**Increment and decrement.**

(++, --)

Some expression can be shortened even more: the increase operator (++) and the decrease operator (--) increase or reduce by one the value stored in a variable. They are equivalent to +=1 and to -=1, respectively. Thus:

|  |  |  |
| --- | --- | --- |
| 1 2 3 | ++x;  x+=1;  x=x+1; |  |

are all equivalent in its functionality; the three of them increase by one the value of *x*.

In the early C compilers, the three previous expressions may have produced different executable code depending on which one was used. Nowadays, this type of code optimization is generally performed automatically by the compiler, thus the three expressions should produce exactly the same executable code.

A peculiarity of this operator is that it can be used both as a prefix and as a suffix. That means that it can be written either before the variable name (++x) or after it (x++). Although in simple expressions like *x++* or *++x*, both have exactly the same meaning; in other expressions in which the result of the increment or decrement operation is evaluated, they may have an important difference in their meaning: In the case that the increase operator is used as a prefix (++x) of the value, the expression evaluates to the final value of *x*, once it is already increased. On the other hand, in case that it is used as a suffix (x++), the value is also increased, but the expression evaluates to the value that *x* had before being increased. Notice the difference:

|  |  |
| --- | --- |
| **Example 1** | **Example 2** |
| x = 3; y = ++x; // x contains 4, y contains 4 | x = 3; y = x++; // x contains 4, y contains 3 |

In *Example 1*, the value assigned to *y* is the value of *x* after being increased. While in *Example 2*, it is the value *x* had before being increased.