println(++i);
```

Az előbbi sor kiírja az `i` változó régi értékét, majd növeli eggyel. Az utóbbi előbb növeli eggyel az `i` értékét, majd kiírja.

Ha még mindig nem világos, akkor írj egy-két próbaprogramot, és vizsgálgasd azt, hogy mi történik.

Összefoglalva

Az `i++` és `++i` jellegű kifejezéseknél két dolog történik: 1.) megnő az `i` értéke, 2.) kiszámítódik a kifejezés értéke (ami rendre a növelés előtti vagy utáni érték).

A Java (és a C-stílusú nyelvek) a 2.)-t tekintik főhatásnak (hiszen ettől lesz a kifejezésnek értéke), az 1.)-est pedig mellékhatásnak (side effect).